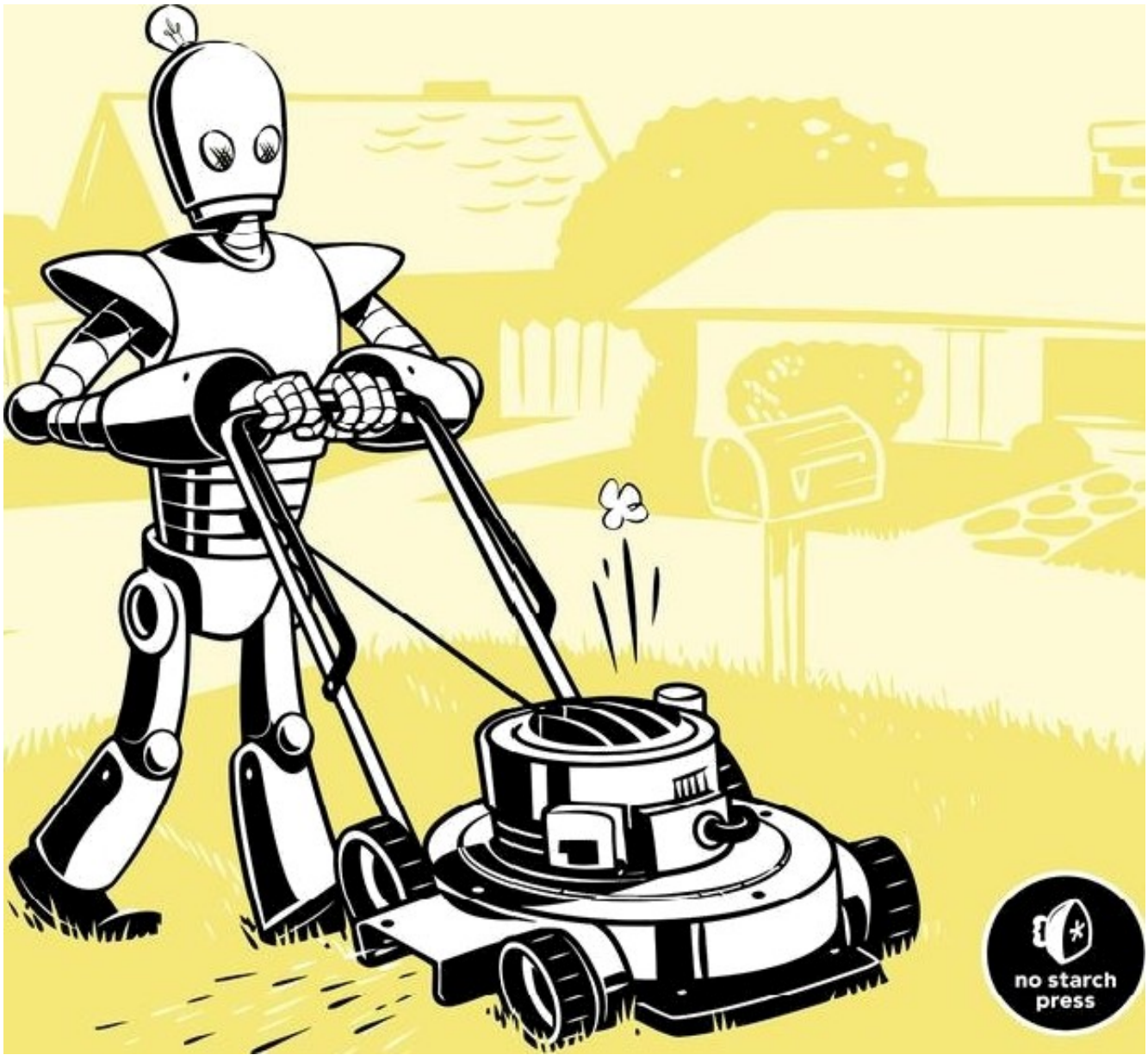
AUTO MATE LO ABURRIDO CON RVTHON

PRACTICAL PROGRAMMING
FOR TOTAL BEGINNERS

AL SWEIGART



Automatice las cosas aburridas con Python: Programación práctica para principiantes

Al Sweigart

Publicado por No Starch Press

ÍNDICE

Agradecimientos Introducción

PARTE I: FUNDAMENTOS DE PROGRAMACIÓN PYTHON

Capítulo 1: Conceptos básicos

de Python Capítulo 2: Control

de flujo Capítulo 3: Funciones

Capítulo 4: Listas

Capítulo Capítulo 6: Manipulación de cadenas de

texto5: Diccionarios y estructuración de datos

PARTE II: AUTOMATIZACIÓN DE TAREAS

Capítulo 7: Comparación de patrones con expresiones

regulares Capítulo 8: Lectura y escritura de archivos

Capítulo 9: Organización de

archivos Capítulo 10:

Depuración Capítulo 11: Web

Scraping

Capítulo Capítulo 13: Trabajar con documentos PDF y

Word 12: Trabajar con hojas de cálculo Excel Capítulo 14:

Trabajar con archivos CSV y datos JSON

Capítulo Capítulo 16: Enviar correo electrónico y mensajes de texto15: Llevar la

cuenta del tiempo, programar tareas e iniciar programas

Capítulo 17: Manipulación de imágenes

Capítulo 18: Control del teclado y el ratón con la automatización GUI Apéndice A: Instalación de módulos de terceros

Apéndice B: Ejecución de programas

Apéndice C: Respuestas a las preguntas prácticas

Índice

CONTENIDOS EN DETALLE

AGRADECIMIENTOS

INTRODUCCIÓN
¿A quién va dirigido este libro?
Convenciones
¿Qué es la programación?
¿Qué es Python?
Los programadores no necesitan saber muchas matemáticas Programar es una actividad creativa
Acerca de este libro
Descarga e instalación de Python Inicio de IDLE
La concha interactiva
Cómo encontrar ayuda
Preguntas inteligentes sobre programación
Resumen

PARTE I: FUNDAMENTOS DE PROGRAMACIÓN PYTHON

1
PITÓN BÁSICO
Introducción de expresiones en el intérprete de comandos Tipos de datos enteros, de coma flotante y de cadenas Concatenación y replicación de cadenas
Almacenamiento de valores en variables
Sentencias de asignación
Nombres de variables
Su primer programa Disección de su programa
Comentarios
La función print()

La La función len()función
input() Impresión del nombre
del usuario
Resumen de las funciones str(), int() y
float()
Preguntas prácticas

2
CONTROL DE CAUDAL
Valores booleanos
Operadores de
comparación Operadores
booleanos
Operadores booleanos binarios
El operador not
Mezcla de operadores booleanos y de
comparación Elementos de control de flujo
Condiciones
Bloques de código
Declaraciones de control de
flujo de ejecución del
programa
if Declaraciones
else
Declaraciones elif
Declaraciones
while Loop Sentencias break
Sentencias continue
Sentencias
Bucles for y la función range() Importación
de módulos
from import Statements
Finalización anticipada de un programa
con sys.exit() Resumen
Preguntas prácticas

3

FUNCIONES
def Sentencias con parámetros Valores de retorno y sentencias return El valor None
Palabra clave Argumentos y print()
Ámbito local y global
Las variables locales Las variables globales pueden leerse desde un ámbito locallocales no pueden utilizarse en el ámbito global Los ámbitos locales no pueden utilizar variables de otros ámbitos
Variables locales y globales con el mismo nombre La declaración global
Gestión de excepciones
Un programa corto: Adivina el número Resumen
Preguntas prácticas
Proyectos prácticos
Validación de entrada de la secuencia de Collatz

4
LISTAS
El tipo de datos List
Obtención de valores individuales en una lista con índices Índices negativos
Obtención de sublistas con Slices
Obtención de la longitud de una lista con len()
Modificación de valores en una lista con índices Concatenación de listas y replicación de listas
Eliminar valores de listas con sentencias del Trabajar con listas
Uso de bucles for con listas Los operadores in y not in El truco de la asignación múltiple

Métodos de los operadores de asignación aumentada
- Encontrar un valor en una lista con el método index()
- Añadir valores a listas con los métodos append() e insert() Eliminar valores de listas con remove()
- Ordenar los Valores de una Lista con el Método sort()

Programa de Ejemplo: Bola 8 Mágica con una Lista
Tipos de lista: Cadenas y tuplas Tipos de datos mutables e inmutables El tipo de datos tupla
- Conversión de tipos con las funciones list() y tuple() Referencias
- Pasar referencias
- Resumen de las funciones copy() y deepcopy() del módulo copy

Preguntas prácticas
Proyectos prácticos
- Código Coma Carácter
- Cuadrícula

5
DICCIONARIOS Y ESTRUCTURACIÓN DE DATOS
El tipo de datos diccionario
- Diccionarios frente a listas
- Los métodos keys(), values() y items()
- Comprobación de la existencia de una clave o un valor en un diccionario Método get()
- El método setdefault() Pretty

Printing
Uso de estructuras de datos para modelar cosas del mundo real Un tablero de tres en raya
- Resumen de diccionarios y listas anidados

Preguntas prácticas
Proyectos prácticos
Inventario de juegos de fantasía
Función de lista a diccionario para el inventario de juegos de fantasía

6
MANIPULACIÓN DE CADENAS
Trabajar con cuerdas
Literales de cadena
Indexación y segmentación de cadenas
Los operadores in y not in con cadenas Métodos útiles para cadenas
Métodos de cadena upper(), lower(), isupper() e islower(Métodos de cadena isX
Los métodos de cadena startswith() y endswith()
métodos de cadena join() y split()
Justificación de texto con rjust(), ljust() y center() Eliminación de espacios en blanco con strip(), rstrip() y lstrip() Copia y pegado de cadenas con el módulo pyperclip
Proyecto Password Locker
Paso Paso 2: Manejo de los argumentos de la línea de comandos 1: Diseño de programas y estructuras de datos Paso 3: Copia de la contraseña correcta
Proyecto: Añadir viñetas al marcado wiki
Paso 1: Copiar y pegar desde el portapapeles
Paso 2Paso 3: Unir las líneas modificadas: Separar las líneas de texto y añadir la estrella
Resumen
Preguntas prácticas
Proyecto práctico
Impresora de mesa

PARTE II: AUTOMATIZACIÓN DE TAREAS

7
CONCORDANCIA DE PATRONES CON EXPRESIONES REGULARES
Búsqueda texto sin expresiones regulares Búsqueda de patrones de de patrones de texto con expresiones regulares
Creación de objetos Regex
Correspondencia de objetos Regex
Revisión de la concordancia de expresiones regulares Más concordancia de patrones con expresiones regulares
Agrupación con paréntesis
Coincidencia Coincidencia cero o más con la estrella de varios grupos con el tubo
Coincidencia opcional con el signo de interrogación de Coincidencia de uno o más con el signo más
Emparejamiento de repeticiones específicas con paréntesis rizados Emparejamiento codicioso y no codicioso El método findall() Clases de caracteres
Cómo crear sus propias clases de caracteres Los caracteres signo de interrogación y signo de dólar El carácter comodín
Coincidencia de el carácter de punto
Coincidencia de nuevas líneas con el carácter de puntotodo con
Revisión de los símbolos
Regex Coincidencia sensible a mayúsculas y minúsculas
Sustitución de cadenas con el método sub()
Gestión de expresiones regulares complejas
Proyecto : Extractor de números de teléfono y direcciones de correo electrónicoque combina re.IGNORECASE, re.DOTALL y re.VERBOSE
Paso Paso 2: Crear una Regex para direcciones de correo electrónico 1: Crear una Regex para números de teléfono Paso 3: Buscar todas las coincidencias en el texto del portapapeles
Paso 4: Unir las coincidencias en una cadena para el portapapeles que ejecuta el programa

Ideas para programas similares
Resumen
Preguntas prácticas
Proyectos prácticos
Versión regex de strip() para la detección de contraseñas seguras

8
LECTURA Y ESCRITURA DE ARCHIVOS
Archivos y rutas de archivos
Barra invertida en Windows y barra diagonal en OS X y Linux El directorio de trabajo actual
Trayectorias absolutas frente a trayectorias relativas
Creación de nuevas carpetas con os.makedirs()
El módulo os.path
Manejo de rutas absolutas y relativas Búsqueda de tamaños de archivos y contenidos de carpetas Comprobación de la validez de la ruta
El proceso de lectura y escritura de archivos
Abrir archivos con la función open() Leer el contenido de los archivos
Escribir en archivos
Guardar variables con el módulo shelve
Guardando Variables con la Función pprint.pformat() Proyecto: Generación de archivos de prueba aleatorios
Paso 1: Almacenar los datos del cuestionario en un diccionario
Paso 2: Paso 3: Crear las opciones de respuestaCrear el archivo del cuestionario y barajar el orden de las preguntas
Paso 4: Redactar el contenido del cuestionario y los archivos de claves de respuesta Proyecto: Multiclipboard
Paso 1: Comentarios y configuración de la estantería
Paso 2: clave Paso 3: Listar palabras clave y cargar el contenido de una palabra claveGuardar el contenido del portapapeles con una palabra

Resumen
Preguntas prácticas
Proyectos prácticos
Ampliación del tablero multiclip
Mad Libs
Búsqueda Regex

9
ORGANIZAR ARCHIVOS
El módulo shutil
Copiar archivos y carpetas
Mover y renombrar archivos y carpetas Borrado
permanente de archivos y carpetas Borrados
seguros con el módulo send2trash
Recorrer un árbol de directorios
Compresión de archivos con el módulo zipfile
Lectura de archivos ZIP
Extracción de archivos ZIP Creación
y adición a archivos ZIP
Proyecto: Renombrar archivos con fechas de estilo americano a fechas de estilo europeo
Paso Paso 2: Identificar las Partes de Fecha de los Nombres
de Archivo 1: Crear un Regex para Fechas de Estilo
Americano Paso 3: Formar el Nuevo Nombre de Archivo y
Renombrar los Archivos Ideas para Programas Similares
Proyecto: Copia de seguridad de una carpeta en un
archivo ZIP Paso 1: Averiguar el nombre del
archivo ZIP Paso 2: Crear el nuevo archivo
ZIP
Paso 3: Recorrer el árbol de directorios y añadir al archivo ZIP
Ideas para programas similares
Resumen
Preguntas prácticas
Proyectos prácticos
Copia selectiva

Eliminar archivos innecesarios
Rellenar huecos

10
DEBUGGING
Planteamiento de excepciones
Obtener el Traceback como una Cadena
de Aserciones
Uso de una aserción en una simulación de semáforo
Desactivación de aserciones
Registro
Uso del módulo logging No
depurar con print() Niveles
de registro Desactivación
del registro Registro en un
archivo
Depurador de
IDLEGo
Step
Over
Out
Quit
Depuración de un número Añadir del programapuntos
de interrupción
Resumen
Preguntas prácticas
Proyecto práctico
Depurar el lanzamiento de monedas

11
RASPADO WEB
Proyecto: mapIt.py con el módulo webbrowser Paso
1: Averiguar la URL
Paso 2: Manejar los argumentos de la línea de comandos

Paso 3: Manejar el Contenido del Portapapeles y Lanzar las del Navegador Ideas para Programas Similares
Descarga de archivos de la Web con el módulo requests Descarga de una página Web con la función requests.get() Comprobación de errores
Guardar archivos descargados en el disco duro
HTML
Recursos para aprender HTML
Un repaso rápido
Ver el código HTML fuente de una página Web Abrir las herramientas de desarrollo del navegador
Uso de las herramientas de desarrollo para buscar elementos HTML Análisis de HTML con el módulo BeautifulSoup
Creación de un objeto BeautifulSoup a partir de HTML Búsqueda de un elemento con el método select() Obtención de datos de los atributos de un elemento
Proyecto: "Me siento afortunado" Búsqueda en Google
Paso 1: Obtener los argumentos de la línea de comandos y solicitar la página de búsqueda
Paso 2: Buscar todos los resultados
Paso 3: Abrir los navegadores para cada resultado
Ideas para programas similares
Proyecto: Descargar todos los cómics de XKCD Paso 1Diseñar el programa :
Paso 2: Descargar la página web
Paso 3: Buscar y descargar la imagen del cómic
Paso 4: Guarde la imagen y busque cómic anteriores las ideas de para programas similares
Control del navegador con el módulo selenium Inicio de un navegador controlado por selenium Búsqueda de elementos en la página
Hacer clic en la página
Cumplimentación y envío de formularios

Envío de teclas especiales
Pulsación de botones del navegador Más información sobre Selenium
Resumen
Preguntas prácticas
Proyectos prácticos
Línea de comandos Emailer Image Site Downloader 2048
Verificación de enlaces

12
TRABAJAR CON HOJAS DE CÁLCULO EXCEL
Documentos Excel
Instalación del módulo openpyxl Lectura de documentos Excel
Abriendo Documentos Excel con OpenPyXL
Obteniendo Hojas del Libro Obteniendo Celdas de las Hojas
Conversión entre Letras y Números de Columna Obtención de Filas y Columnas de las Hojas Libros de Trabajo, Hojas, Celdas
Proyecto: Lectura de datos de una hoja de cálculo
Paso 1: Leer los datos de la hoja de cálculo Paso 2: Rellenar la estructura de datos Paso 3: Escribir los resultados en un archivo Ideas para programas similares
Redacción de documentos Excel
Creación Creación y Eliminación de Hojas y Guardado de Documentos Excel Escritura de Valores en Celdas
Proyecto: Actualización de una hoja de cálculo
Paso 1: Configurar una estructura de datos con la información de actualización

Paso 2: Comprobar todas las filas y actualizar los precios incorrectos Ideas para programas similares
Configuración del estilo de fuente de las celdaslos objetos de fuente de Fórmulas
Ajuste de filas y columnas
Ajustar el Alto de Fila y el Ancho de Columna
Fusionar y Deshacer Celdas
Paneles de congelación
Resumen de gráficos
Preguntas prácticas
Proyectos prácticos
Creador de tablas de multiplicar
Insertor de filas en blanco Hoja de cálculo Inversor de celdas
Archivos de texto a hoja de cálculo Hoja de cálculo a archivos de texto

13
TRABAJAR CON DOCUMENTOS PDF Y WORD
Documentos PDF
Extraer texto de PDF
Descifrar PDF Crear PDF
Proyecto: Combinar páginas seleccionadas de varios PDF
Paso 1: Buscar todos los archivos PDF
Paso 2: Abrir cada PDF Paso 3: Añadir cada página Paso 4: Guardar los resultados Ideas para programas similares
Documentos Word
Lectura de documentos Word

Obtención del texto completo de un archivo .docx Estilización de objetos de párrafo y de ejecución
Creación de documentos Word con estilos no predeterminadosatributos de ejecución de
Escribir documentos de Word
Añadir títulos
Añadir saltos de línea y de página
Añadir imágenes
Resumen
Preguntas prácticas
Proyectos prácticos
PDF Paranoia
Invitaciones personalizadas como documentos de Word Brute-Force PDF Password Breaker

14
TRABAJAR CON ARCHIVOS CSV Y DATOS JSON
El módulo csv
Objetos de lectura
Lectura de datos de objetos de lectura en un bucle for
Objetos de escritura
El delimitador y el lineterminador Argumentos de palabras clave
Proyecto: Eliminación de la cabecera de los archivos CSV
Paso Paso 2: Leer el archivo CSV1:
Recorrer cada archivo CSV
Paso 3: Escriba el archivo CSV sin la primera fila Ideas para programas similares
JSON y API El módulo json
Lectura de JSON con la función loads() Escritura de JSON con la función dumps()
Proyecto: Obtención de datos meteorológicos actuales
Paso 1: Obtener la ubicación del argumento de la línea de comandos

Paso 2: Descargar los datos JSON
Paso 3: Cargar datos JSON e imprimir
meteorológicas ideas para programas similares
Resumen
Preguntas prácticas
Proyecto práctico
Conversor de Excel a CSV

15
LLEVAR LA CUENTA DEL TIEMPO, PROGRAMAR TAREAS E INICIAR PROGRAMAS
El módulo de tiempo
función time.time() La
función time.sleep()
Proyecto de : Súper
cronómetroredondeo de
números
Paso 1: Paso 2: Controlar e imprimir los tiempos
por vueltaConfigurar el programa para controlar los
tiempos
Ideas para programas similares
El módulo datetime
El tipo de datos timedelta Pausa
hasta una fecha específica
Conversión objetos en cadenas Conversión de datetime
de cadenas en objetos datetime
Revisión de las funciones temporales de Python
Multithreading
Paso de argumentos a la función de destino del subproceso
Problemas de concurrencia
Proyecto: Descargador XKCD multihilo
Paso Paso 2: Crear e iniciar subprocesos1: Modificar el
programa para utilizar una función
Paso 3: Esperar a que todos los hilos
finalicen el lanzamiento de otros programas desde
Python
Pasar argumentos de línea de comandos a Popen()

Programador de tareas, launchd y cron
Abrir sitios web con Python Ejecutar otros scripts de Python Abrir archivos con aplicaciones predeterminadas
Proyecto: Programa simple de cuenta atrás Paso 1: Cuenta atrás
Paso 2: Reproducir el archivo de sonido Ideas para programas similares
Resumen
Preguntas prácticas
Proyectos prácticos
Cronómetro embellecido
Descargador programado de Webcómics

16
ENVÍO DE CORREO ELECTRÓNICO Y MENSAJES DE TEXTO
SMTP
Envío de correo electrónico
Conexión a un servidor SMTP Envío del mensaje SMTP "Hola" Inicio del cifrado TLS
Inicio de sesión en el servidor SMTP Envío de un correo electrónico
Desconexión del servidor SMTP
IMAP
Recuperación y eliminación de correos electrónicos con IMAP Conexión a un servidor IMAP Inicio de sesión en el servidor IMAP Búsqueda de correo electrónico
Obtención de direcciones de correo electrónico de un mensaje sin procesar Obtención del cuerpo de un mensaje sin procesar Eliminación de mensajes de correo electrónico

Desconexión del servidor IMAPproyecto de :
Envío de recordatorios de cuotas
Paso Paso 2: Buscar todos los socios
impagados1: Abrir el archivo Excel
Paso 3: Enviar recordatorios personalizados por correo
electrónico Enviando mensajes de texto con Twilio
Suscripción a una cuenta Twilio Envío de
mensajes de texto
Proyecto: "del módulo "Just Text
MeResumen
Preguntas prácticas
Proyectos prácticos
Recordatorio de tareas aleatorias correo
electrónico
Baja automática
Control del ordenador a través del correo electrónico

17
MANIPULACIÓN DE IMÁGENES
Fundamentos de la Imagen por
Ordenador Colores y Valores
RGBA Coordenadas y Tuplas de
Caja
Manipular Imágenes con Pillow Trabajar con el
Tipo de Datos de Imagen Recortar
Imágenes
Copiar y pegar imágenes en otras imágenes Cambiar el
tamaño de una imagen
Girar y voltear imágenes Cambiando
píxeles individuales
Proyecto: Añadir un logotipo
Paso 1: Abrir la imagen del logotipo
Paso 2Paso 3: Cambiar el tamaño de las imágenes:
Recorrer todos los archivos y abrir las imágenes

Paso 4: Añadir el logotipo y guardar los cambios Ideas para programas similares
Dibujar sobre imágenes
Dibujar formas Dibujar texto
Resumen
Preguntas prácticas
Proyectos prácticos
Ampliación y reparación de los programas de proyectos de capítulos Identificación de carpetas de fotos en el disco duro Tarjetas de asiento personalizadas

18
CONTROL DEL TECLADO Y EL RATÓN CON AUTOMATIZACIÓN GUI
Instalar el módulo pyautogui Mantener el rumbo
Apagar todo desconectando pausas y sistemas de seguridad
Controlar el movimiento del ratón Mover el ratón
Obtener la posición del ratón Proyecto: "¿Dónde está el ratón ahora mismo?"
Paso 1: Importar el módulo
Paso 2: Paso 3: Obtener e imprimir las coordenadas del ratónConfigurar el código de salida y el bucle infinito
Control de la interacción del ratón
Hacer clic con el ratón
Arrastrar el ratón Desplazar el ratón
Trabajar con la pantalla Obtener una captura de pantalla Analizar la captura de pantalla
Proyecto: Ampliación del programa mouseNow

Reconocimiento de imágenes
Control del teclado
Envío de una cadena a partir de del tecladolos nombres de las teclas
Pulsar y soltar las del tecladocombinaciones de teclas de acceso rápido
Revisión del Funciones
PyAutoGUIProyecto : Rellenador Automático de Formularios
Paso Paso 3: Empieza a introducir datos1: Calcula los pasos Paso 2: Configura las coordenadas
Paso 4Paso 5: Enviar el formulario y esperar: Manejar listas de selección y botones de radio
Resumen
Preguntas prácticas
Proyectos prácticos
Ocupado
Tutorial para jugar al bot de mensajería instantánea

A
INSTALACIÓN DE MÓDULOS DE TERCEROS
La herramienta pip
Instalación de módulos de terceros

B
PROGRAMAS EN EJECUCIÓN
Línea Shebang
Ejecutar programas PyEjecutar programas Python en OS X y Linuxthon en Windows
Ejecutar programas Python con aserciones desactivadas

C
RESPUESTAS A LAS PREGUNTAS PRÁCTICAS

Capítulo 1
Capítulo 2
Capítulo 3
Capítulo 4
Capítulo 5
Capítulo 6
Capítulo 7
Capítulo 8
Capítulo 9
Capítulo 10
Capítulo XI
Capítulo 12
Capítulo 13
Capítulo 14
Capítulo 15
Capítulo 16
Capítulo 17
Capítulo 18

ÍNDICE

Para mi sobrino Jack

Sobre el autor

Al Sweigart es un desarrollador de software y autor de libros de tecnología que vive en San Francisco. Su lenguaje de programación favorito es Python, para el que ha desarrollado varios módulos de código abierto. Sus otros libros están disponibles gratuitamente bajo licencia Creative Commons en su sitio web http://www.inventwithpython.com/. Su gato pesa cuatro kilos.

Acerca del revisor técnico

Ari Lacenski es desarrollador de aplicaciones Android y software Python. Vive en San Francisco, donde escribe sobre programación Android en *http://gradlewhy.ghost.io/* y ejerce de mentora en Women Who Code. También es guitarrista folk.

Agradecimientos

No podría haber escrito un libro como éste sin la ayuda de mucha gente. Me gustaría dar las gracias a Bill Pollock; a mis editores, Laurel Chun, Leslie Shen, Greg Poulos y Jennifer Griffith-Delgado; y al resto del personal de No Starch Press por su inestimable ayuda. Gracias a mi revisor técnico, Ari Lacenski, por sus estupendas sugerencias, ediciones y apoyo.

Muchas gracias a nuestro Benevolente Dictador Vitalicio, Guido van Rossum, y a todos en la Python Software Foundation por su gran trabajo. La comunidad Python es la mejor que he encontrado en la industria tecnológica.

Por último, me gustaría dar las gracias a mi familia, a mis amigos y a la pandilla de Shotwell's por no importarme la ajetreada vida que he llevado mientras escribía este libro. Salud.

Introducción

"Acabas de hacer en dos horas lo que a nosotros tres nos lleva dos días". Mi compañero de universidad trabajaba en una tienda de electrónica principios de la década de 2000.
De vez en cuando, la tienda recibía de su competidor una hoja de cálculo con miles de precios de productos. Un equipo de tres empleados imprimía la hoja de cálculo en una gruesa pila de papel y se la repartía entre ellos. Para cada precio de producto, buscaban el precio de su tienda y anotaban todos los productos que sus competidores vendían por menos. Normalmente tardaban un par de días.

"Sabéis, podría escribir un programa para hacerlo si tenéis el archivo original de las impresiones", les dijo mi compañero de piso, cuando los vio sentados en el suelo con papeles esparcidos y apilados a su alrededor.

Al cabo de un par de horas, tenía un pequeño programa que leía el precio de un competidor de un archivo, encontraba el producto en la base de datos de la tienda y anotaba si el competidor era más barato. Aún no sabía programar y pasó la mayor parte del tiempo buscando documentación en un libro de programación. El programa sólo tardaba unos segundos en ejecutarse. Ese día, mi compañero de piso y sus compañeros se tomaron un almuerzo extra largo.

Este es el poder de la programación informática. Un ordenador es como una navaja suiza que puedes configurar para innumerables tareas. Muchas personas pasan horas haciendo clic y tecleando para realizar tareas repetitivas, sin saber que la máquina que utilizan podría hacer su trabajo en segundos si le dieran las instrucciones adecuadas.

¿A quién va dirigido este libro?

El software es la base de muchas de las herramientas que utilizamos hoy en día: Casi todo el mundo utiliza las redes sociales para comunicarse, muchas personas tienen ordenadores conectados a Internet en sus teléfonos y la mayoría de los trabajos de oficina implican interactuar con un ordenador para hacer el trabajo. Como consecuencia, la demanda de personas que sepan programar se ha disparado. Innumerables libros, tutoriales interactivos en Internet y campamentos de desarrolladores prometen convertir a principiantes ambiciosos en ingenieros de software con sueldos de seis cifras.

Este libro no es para esas personas. Es para todos los demás.

Por sí solo, este libro no le convertirá en un desarrollador de software profesional, igual que unas cuantas lecciones de guitarra no le convertirán en una estrella del rock. Pero si es usted oficinista, administrador, académico o cualquier otra persona que utilice un ordenador para trabajar o divertirse, aprenderá los fundamentos de la programación para poder automatizar tareas sencillas como las siguientes:

- Mover y renombrar miles de archivos y ordenarlos en carpetas
- Rellenar formularios en línea, sin necesidad de teclear
- Descargar archivos o copiar texto de un sitio web cada vez que se actualiza
- Que tu ordenador te envíe notificaciones personalizadas
- Actualización o formateo de hojas de cálculo Excel
- Comprobación del correo electrónico y envío de respuestas preescritas

Estas tareas son sencillas pero consumen mucho tiempo a los humanos, y a menudo son tan triviales o específicas que no hay software preparado para realizarlas. Con un poco de conocimientos de programación, puedes hacer que tu ordenador realice estas tareas por ti.

Convenciones

Este libro no está concebido como un manual de referencia, sino como una guía para principiantes. El estilo de codificación a veces va en contra de las mejores prácticas (por ejemplo, algunos programas utilizan variables globales), pero es una compensación para que el código sea más sencillo de aprender.
Este libro está hecho para que la gente escriba código desechable, por lo que no se dedica mucho tiempo al estilo y la elegancia. Los conceptos de programación sofisticados -como la programación orientada a objetos, la comprensión de listas, y los generadores- no se tratan debido a la complejidad que añaden. Los programadores veteranos pueden señalar formas en las que el código de este libro podría modificarse para mejorar la eficiencia, pero este libro se centra principalmente en conseguir que los programas funcionen con el mínimo esfuerzo.

¿Qué es la programación?

Los programas de televisión y las películas suelen mostrar a programadores tecleando furiosamente secuencias crípticas de 1s y 0s en pantallas brillantes, pero la programación moderna no es tan misteriosa. *Programar* no es más que introducir instrucciones para que el ordenador las ejecute. Estas instrucciones pueden servir para hacer números, modificar texto, buscar información en archivos o comunicarse con otros ordenadores a través de Internet.

Todos los programas utilizan instrucciones básicas como bloques de construcción. He aquí algunas de las más comunes, en inglés:

- **"Haz esto; luego haz aquello".**
- **"Si esta condición es verdadera, realiza esta ; de lo contrario, realiza esa acción".**
- **"Haz esta acción tal número de veces".**
- **"Sigue haciéndolo hasta que esta condición se cumpla".**

También puedes combinar estos bloques para tomar decisiones más complejas. Por ejemplo, aquí están las instrucciones de programación, llamadas *código fuente*, de un programa sencillo escrito en el lenguaje de programación Python. Empezando por la parte superior, el software Python ejecuta cada línea de código (algunas líneas se ejecutan sólo *si* se cumple una determinada condición o*, de lo contrario,* Python ejecuta alguna otra línea) hasta llegar a la parte inferior.

```
❶ passwordFile= open('SecretPasswordFile.txt')
❷ secretPassword= passwordFile.read()
➲  print('Introduzca su
   contraseña.') typedPassword=
   input()
❹ si contraseña_escrita== contraseña_secreta:
❺       print('Acceso concedido')
❻       si contraseña tecleada== '12345':
❼           print('Esa contraseña es la que pone un idiota en su equipaje.') else:
❽       print('Acceso denegado')
```

Puede que no sepas nada de programación, pero probablemente podrías hacer una suposición razonable de lo que hace el código anterior con sólo . Primero, se abre el archivo *SecretPasswordFile.txt* ❶, y se lee la contraseña secreta que contiene ❷. A continuación, se pide al usuario que introduzca una contraseña (desde el teclado)❸. Estas dos contraseñas se comparan ❹, y si son iguales, el programa imprime *Acceso concedido* en la pantalla ❺. A continuación, el programa comprueba si la contraseña es *12345* ❻ e insinúa que esta elección podría no ser la mejor para un

contraseña❼. Si las contraseñas no coinciden, el programa imprime *Acceso denegado* a la pantalla❽.

¿Qué es Python?

Python hace referencia al lenguaje de programación Python (con reglas sintácticas para escribir lo que se considera código Python válido) y al software intérprete de Python que lee el código fuente (escrito en el lenguaje Python) y ejecuta sus instrucciones. El intérprete de Python se puede descargar gratuitamente de http://python.org/, y existen versiones para Linux, OS X y Windows.

El nombre Python procede del surrealista grupo cómico británico Monty Python, no de la serpiente. A los programadores de Python se les llama cariñosamente pythonistas, y tanto las referencias a Monty Python como a la serpiente suelen salpicar los tutoriales y la documentación de Python.

Los programadores no necesitan saber muchas matemáticas

La ansiedad más común que oigo sobre aprender a programar es que la gente piensa requiere muchas matemáticas. En realidad, la mayor parte de la programación no requiere más matemáticas que la aritmética básica. De hecho, ser bueno programando no es muy distinto de ser bueno resolviendo sudokus.

Para resolver un rompecabezas Sudoku, hay que rellenar los números del 1 al 9 en cada fila, cada columna y cada cuadrado interior de 3×3 del tablero completo de 9×9. La solución se encuentra aplicando la deducción y la lógica a partir de los números iniciales. Por ejemplo, como el 5 aparece en la parte superior izquierda del Sudoku de la figura 1, no puede aparecer en ninguna otra parte de la fila superior, en la columna de más a la izquierda o en el cuadrado 3×3 de arriba a la izquierda. Resolver una fila, columna o cuadrado a la vez proporcionará más pistas numéricas para el resto del rompecabezas.

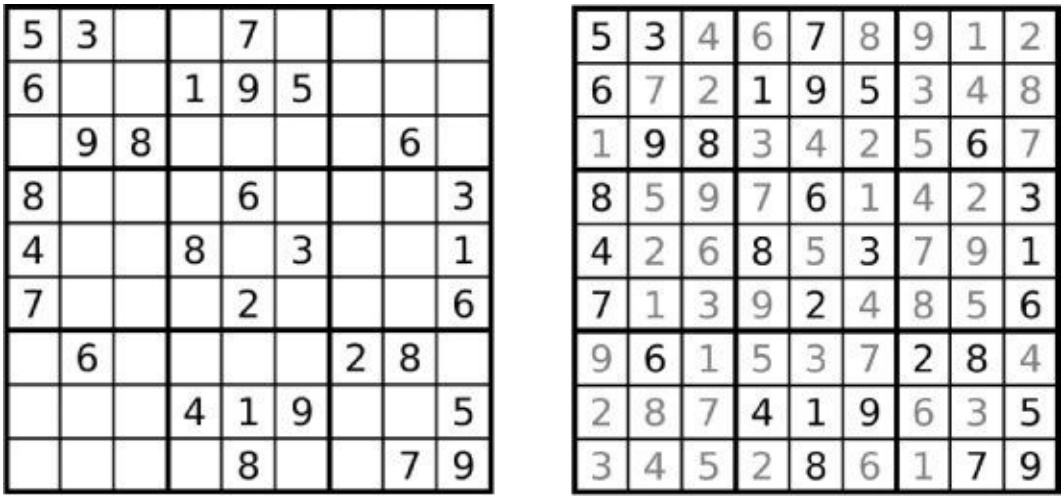


Figura 1. Un nuevo Sudoku (izquierda) y su solución (derecha). A pesar de usar números, el Sudoku no implica muchas matemáticas. (Imágenes © Wikimedia Commons)

El hecho de que el Sudoku incluya números no significa que haya que ser bueno en matemáticas para encontrar la solución. Lo mismo ocurre con la programación. Al igual que para resolver un Sudoku, para escribir programas hay que descomponer un problema en pasos individuales y detallados. Del mismo modo, al *depurar* programas (es decir, encontrar y corregir errores) tendrás que observar pacientemente lo que hace el programa y encontrar la causa de los fallos. Y como todas las habilidades, cuanto más programes, mejor te harás.

La programación es una actividad creativa

Programar es una tarea creativa, algo así como construir un castillo con ladrillos de LEGO. Empiezas con una idea básica de quieres que sea tu castillo y haces un inventario de los bloques disponibles. Luego empiezas a construir. Una vez que hayas terminado de construir tu programa, puedes embellecer tu código igual que harías con tu castillo.

La diferencia entre la programación y otras actividades creativas es que, al programar, tienes toda la materia prima que necesitas en tu ordenador; no necesitas comprar ningún lienzo, pintura, película, hilo, ladrillos LEGO o componentes electrónicos adicionales. Una vez escrito tu programa, puedes compartirlo fácilmente en línea con todo el mundo. Y aunque cometerás errores al programar, la actividad sigue siendo muy divertida.

Acerca de este libro

La primera parte de este libro cubre conceptos básicos de programación en Python, y la segunda parte cubre varias tareas que puedes hacer que tu ordenador automatice. Cada capítulo de la segunda parte contiene proyectos de programas para que los estudies. Aquí tienes un breve resumen de lo que encontrarás en cada capítulo:

Parte I

- **Capítulo 1**. Cubre las expresiones, el tipo más básico de instrucción en Python, y cómo utilizar el software de shell interactivo de Python para experimentar con el código.
- **Capítulo 2**. Explica cómo hacer que los programas decidan qué instrucciones ejecutar para que su código pueda responder inteligentemente a diferentes condiciones.
- **Capítulo 3**. Le enseña a definir sus propias funciones para que pueda organizar su código en trozos más manejables.
- **Capítulo 4**. Introduce el tipo de datos lista y explica cómo organizar los datos.
- **Capítulo 5**. Introduce el tipo de datos diccionario y muestra formas más potentes de organizar los datos.
- **Capítulo 6**. Cubre el trabajo con datos de texto (llamados *cadenas* en Python).

Parte II

- **Capítulo 7**. Cubre cómo Python puede manipular cadenas y buscar patrones de texto con expresiones regulares.
- **Capítulo 8**. Explica cómo tus programas pueden leer el contenido de archivos de texto y guardar información en archivos de tu disco duro.
- **Capítulo 9**. Muestra cómo Python puede copiar, mover, renombrar y borrar grandes cantidades de archivos mucho más rápido que un usuario humano. También explica cómo comprimir y descomprimir archivos.
- **Capítulo 10**. Muestra cómo utilizar las diversas herramientas de Python para encontrar y corregir errores.
- **Capítulo 11**. Muestra cómo escribir programas que pueden descargar automáticamente páginas web y analizarlas en busca de información. Esto se llama *web scraping*.
- **Capítulo 12**. Trata de la manipulación programática de hojas de cálculo de Excel para que no tengas que leerlas. Esto resulta útil cuando el número de documentos que tienes que analizar es de cientos o miles.
- **Capítulo 13**. Cubre la lectura programática de documentos Word y PDF.

- **Capítulo 14**. Continúa explicando cómo manipular mediante programación documentos con archivos CSV y JSON.
- **Capítulo 15**. Explica cómo los programas Python manejan la hora y las fechas y cómo programar tu ordenador para que realice tareas a determinadas horas. Este capítulo también muestra cómo tus programas Python pueden lanzar programas que no son Python.
- **Capítulo 16**. Explica cómo escribir programas que puedan enviar correos electrónicos y mensajes de texto en tu nombre.
- **Capítulo 17**. Explica cómo manipular mediante programación imágenes como archivos JPEG o PNG.
- **Capítulo 18**. Explica cómo controlar mediante programación el ratón y el teclado para automatizar los clics y las pulsaciones de teclas.

Descarga e instalación de Python

Puedes descargar Python para Windows, OS X y Ubuntu de forma gratuita desde http://python.org/downloads/. Si descargas la última versión desde la página de descargas del sitio web, todos los programas de este libro deberían funcionar.

ADVERTENCIA

Asegúrese de descargar una versión de Python 3 (como la 3.4.0). Los programas de este libro están escritos para ejecutarse en Python 3 y puede que no funcionen correctamente, si es que lo hacen, en Python 2.

Encontrarás instaladores de Python para ordenadores de 64 y 32 bits para cada sistema operativo en la página de descargas, así que primero averigua qué instalador necesitas. Si compraste tu ordenador en 2007 o después, lo más probable es que sea un sistema de 64 bits.

Si no es así, tienes una versión de 32 bits, pero aquí tienes cómo averiguarlo con seguridad:

- En Windows, seleccione **Inicio▸ Panel de control▸ Sistema** y compruebe si el Tipo de sistema dice 64-bit o 32-bit.
- En OS X, vaya al menú Apple, seleccione **Acerca de este Mac▸ Más información▸ Informe del sistema▸ Hardware** y, a continuación, fíjese en el campo Nombre del procesador. Si dice Intel Core Solo o Intel Core Duo, tienes una máquina de 32 bits. Si pone cualquier otra cosa (incluido Intel Core 2 Duo), tiene un equipo de 64 bits.
- En Ubuntu Linux, abra un Terminal y ejecute el comando **`uname -m`**. Una respuesta de `i686` significa 32 bits, y `x86_64` significa 64 bits.

En Windows, descarga el instalador de Python (el nombre del archivo terminará en *.msi*) y haz doble clic sobre él. Siga las instrucciones que el instalador muestra en pantalla para instalar Python, como se indica aquí:

1. Seleccione **Instalar para todos los usuarios** y haga clic en **Siguiente**.
2. Instale en la carpeta *C:\Python34* haciendo clic en **Siguiente**.
3. Vuelva a hacer clic en **Siguiente** para omitir la sección Personalizar Python.

En Mac OS X, descarga el archivo *.dmg* adecuado para tu versión de OS X y haz doble clic sobre él. Sigue las instrucciones que el instalador muestra en pantalla para instalar Python, como se indica aquí:

1. Cuando el paquete DMG se abra en una nueva ventana, haga doble clic en el icono *Python.*. Es posible que tenga que introducir la contraseña de administrador.
2. Haga clic en **Continuar** a través de la sección de bienvenida y haga clic en **Aceptar** para aceptar

la licencia.

3. Seleccione **HD Macintosh** (o el nombre que tenga su disco duro) y haga clic en **Instalar**.

Si estás ejecutando Ubuntu, puedes instalar Python desde la Terminal siguiendo estos pasos:

1. Abra la ventana Terminal.
2. Introduzca **sudo apt-get install python3**.
3. Introduce **sudo apt-get install idle3**.
4. Introduzca **sudo apt-get install python3-pip**.

Iniciar IDLE

Mientras que el *intérprete de Python* es el software que ejecuta tus programas Python, el software *del entorno de desarrollo interactivo (IDLE*) es donde introducirás tus programas, de forma parecida a un procesador de textos. Iniciemos IDLE ahora.

- En Windows 7 o posterior, haga clic en el icono Inicio situado en la esquina inferior izquierda de la pantalla, introduzca **IDLE** en el cuadro de búsqueda y seleccione **IDLE (Python GUI)**.
- En Windows , haga clic en el botón **Inicio** y seleccione **Programas▸ Python 3.4▸ IDLE (Python GUI)**.
- En Mac OS X, abra la ventana del Finder, haga clic en **Aplicaciones**, en **Python 3.4** y, a continuación, en el icono IDLE.
- En Ubuntu, seleccione **Aplicaciones▸ Accesorios▸ Terminal** y, a continuación, introduzca **idle3**. (También puede hacer clic en **Aplicaciones** en la parte superior de la pantalla, seleccionar **Programación** y, a continuación, hacer clic en **IDLE 3**.)

El caparazón interactivo

Independientemente del sistema operativo que estés ejecutando, la ventana de IDLE que aparece por primera vez debería estar casi en blanco, excepto por un texto parecido a éste:

```
Python 3.4.0 (v3.4.0:04f714765c13, 16 mar 2014, 19:25:23) [MSC v.1600 64
bit (AMD64)] en win32Escriba "copyright", "credits" o "license()" para obtener más
información.
>>>
```

Esta ventana se denomina *shell interactivo*. Un intérprete de comandos es un programa que te permite escribir instrucciones en el ordenador, como la Terminal o el Símbolo del sistema en OS X y Windows, respectivamente. La shell interactiva de Python te permite introducir instrucciones para que el intérprete de Python las ejecute. El ordenador lee las instrucciones que introduces y las ejecuta inmediatamente.

Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo junto a **>>>** rápido:

```
>>> print('¡Hola mundo!')
```

Después de escribir esa línea y pulsar ENTER, el intérprete de comandos interactivo debería mostrar esto como respuesta:

```
>>> print('¡Hola mundo!')
¡Hola, mundo!
```

Cómo encontrar ayuda

Resolver problemas de programación por tu cuenta es más fácil de lo que crees. Si no estás convencido, vamos a provocar un error a propósito: Introduce **'42'+ 3** en el intérprete de comandos interactivo. No necesitas saber qué significa esta instrucción en este momento, pero el resultado debería verse así:

```
>>> '42'+ 3
❶ Traceback (most recent call last):
    File "<pyshell#0>", line 1, in <module>
      '42'+ 3
❷ TypeError: No se puede convertir implícitamente el objeto 'int' en str
  >>>
```

El mensaje de error ❷ apareció aquí porque Python no pudo entender tu instrucción. La parte de rastreo ❶ del mensaje de error muestra la instrucción específica y el número de línea con el que Python tuvo problemas. Si no estás seguro qué hacer con un mensaje de error en particular, busca en línea el mensaje de error exacto. Escriba **"TypeError: Can't convert 'int' object to str implicitly"** (incluyendo las comillas) en tu motor de búsqueda favorito, y deberías ver montones de enlaces explicando qué significa el mensaje de error y qué lo causa, se muestra en
Figura 2.

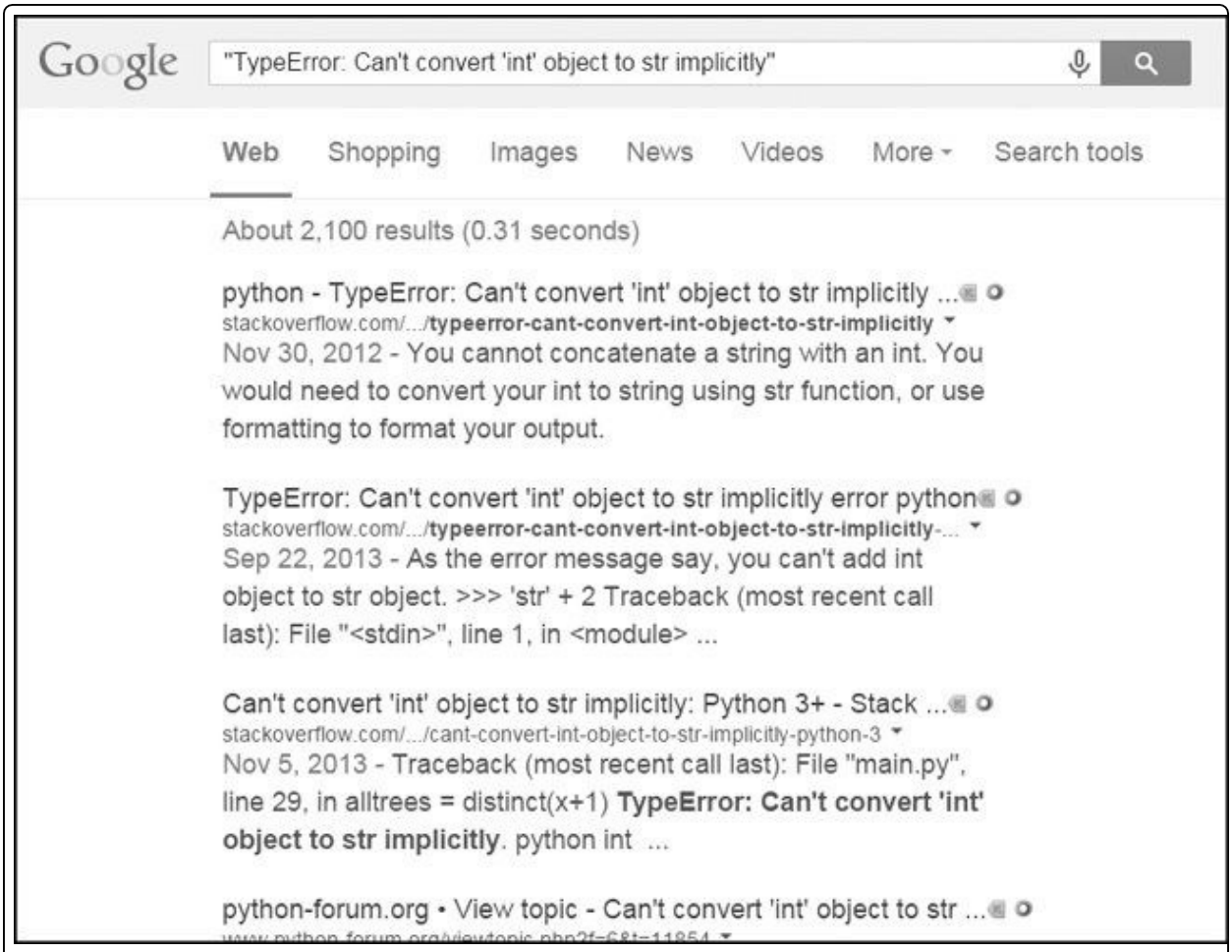


Figura 2. Los resultados de Google para un mensaje de error pueden ser muy útiles.

A menudo descubrirás que otra persona tenía la misma pregunta que tú y que alguna otra persona útil ya la ha respondido. Nadie puede saberlo todo sobre programación, así que una parte cotidiana del trabajo de cualquier desarrollador de software consiste en buscar respuestas a preguntas técnicas.

Preguntas inteligentes sobre programación

Si no encuentras la respuesta buscando en Internet, prueba a preguntar en un foro web como Stack Overlow *(*http://stackoverflow.com/) o en el subreddit "aprender programación" en http://reddit.com/r/learnprogramming/. Pero ten en cuenta que hay formas inteligentes de hacer preguntas sobre programación que ayudan a otros a ayudarte. Asegúrate de leer las secciones de preguntas frecuentes que tienen estos sitios web sobre la forma correcta de publicar preguntas.

Cuando hagas preguntas sobre programación, recuerda hacer lo siguiente:

- Explica lo que intentas hacer, no sólo lo que has hecho. Esto permite a tu ayudante saber si vas por mal camino.
- Especifique el momento en que se produce el error. Se produce nada más empezar el programa o sólo después de realizar una determinada acción?
- Copie y pegue *todo el* mensaje de error y su código en *http://pastebin.com/* o http://gist.github.com/.

  Estos sitios web permiten compartir fácilmente grandes cantidades de código con otras personas a través de la Red, sin riesgo de perder el formato del texto. Puedes poner la URL del código publicado en tu correo electrónico o en un mensaje del foro. Por ejemplo, aquí tienes algunos fragmentos de código que he publicado: *http://pastebin.com/SzP2DbFx/* y https://gist.github.com/asweigart/6912168/.
- Explica lo que ya has intentado hacer para resolver tu problema. Esto indica a los demás que ya te has esforzado por resolver las cosas por tu cuenta.
- Indique la versión de Python que está utilizando. (Hay algunas diferencias clave entre los intérpretes de Python de la versión 2 y los de la versión 3). También, di qué sistema operativo y versión estás ejecutando.
- Si el error aparece después de haber modificado el código, explique exactamente qué ha cambiado.
- Diga si es capaz de reproducir el error cada vez que ejecuta el programa o si sólo se produce después de realizar determinadas acciones. Explica cuáles son esas acciones, en caso afirmativo.

Sigue siempre las normas de etiqueta en Internet. Por ejemplo, no escribas tus preguntas en mayúsculas ni exijas demasiado a las personas que intentan ayudarte.

Resumen

Para la mayoría de la gente, el ordenador no es más que un aparato en lugar de una herramienta. Pero si aprendes a programar, tendrás acceso a una de las herramientas más potentes del mundo moderno y, de paso, te divertirás. Programar no es una operación de cerebro: está bien que los aficionados experimenten y cometan errores.

Me encanta ayudar a la gente a descubrir Python. Escribo tutoriales de programación en mi blog http://inventwithpython.com/blog/, y puedes ponerte en contacto conmigo si tienes preguntas en *.al@inventwithpython.com*

Con este libro partirá de cero en conocimientos de programación, pero es posible que tenga preguntas más allá de su alcance. Recuerda que hacer preguntas eficaces y saber cómo encontrar respuestas son herramientas inestimables en tu viaje por la programación.

¡Comencemos!

Parte I. Conceptos básicos de programación en Python

Capítulo 1. Conceptos básicos de Python Conceptos básicos de Python

El lenguaje de programación Python tiene una amplia gama de construcciones sintácticas, funciones de biblioteca estándar y características del entorno de desarrollo interactivo. Afortunadamente, puedes ignorar la mayor parte de eso; sólo necesitas aprender lo suficiente para escribir algunos pequeños programas prácticos.

Sin embargo, tendrás que aprender algunos conceptos básicos de programación antes de poder hacer nada. Al igual que un mago en prácticas, puede que estos conceptos te parezcan arcanos y tediosos, pero con un poco de conocimiento y práctica, serás capaz de manejar tu ordenador como una varita mágica para realizar hazañas increíbles.

Este capítulo tiene algunos ejemplos que te animan a escribir en la shell interactiva, que te permite ejecutar instrucciones Python de una en una y muestra los resultados instantáneamente. Usar la shell interactiva es genial para aprender lo que hacen las instrucciones básicas de Python, así que inténtalo mientras sigues. Recordarás mucho mejor las cosas que haces que las que sólo lees.

Introducción de expresiones en el intérprete de comandos interactivo

Para ejecutar el intérprete de comandos interactivo, ejecute IDLE, que instaló con Python en la introducción. En Windows, abra el menú Inicio, seleccione **Todos los programas▸ Python 3.3**, y luego seleccione **IDLE (Python GUI)**. En OS X, selecciona **Aplicaciones▸ MacPython 3.3▸ IDLE**. En Ubuntu, abre una nueva ventana de Terminal e introduce **idle3**.

Debería aparecer una ventana con el prompt`>>>` ; es el shell interactivo. Introduzca **`2+ 2`** para que Python realice algunas operaciones matemáticas sencillas.

```
>>> 2+ 2
4
```

La ventana IDLE debería mostrar ahora un texto como este:

```
Python 3.3.2 (v3.3.2:d047928ae3f6, 16 de mayo de 2013, 00:06:53) [MSC v.1600 64 bits
(AMD64)] en win32
Escriba "copyright", "credits" o "license()" para obtener más información.
>>> 2+ 2
4
>>>
```

En Python, `2+ 2` se denomina *expresión*, que es el tipo más básico de instrucción de programación del lenguaje. Las expresiones consisten en *valores* (como `2`) y *operadores* (como`+`), y siempre pueden *evaluarse* (es decir, reducirse) a un único valor. Esto significa que puedes usar expresiones en cualquier lugar del código Python en el que también podrías usar un valor.

En el ejemplo anterior, `2+ 2` se evalúa hasta un único valor, `4`. Un único valor sin operadores también se considera una expresión, aunque sólo se evalúa a sí mismo, como se muestra aquí:

```
>>> 2
2
```

¡LOS ERRORES ESTÁN BIEN!

Los programas se bloquearán si contienen código que el ordenador no puede entender, lo que hará que Python muestre un mensaje de error. Sin embargo, un mensaje de error no romperá tu ordenador, así que no tengas miedo de cometer errores. Un *fallo* sólo significa que el programa ha dejado de ejecutarse inesperadamente.

Si quieres saber más sobre un mensaje de error, puedes buscar el texto exacto del mensaje en línea para saber más sobre ese error específico. También puedes consultar los recursos en *http://nostarch.com/automatestuff/* para ver una lista de mensajes de error comunes de Python y sus significados.

Hay muchos otros operadores que también puedes usar en las expresiones de Python. Por ejemplo, la Tabla 1-1 lista todos los operadores matemáticos de Python.

Tabla 1-1. Operadores matemáticos de mayor a menor precedencia

Operador	Operación	Ejemplo	Evalúa a...
**	Exponente	2 ** 3	8
%	Módulo/remanente	22 % 8	6
//	División de enteros/cociente de enteros	22 // 8	2
/	División	22 / 8	2.75
*	Multiplicación	3 * 5	15
–	Resta	5 - 2	3
+	Adición	2+ 2	4

El orden de las operaciones (también llamado *precedencia*) de los operadores matemáticos de Python es similar al de las matemáticas. El operador ** se evalúa primero; los operadores *, /, // y % se evalúan a continuación, de izquierda a derecha; y los operadores+ y - se evalúan en último lugar (también de izquierda a derecha). Si es necesario, puede utilizar paréntesis para anular la precedencia habitual. Introduzca las siguientes expresiones en el intérprete de comandos interactivo:

```
>>> 2+ 3 * 6
20
>>> (2+ 3) * 6
30
>>> 48565878 * 578453
28093077826734
>>> 2 ** 8
256
>>> 23 / 7
3.2857142857142856
>>>  23  // 7
3
>>>  23  %7
2
>>>  2        +                2
4
>>> (5 - 1) * ((7+ 1) / (3 - 1))
```

```
16.0
```

En cada caso, tú como programador debes introducir la expresión, pero Python hace la parte difícil de evaluarla hasta obtener un único valor. Python seguirá evaluando partes de la expresión hasta que se convierta en un único valor, como se muestra en la Figura 1-1.

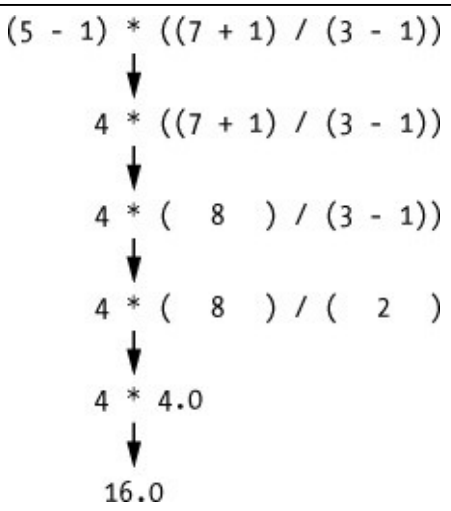


Figura 1-1. La evaluación de una expresión la reduce a un único valor.

Estas reglas para juntar operadores y valores para formar expresiones son una parte fundamental de Python como lenguaje de programación, igual que las reglas gramaticales que nos ayudan a comunicarnos. He aquí un ejemplo:

- **Se trata de una frase inglesa gramaticalmente correcta.**
- **Esto gramaticalmente es frase no inglés correcto a.**

La segunda línea es difícil de analizar porque no sigue las reglas del inglés. Del mismo modo, si escribes una instrucción Python incorrecta, Python no será capaz de entenderla y mostrará un mensaje de error SyntaxError, como se muestra aquí:

```
>>>+
  Archivo "<stdin>",
    línea 1 +
      ^
SyntaxError: sintaxis inválida
>>> 42+ 5+ * 2
  Archivo "<stdin>",
    línea 1 42+ 5+ * 2
              ^
SyntaxError: sintaxis inválida
```

Siempre puedes probar si una instrucción funciona escribiéndola en el intérprete de comandos interactivo. No te preocupes por romper el ordenador: Lo peor que puede pasar es que Python responda con un mensaje de error. Profesional

Los desarrolladores de software reciben mensajes de error mientras escriben código todo el tiempo.

Tipos de datos enteros, de coma flotante y de cadena

Recuerde que las expresiones son sólo valores combinados con operadores, y siempre se evalúan a un único valor. Un *tipo de datos* es una categoría para valores, y cada valor pertenece exactamente a un tipo de datos. Los tipos de datos más comunes en Python se listan en la Tabla 1-2. Los valores -2 y 30, por ejemplo, son valores *enteros*. El tipo de datos entero (o *int*) indica valores que son números enteros. Los números con un punto decimal, como 3.14, se llaman *números de punto flotante* (o *floats*). Observe que, aunque el valor 42 es un número entero, el valor
42,0 sería un número de coma flotante.

Tabla 1-2. Tipos de datos comunes

Tipo de datos	Ejemplos
Entero	-2, -1, 0, 1, 2, 3, 4, 5
Números en coma flotante	-1.25, -1.0, --0.5, 0.0, 0.5, 1.0, 1.25
Cuerdas	'a', 'aa', 'aaa', '¡Hola!', '11 gatos'

Los programas Python también pueden tener valores de texto llamados *cadenas*, o *strs* (pronunciado "stirs"). Rodea siempre tu cadena con comillas simples (') (como en 'Hola' o '¡Adiós mundo cruel!') para que Python sepa dónde empieza y termina la cadena. Incluso puedes tener una cadena sin caracteres, '', *cadena en blanco*. Las cadenas se explican con más detalle en el Capítulo 4.

Si alguna vez ve el mensaje de error SyntaxError: EOL al escanear literales de cadena, es probable que haya olvidado el carácter final de comillas simples al final de la cadena, como en este ejemplo:

```
>>> '¡Hola mundo!
SyntaxError: EOL al escanear literal de cadena
```

Concatenación y replicación de cadenas

El significado de un operador puede cambiar en función de los tipos de datos de los valores junto a él. Por ejemplo,+ es el operador de suma cuando opera sobre dos enteros o valores de coma flotante. Sin embargo, cuando+ se utiliza sobre dos valores de cadenaune las cadenas como operador *de concatenación de cadenas*. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> 'Alice'+ 'Bob'
AliceBob
```

La expresión se evalúa hasta un único valor de cadena nuevo que combina el texto de las dos cadenas. Sin embargo, si intenta utilizar el operador+ en una cadena y un valor entero, Python no sabrá cómo manejar esto, y mostrará un mensaje de error.

```
>>> 'Alice'+ 42
Traceback (llamada más reciente):
  File "<pyshell#26>", line 1, in <module>
    'Alice'+ 42
TypeError: No se puede convertir el objeto 'int' en str implícitamente
```

El mensaje de error Can't convert 'int' object to str implicitly significa que Python pensó que estabas intentando concatenar un entero a la cadena 'Alice'. Tu código tendrá que convertir explícitamente el entero en una cadena, porque Python no puede hacerlo automáticamente. (La conversión de tipos de datos se explicará en Diseccionando tu programa cuando hablemos de las funciones str(), int() y float()).

El operador * se utiliza para multiplicar cuando opera sobre dos valores enteros o de coma flotante. Pero cuando el operador * se utiliza sobre un valor de cadena y un valor entero, se convierte en el operador de *multiplicación de cadenas*. Introduzca una cadena multiplicada por un número en el intérprete de comandos interactivo para ver esto en acción.

```
>>> Alice * 5
'AliceAliceAliceAliceAlice'
```

La expresión se evalúa hasta un único valor de cadena que repite el original un número de veces igual al valor entero. La repetición de cadenas es un truco útil, pero no se utiliza tan a menudo como la concatenación de cadenas.

El operador * sólo puede utilizarse con dos valores numéricos (para la multiplicación) o un valor de cadena y un valor entero (para la replicación de cadenas). En caso contrario, Python

sólo mostrará un mensaje de error.

```
>>> Alice * Bob
Traceback (llamada más reciente):
  File "<pyshell#32>", line 1, in <module>
    'Alice' * 'Bob'
TypeError: can't multiply sequence by non-int of type 'str'
>>> 'Alice' * 5.0
Traceback (llamada más reciente):
  File "<pyshell#33>", line 1, in <module> 'Alice' *
    5.0
TypeError: can't multiply sequence by non-int of type 'float'
```

Es lógico que Python no entienda estas expresiones: No puedes multiplicar dos palabras, y es difícil replicar una cadena arbitraria un número fraccionario de veces.

Almacenamiento de valores en variables

Una *variable* es como una caja en la memoria del ordenador donde puedes guardar un único valor. Si desea utilizar el resultado de una expresión evaluada más adelante en su programa, puede guardarlo dentro de una variable.

Declaraciones de asignación

Los valores se almacenan en variables mediante una *sentencia de asignación*. Una sentencia de asignación consiste en un nombre de variable, un signo igual (llamado *operador de asignación*) y el valor a almacenar. Si introduce la sentencia de asignación `spam = 42`, entonces una variable llamada `spam` tendrá el valor entero `42` almacenado en ella.

Piense en una variable como una caja etiquetada en la que se coloca un valor, como en la

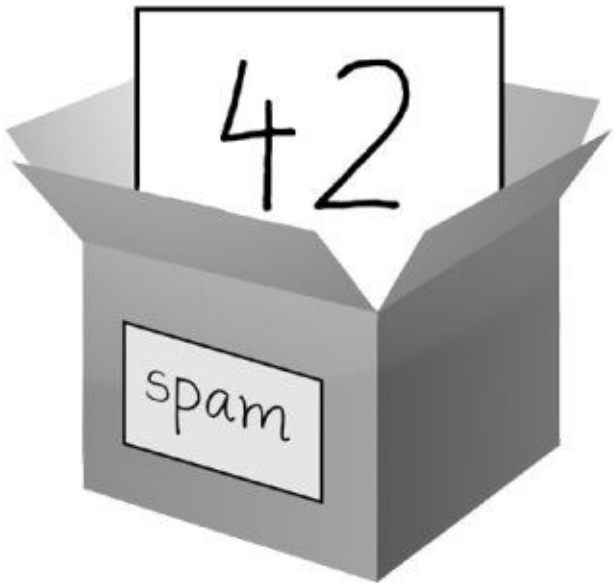


Figura 1-2. spam = 42 es como decirle al programa: "La variable spam tiene ahora el valor entero 42".

Figura 1-2.

Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> ❶spam= 40
    >>> spam
    40
    >>> HUEVOS= 2
>>> ❷spam+ HUEVOS
    42
    >>> spam+ HUEVOS+ spam
    82
➲   >>> spam= spam+ 2
    >>> spam
    42
```

Una variable se *inicializa* (o crea) la primera vez que se almacena un valor en ella ❶.

Después, se puede utilizar en expresiones con otras variables y valores ❷. Cuando a una variable se le asigna un nuevo valor ❸, el valor anterior se olvida, por eso

se evalúa a 42 en lugar de a 40 al final del ejemplo. Esto se llama *sobrescribir* la variable. Introduzca el siguiente código en el intérprete de comandos interactivo para intentar sobrescribir una cadena:

```
>>> spam= 'Hola'
>>> spam
Hola.
>>> spam= 'Adiós
>>> spam
Adiós
```

Al igual que el cuadro de la Figura 1-3, la variable spam de este ejemplo almacena "Hola". hasta que lo sustituyas por ""Adiós"".

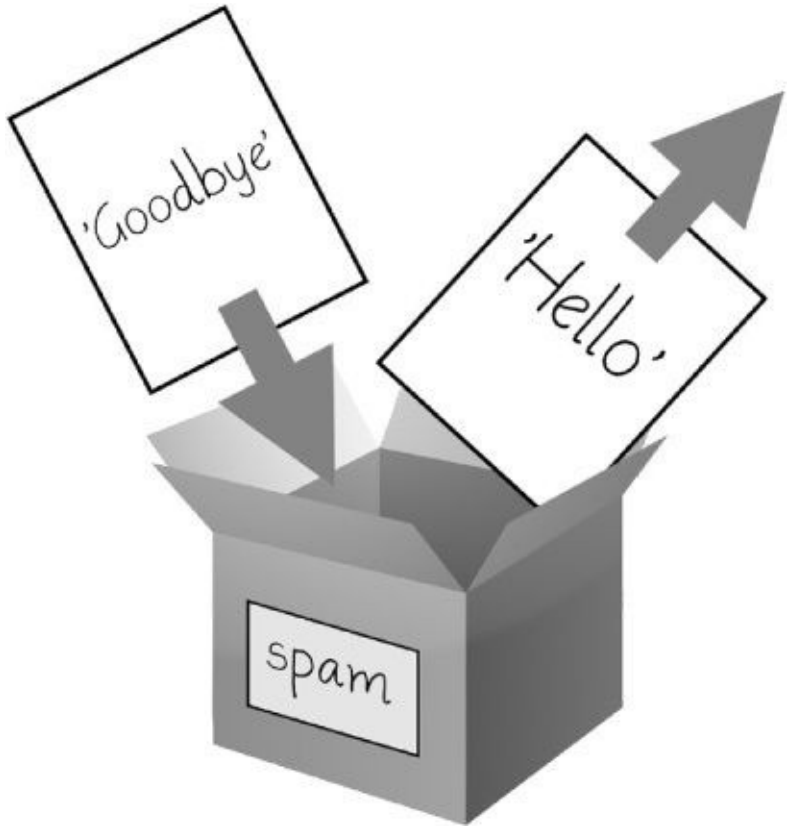


Figura 1-3. Cuando se asigna un nuevo valor a una variable, se olvida el anterior.

Nombres de las variables

La Tabla 1-3 contiene ejemplos de nombres legales de variables. Puedes nombrar una variable como quieras siempre que cumpla las tres reglas siguientes:

1. Sólo puede ser una palabra.
2. Sólo puede utilizar letras, números y el carácter de subrayado (_).
3. No puede empezar por un número.

Tabla 1-3. Nombres de variables válidos e inválidos

Nombres de variables válidos	Nombres de variables no válidos
saldo	current-balance (no se permiten guiones)
currentBalance	saldo actual (no se admiten espacios)
saldo_actual	4cuenta (no puede empezar por un número)
_spam	42 (no puede empezar con un número)
SPAM	total_$um (no se admiten caracteres especiales como$)
cuenta4	hola' (no se admiten caracteres especiales como ')

Los nombres de las variables distinguen entre mayúsculas y minúsculas, lo que significa que spam, SPAM, Spam y sPaM son cuatro variables diferentes. Es una convención de Python comenzar las variables con una letra minúscula.

Este libro utiliza camelcase para los nombres de variables en lugar de guiones bajos; es decir, las variables lookLikeThis en lugar de looking_like_this. Algunos programadores experimentados pueden señalar que el estilo de código oficial de Python, PEP 8, dice que deben usarse guiones bajos. Yo prefiero sin paliativos el camelcase y apunto a "A Foolish Consistency Is the Hobgoblin of Little Minds" en el propio PEP 8:

> *"La coherencia con la guía de estilo es importante. Pero lo más importante es saber cuándo hay que ser incoherente: a veces la guía de estilo no es aplicable. En caso de duda, usa tu mejor criterio".*

Un buen nombre de variable describe los datos que . Imagina que te mudas a una casa nueva y etiquetas todas las cajas de la mudanza como *Cosas*. Nunca encontraría nada. Los nombres de las variables spam, eggs y bacon se usan como nombres genéricos en los ejemplos de este libro y en gran parte de la documentación de Python (inspirados en el sketch de Monty Python "Spam"), pero en tus programas, un nombre descriptivo

ayudará a que su código sea más legible.

Su primer programa

Mientras que el intérprete de comandos interactivo es bueno para ejecutar instrucciones Python de una en una, para escribir programas Python completos, escribirás las instrucciones en el editor de archivos. El *editor de archivos* es similar a los editores de texto como Notepad o TextMate, pero tiene algunas características específicas para escribir código fuente. Para abrir el editor de archivos en IDLE, seleccione **Archivo▸ Nueva ventana**.

La ventana que aparece debe contener un cursor esperando tu entrada, pero es diferente de la shell interactiva, que ejecuta las instrucciones de Python en cuanto pulsas ENTER. El editor de archivos te permite escribir muchas instrucciones, guardar el archivo y ejecutar el programa. Así es como puedes diferenciar entre los dos:

- La ventana del shell interactivo será siempre la que tenga el prompt**>>>** .
- La ventana del editor de archivos no tendrá la indicación**>>>** .

Ahora es el momento de crear tu primer programa. Cuando se abra la ventana del editor de archivos, escribe lo siguiente en ella:

```
❶ # Este programa me saluda y me pregunta mi nombre.

❷ print('¡Hola mundo!')
   print("¿Cómo te llamas?")          # pregunta su nombre
➲  myName= input()
❹ print('Encantado de , '+ myName)
❺ print('La longitud de tu nombre es:')
   print(len(miNombre))
❻ print('¿Cuál es tu edad?')  # pregunta por su
   edad myAge= input()
   print('Tendrás '+ str(int(miEdad)+ 1)+ ' dentro de un año.')
```

Una vez introducido el código fuente, guárdalo para no tener que volver a cada vez que inicies IDLE. En el menú de la parte superior de la ventana del editor de archivos, selecciona **Archivo▸ Guardar como**. En la ventana Guardar como, introduce **hola.py** en el campo Nombre de archivo y haz clic en **Guardar**.

Deberías guardar tus programas de vez en cuando mientras los escribes. De esta forma, si el ordenador se bloquea o sales accidentalmente de IDLE, no perderás el código. Como atajo, puedes pulsar CTRL-S en Windows y Linux o⌘ -S en OS X para guardar tu archivo.

Una vez guardado, a ejecutar nuestro programa. Selecciona **Ejecutar▸ Ejecutar Módulo** o simplemente pulsa la tecla **F5**. Tu programa debería ejecutarse en la ventana del shell interactivo que apareció cuando arrancaste IDLE por primera vez. Recuerda, tienes que pulsar F5 desde la ventana

ventana del editor de archivos, no la ventana del intérprete de comandos interactivo. Introduzca su nombre cuando el programa se lo pida. La salida del programa en el intérprete de comandos interactivo debería tener este aspecto:

```
Python 3.3.2 (v3.3.2:d047928ae3f6, 16 de mayo de 2013, 00:06:53) [MSC v.1600 64 bits
(AMD64)] en win32
Escriba "copyright", "credits" o "license()" para obtener más información.
>>>========== REINICIA==========
>>>
¡Hola, mundo!
¿Cómo te llamas?
Al
Es un placer , Al La longitud
de su nombre es:
2
¿Cuál es su edad?
4
Tendrás 5 años dentro de un año.
>>>
```

Cuando no hay más líneas de código que ejecutar, el programa Python *termina*; es decir, deja de ejecutarse. (También se puede decir que el programa Python *sale*).

Puede cerrar el editor de archivos pulsando la X situada en la parte superior de la ventana. Para volver a cargar un programa guardado, seleccione **Archivo► Abrir** en el menú. Hazlo ahora, y en la ventana que aparece, elige ***hola.py***, y pulsa el botón **Abrir**. Tu programa *.py* previamente guardado debería abrirse en la ventana del editor de archivos.

Diseccionando su programa

Con tu nuevo programa abierto en el editor de archivos, vamos a hacer un rápido recorrido por las instrucciones Python que utiliza viendo lo que hace cada línea de código.

Comentarios

La línea siguiente se denomina *comentario*.

```
❶ # Este programa me saluda y me pregunta mi nombre.
```

Python ignora los comentarios, y puedes usarlos para escribir notas o recordarte a ti mismo lo que el código está intentando hacer. Cualquier texto del resto de la línea que siga a un signo de almohadilla (**#**) forma parte de un comentario.

A veces, los programadores ponen un **#** delante de una línea de código para eliminarla temporalmente mientras prueban un programa. Esto se llama *comentar el* código, y puede ser útil cuando estás tratando de averiguar por qué un programa no funciona. Puedes quitar el **#** más tarde, cuando estés listo para volver a poner la línea.

Python también ignora la línea en blanco después del comentario. Puedes añadir tantas líneas en blanco a tu programa como quieras. Esto puede hacer que tu código sea más fácil de leer, como los párrafos de un libro.

La función print()

La función `print()` muestra en pantalla el valor de la cadena dentro del paréntesis.

```
❷ print('¡Hola mundo!')
  print("¿Cuál es tu nombre?") # pregunta por su nombre
```

La línea `print('¡Hola mundo!')` significa "Imprime el texto de la cadena `'¡Hola mundo!'`". Cuando Python ejecuta esta , dice que Python está *llamando a* la función `print()` y el valor de la cadena está siendo *pasado* a la función. Un valor que se pasa a una llamada de función es un *argumento*. Observa que las comillas no se imprimen en la pantalla. Sólo marcan dónde empieza y termina la cadena; no son parte del de la cadena.

NOTA

También puede utilizar esta función para poner una línea en blanco en la pantalla; sólo tiene que llamar a `print(`*sin nada entre los paréntesis.*

Al escribir el nombre de una función, los paréntesis de apertura y cierre al final la identifican como el nombre de una función. Esta es la razón por la que en este libro verás `print()` en lugar de `print`. El capítulo 2 describe las funciones con más detalle.

La función input()

La función input() espera a que el usuario escriba algún texto en el teclado y pulse ENTER.

```
➲ myName= input()
```

Esta llamada a la función evalúa a una cadena igual al texto del usuario, y la línea de código anterior asigna la variable myName a este valor de cadena.

Puede pensar en la llamada a la función input() como una expresión que se evalúa a cualquier cadena que el usuario haya introducido. Si el usuario introdujo "Al" la expresión se evaluaría como myName= "Al".

Imprimir el nombre del usuario

La siguiente llamada a `print()` contiene en realidad la expresión `'Me alegro de conocerte, '+ myName` entre los paréntesis.

```
❹ print('Encantado de , '+ myName)
```

Recuerde que las expresiones siempre pueden evaluarse a un único valor. Si `'Al'` es el valor almacenado en `myName` en la línea anterior, entonces esta expresión se evalúa a `'It is good to meet you, Al'`. Este único valor de cadena se pasa a `print()`, que lo imprime en la pantalla.

La función len()

Puede pasar a la función `len()` un valor de cadena (o una variable que contenga una cadena), y la función evaluará el valor entero del número de caracteres de esa cadena.

```
❺ print('La longitud de tu nombre es:')
  print(len(miNombre))
```

Introduzca lo siguiente en el intérprete de comandos interactivo para :

```
>>> len('hola')
5
>>> len('Mi monstruo muy enérgico acaba de devorar NACHOS.')
46
>>> len('')
0
```

Al igual que en los ejemplos anteriores, `len(miNombre)` se evalúa como un entero. Luego se pasa a `print()` para que lo muestre en la pantalla. Observe que `print()` le permite pasar valores enteros o valores de cadena. Pero fíjate en el error que cuando escribes lo siguiente en el intérprete de comandos interactivo:

```
>>> print('I+ 29+ ' AÑOS.')
Traceback (llamada más reciente):
  File "<pyshell#6>", line 1, in <module>
    print('Tengo '+ 29+ ' años.')
TypeError: No se puede convertir el objeto 'int' en str implícitamente
```

La función `print()` no está causando ese error, sino que es la expresión que intentaste pasar a `print()`. Obtendrá el mismo mensaje de error si escribe la expresión en el intérprete de comandos interactivo por sí sola.

```
>>> 'I+ 29+ ' AÑOS.'
Traceback (llamada más reciente):
  File "<pyshell#7>", line 1, in <module>
    'Tengo '+ 29+ ' años.'
TypeError: No se puede convertir el objeto 'int' en str implícitamente
```

Python da un error porque sólo puedes usar el operador`+` para sumar dos enteros o concatenar dos cadenas. No puedes sumar un entero a una cadena porque no es gramatical en Python. Puedes solucionarlo utilizando una versión de cadena del entero, como se explica en la siguiente sección.

Funciones str(), int() y float()

Si desea concatenar un entero como 29 con una cadena para pasarla a `print()`, necesitará obtener el valor `'29'`, que es la forma de cadena de 29. A la función `str()` se le puede pasar un valor entero y se evaluará a una versión de valor de cadena del mismo, como sigue:

```
>>> str(29)
'29'
>>> print('Tengo '+ str(29)+ ' años.')
Tengo 29 .
```

Dado que `str(29)` se evalúa como `'29'`, la expresión `'Tengo '+ str(29)+ ' años.'` se evalúa como `'Tengo '+ '29'+ ' años.'`, que a su vez se evalúa como `'Tengo 29 años.'`. Este es el valor que se pasa a la función `print()`.

Las funciones `str()`, `int()` y `float()` se evaluarán a las formas de cadena, entero y punto flotante del valor que pases, respectivamente. Pruebe a convertir algunos valores en el shell interactivo con estas funciones y observe lo que ocurre.

```
>>> str(0)
'0'
>>> str(-3.14)
'-3.14'
>>> int('42')
42
>>> int('-99')
-99
>>> int(1,25)
1
>>> int(1,99)
1
>>> float('3.14')
3.14
>>> float(10)
10.0
```

Los ejemplos anteriores llaman a las funciones `str()`, `int()` y `float()` y les pasan valores de los otros tipos de datos para obtener una forma de cadena, entero o punto flotante de esos valores.

La función `str()` resulta útil cuando se tiene un número entero o flotante que se quiere

concatenar a una cadena. La función **int()** también es útil si tiene un número como valor de cadena que desea utilizar en algunas matemáticas. Por ejemplo, la **input()** siempre devuelve una cadena, aunque el usuario introduzca un número. Introduzca **spam= input()** en el intérprete de comandos interactivo e introduzca **101** cuando espere su texto.

```
>>> spam= input()
101
>>> spam
'101'
```

El valor almacenado dentro de spam no es el entero 101 sino la cadena **'101'**. Si desea realizar operaciones matemáticas con el valor de spam, utilice la **int()** para obtener la forma entera de spam y, a continuación, guárdela como nuevo valor en spam.

```
>>> spam= int(spam)
>>> spam
101
```

Ahora deberías poder tratar la variable spam como un entero en lugar de como una cadena.

```
>>> spam * 10 / 5
202.0
```

Tenga en cuenta que si pasa un valor a **int()** que no puede evaluar como un entero, Python mostrará un mensaje de error.

```
>>> int('99.99')
Traceback (llamada más reciente):
  File "<pyshell#18>", line 1, in <module>
    int('99.99')
ValueError: literal inválido para int() con base 10: '99.99'
>>> int('doce')
Traceback (llamada más reciente):
  File "<pyshell#19>", line 1, in <module> int('doce')
ValueError: literal inválido para int() con base 10: 'doce'
```

La función **int()** también es útil si necesita redondear a la baja un número de coma flotante.

```
>>> int(7,7)
7
>>> int(7.7)+ 1
8
```

En tu programa, has utilizado las funciones `int()` y `str()` en las tres últimas líneas para obtener un valor del tipo de datos apropiado para el código.

```
❻ print('¿Cuál es tu edad?') # pregunta por su
   edad myAge= input()
   print('Tendrás '+ str(int(miEdad)+ 1)+ ' dentro de un año.')
```

La variable myAge contiene el valor devuelto por `input()`. Dado que la función `input()` siempre devuelve una cadena (aunque el usuario haya introducido un número), puede utilizar el código `int(miEdad)` para devolver un valor entero de la cadena en `miEdad`. Este valor entero se suma entonces a `1` en la expresión `int(miEdad)+ 1`.

El resultado de esta suma se pasa a la función `str()`: `str(int(miEdad+ 1)`. El valor de cadena devuelto se concatena con las cadenas `'Tendrás '` y `' dentro de un .'` para evaluarlo en un valor de cadena grande. Esta cadena grande se pasa finalmente a `print()` para que se muestre en la pantalla.

Digamos que el usuario introduce la cadena `'4'` para `miEdad`. La cadena `"4"` se convierte en un número entero por lo que puede sumarle uno. El resultado es `5`. La función `str()` convierte el resultado de nuevo en una cadena, para que puedas concatenarla con la segunda cadena, `'en un año.'`, para crear el mensaje final. Estos pasos de evaluación se parecerían a la Figura 1-4.

EQUIVALENCIA TEXTUAL Y NUMÉRICA

Aunque el valor de cadena de un número se considera un valor completamente distinto de la versión entera o de coma flotante, un número entero puede ser igual a uno de coma flotante.

```
>>> 42== '42'
Falso
>>> 42== 42.0
Verdadero
>>> 42,0== 0042.000
Verdadero
```

Python hace esta distinción porque las cadenas son texto, mientras que los enteros y los flotantes son números.

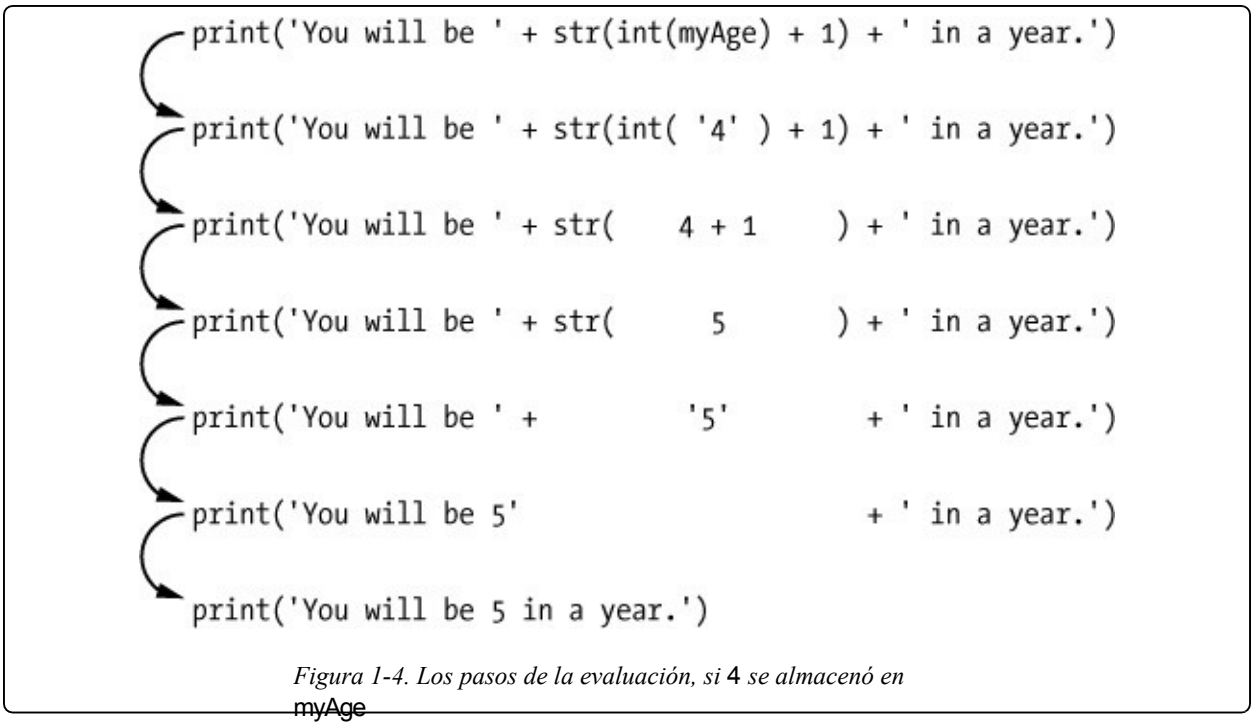


Figura 1-4. Los pasos de la evaluación, si 4 *se almacenó en* myAge

Resumen

Puedes calcular expresiones con una calculadora o escribir concatenaciones de cadenas con un procesador de textos. Incluso puedes replicar cadenas fácilmente copiando y pegando texto. Pero las expresiones y los valores que las componen -operadores, variables y llamadas a funciones- son los elementos básicos que conforman los programas. Una vez que sepas cómo manejar estos elementos, podrás instruir a Python para que opere con grandes cantidades de datos por ti.

Es bueno recordar los diferentes tipos de operadores (`+`, `-`, `*`, `/`, `//`, `%` y `**` para operaciones matemáticas, y`+` y `*` para operaciones con cadenas) y los tres tipos de datos (enteros, números en coma flotante y cadenas) introducidos en este capítulo.

También se han introducido algunas funciones diferentes. Las funciones `print()` y `input()` gestionan la salida de texto simple (a la pantalla) y la entrada (desde el teclado). La función `len()` toma una cadena y evalúa a un int el número de caracteres de la cadena. Las funciones `str()`, `int()` y `float()` evalúan la cadena, el número entero o el número en coma flotante del valor que se les pasa.

En el siguiente capítulo, aprenderás cómo decirle a Python que tome decisiones inteligentes sobre qué código ejecutar, qué código omitir y qué código repetir basándose en los valores que tiene. Esto se conoce como *control de flujo*, y te permite escribir programas que toman decisiones inteligentes.

Preguntas prácticas

P: 1. ¿Cuáles de los siguientes son operadores y cuáles son valores?

```
* 'hola'
-88.8
-
/
+
5
```

P: 2. ¿Cuál de las siguientes es una variable y cuál es una cadena?

```
spam "correo
basura"
```

P: 3. Nombra tres tipos de datos.

P: 4. ¿De qué se compone una expresión? ¿Qué hacen todas las expresiones?

P: 5. Este capítulo introdujo las declaraciones de asignación, como el spam= 10. ¿Cuál es la diferencia entre una expresión y una sentencia?

P: 6. ¿Qué contiene la variable bacon después de que se ejecute el siguiente código?

```
bacon= 20
bacon+ 1
```

P: 7. ¿A cuánto deberían evaluarse las dos expresiones siguientes?

```
"spam" + 'spamspam"   "spam"  * 3
```

P: 8. ¿Por qué huevos es un nombre de variable válido mientras que 100 no lo es?

P: 9. ¿Qué tres funciones pueden utilizarse para obtener la versión entera, en coma flotante o en cadena de un valor?

P: 10. ¿Por qué da error esta expresión? ¿Cómo puedes ?

```
"He comido "+ 99+  " burritos."
```

Crédito extra: Busca en Internet la documentación de Python para la función len(). Estará en una página web titulada "Funciones incorporadas". Hojea la lista de otras funciones que tiene Python, busca lo que hace la función round() y experimenta con ella en el intérprete de comandos interactivo.

Capítulo 2. Control del flujo

Así que ya sabes lo básico de las instrucciones individuales y que un programa no es más que una serie de instrucciones. Pero la verdadera fuerza de la programación no es simplemente *ejecutar* una instrucción tras otra como una lista de recados de fin de semana. En función de cómo se evalúen las expresiones, el programa puede decidir saltarse instrucciones, repetirlas o elegir una de entre varias instrucciones para ejecutar. De hecho, casi nunca querrás que tus programas comiencen desde la primera línea de código y simplemente ejecuten cada línea, directamente hasta el final. *Las sentencias de control de flujo* pueden decidir qué instrucciones Python ejecutar bajo qué condiciones.

Estas sentencias de control de flujo corresponden directamente a los símbolos de un diagrama de flujo, por lo que proporcionaré versiones de diagramas de flujo del código discutido en este capítulo. La Figura 2- 1 muestra un diagrama de flujo sobre qué hacer si está lloviendo. Sigue el camino hecho por las flechas desde el Inicio hasta el Fin.

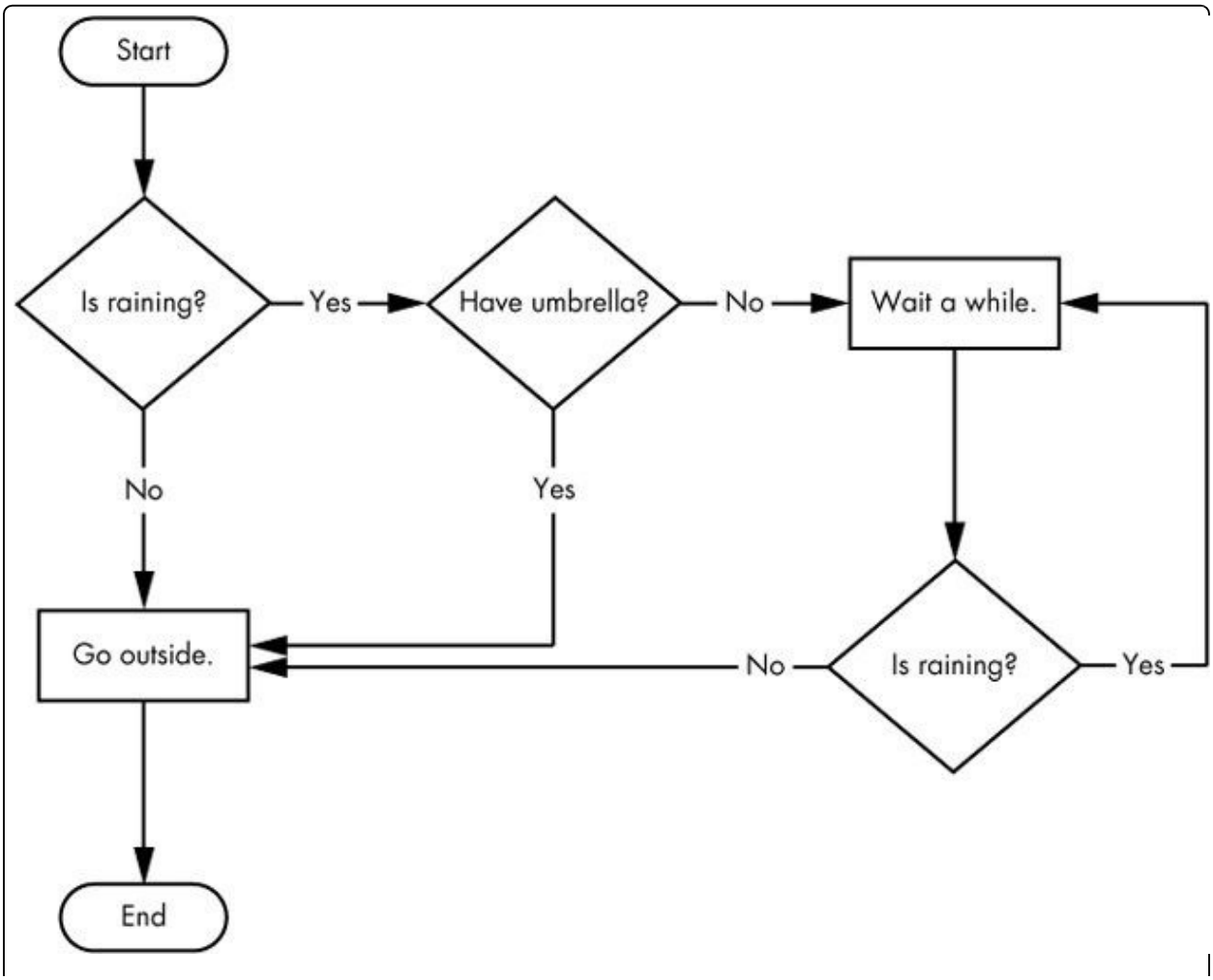


Figura 2-1. Diagrama de flujo para saber qué hacer si llueve

En un diagrama de flujo, suele haber más de una forma de ir del principio al final. Lo mismo ocurre con las líneas de código de un programa informático. Los diagramas de flujo representan estos puntos de ramificación con rombos, mientras que los demás pasos se representan con rectángulos. Los pasos inicial y final se representan con rectángulos redondeados.

Pero antes de aprender sobre las sentencias de control de flujo, primero necesitas aprender representar esas opciones de *sí* y *no*, y necesitas entender cómo escribir esos puntos de ramificación como código Python. Para ello, vamos a explorar los valores booleanos, los operadores de comparación y los operadores booleanos.

Valores booleanos

Mientras que los tipos de datos entero, coma flotante y cadena tienen un número ilimitado de valores posibles, el tipo de datos *booleano* sólo tiene dos valores: Verdadero y Falso. (Boolean se escribe con mayúsculas porque el tipo de datos debe su nombre al matemático George Boole). Cuando se escriben como código Python, los valores booleanos True y False carecen de las comillas que se colocan alrededor de las cadenas, y siempre comienzan con una *T* o *F* mayúscula, con el resto de la palabra en minúsculas. Introduzca lo siguiente en el intérprete de comandos interactivo. (Algunas de estas instrucciones son intencionadamente incorrectas, y provocarán la aparición de mensajes de error).

```
>>> ❶spam= Verdadero
    >>> spam
    Verdadero
>>> ❷true
    Traceback (llamada más reciente):
      File "<pyshell#2>", line 1, in <module> true
    NameError: el nombre 'true' no está definido
➲   >>> Verdadero= 2+ 2
    SyntaxError: asignación a palabra clave
```

Como cualquier otro valor, los valores booleanos se usan en expresiones y se pueden almacenar en variables ❶. Si no utilizas las mayúsculas y minúsculas adecuadas ❷ o intentas utilizar Verdadero y Falso para los nombres de las variables Python te dará un mensaje de error.

Operadores de comparación

Los operadores de comparación comparan dos valores y los evalúan en un único valor booleano. En la Tabla 2-1 se enumeran los operadores de comparación.

Tabla 2-1. Operadores de comparación

Operador	Significado
==	Igual a
!=	No es igual a
<	Menos de
>	Mayor que
<=	Inferior o igual a
>=	Mayor o igual que

Estos operadores se evalúan como Verdadero o Falso dependiendo de los valores que les des. Probemos ahora algunos operadores, empezando por== y !=.

```
>>> 42== 42
Verdadero
>>> 42== 99
Falso
>>> 2 != 3
Verdadero
>>> 2 != 2
Falso
```

Como es de esperar,== (igual a) se evalúa como Verdadero cuando los valores de ambos lados son iguales, y != (no igual a) se evalúa como Verdadero cuando los dos valores son diferentes. En realidad, los operadores== y != pueden funcionar con valores de cualquier tipo de datos.

```
>>> 'hola'== 'hola'
    Verdadero
    >>> 'hola'== 'Hola'
    Falso
    >>> perro' != 'gato'
    Verdadero
```

```
>>> Verdadero== Verdadero
Verdadero
>>> Verdadero != Falso
Verdadero
>>> 42== 42.0
Verdadero
>>> ❶42== '42'
Falso
```

Tenga en cuenta que un valor entero o de punto flotante siempre será desigual a un valor de cadena. La expresión `42== '42'` ❶ se evalúa como `False` porque Python considera que el entero `42` es diferente de la cadena `'42'`.

En cambio, los operadores , ,`<>`<= y>= sólo funcionan correctamente con valores enteros y de coma flotante.

```
>>> 42< 100
Verdadero
>>> 42> 100
Falso
>>> 42< 42
Falso
>>> eggCount= 42
>>> ❶eggCount<= 42
Verdadero
>>> miEdad= 29
>>> ❷myAge>= 10
Verdadero
```

LA DIFERENCIA ENTRE LOS OPERADORES== Y =

Puede que te hayas dado cuenta de que el operador**==** (igual a) tiene dos signos iguales, mientras que el operador**=** (asignación) sólo tiene un signo igual. Es fácil confundir estos dos operadores. Sólo tienes que recordar estos puntos:

El operador**==** (igual a) pregunta si dos valores son iguales entre . El operador**=** (asignación)

- coloca el valor de la derecha en la variable de la izquierda.

Para recordar cuál es cuál, observe que el operador**==** (igual a) consta de dos caracteres, al igual que el operador **!=** (no igual a) consta de dos caracteres.

A menudo utilizarás operadores de comparación para comparar el valor de una variable con otro valor, como en los ejemplos de `eggCount<= 42` ❶ y `myAge>= 10` ❷.

(Después de todo, en lugar de escribir **`'perro' != 'gato'`** en tu código, podrías haber escrito simplemente **`True`**). Verás más ejemplos de esto más adelante cuando aprendas sobre las sentencias de control de flujo.

Operadores booleanos

Los tres operadores booleanos (and, or, and not) se utilizan para comparar valores booleanos. Al igual que los operadores de comparación, evalúan estas expresiones hasta un valor booleano. Exploremos estos operadores en detalle, empezando por el operador and.

Operadores booleanos binarios

Los operadores and y or siempre toman dos valores booleanos (o expresiones), por lo que se consideran operadores *binarios*. El operador and evalúa una expresión a True si *ambos* valores booleanos son True; de lo contrario, se evalúa a False. Introduzca algunas expresiones utilizando y en el intérprete de comandos interactivo para verlo en acción.

```
>>> Cierto y verdadero
Verdadero
>>> Verdadero y falso
Falso
```

Una *tabla verdadero-falso* muestra todos los resultados posibles de un operador booleano. La Tabla 2-2 es la tabla verdadero-falso para el operador and.

Tabla 2-2. Tabla verdadero-falso del operador and

Expresión	Evalúa a...
Cierto y verdadero	Verdadero
Verdadero y falso	Falso
Falso y verdadero	Falso
Falso y falso	Falso

Por otro lado, el operador or evalúa una expresión a True si *cualquiera de* los dos valores booleanos es True. Si ambos son falsos, evalúa como falso.

```
>>> Falso o verdadero
Verdadero
>>> Falso o falso
Falso
```

Puede ver todos los resultados posibles del operador or en su tabla verdadero-falso, que se muestra en la Tabla 2-3.

Tabla 2-3. Tabla verdadero-falso del operador or

Expresión	Evalúa a...
Verdadero o Verdadero	Verdadero
Verdadero o falso	Verdadero

Falso o verdadero	Verdadero
Falso o falso	Falso

El operador no

A diferencia de and y or, el operador not opera sobre un único valor booleano (o expresión). El operador not simplemente evalúa el valor booleano opuesto.

```
>>> no Verdadero
    Falso
>>> ❶no no no Verdadero
    Verdadero
```

Al igual que con los dobles negativos en el habla y la escritura, puede anidar los operadores not ❶, aunque nunca hay ninguna razón para hacer esto en programas reales. La Tabla 2-4 muestra la tabla verdadero-falso de not.

Tabla 2-4. Tabla verdadero-falso del operador not

Expresión	Evalúa a...
no Verdadero	Falso
no Falso	Verdadero

Combinación de operadores booleanos y de comparación

Dado que los operadores de comparación se evalúan a valores booleanos, puede utilizarlos en expresiones con operadores booleanos.

Recuerde que los operadores y, o, y no se llaman operadores booleanos porque siempre operan sobre los valores booleanos Verdadero y Falso. Aunque expresiones como 4< 5 no son valores booleanos, son expresiones que se evalúan a valores booleanos. Intente introducir algunas expresiones booleanas que utilicen operadores de comparación en el intérprete de comandos interactivo.

```
>>> (4< 5) y (5< 6)
Verdadero
>>> (4< 5) y (9< 6)
Falso
>>> (1== 2) o (2== 2)
Verdadero
```

El ordenador evaluará primero la expresión de la izquierda y después la de la derecha. Cuando conozca el valor booleano de cada una, evaluará toda la expresión hasta llegar a un valor booleano. Puedes pensar en el proceso de evaluación ordenador para (4< 5) y (5< 6) como se muestra en la Figura 2-del
2.

También puede utilizar varios operadores booleanos en una expresión, junto con los operadores de comparación.

```
>>> 2+ 2== 4 y no 2+ 2== 5 y 2 * 2== 2+ 2
Verdadero
```

Los operadores booleanos tienen un orden de operaciones igual los operadores matemáticos. Después de evaluar cualquier operador matemático o de comparación, Python evalúa primero los operadores not, luego los operadores and y por último los operadores or.

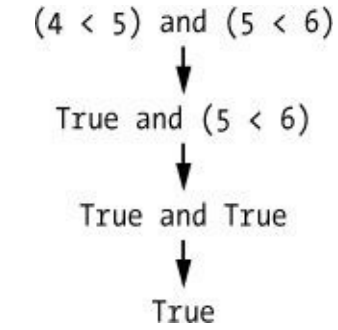


Figura 2-2. El proceso de evaluación de (4< 5) *y* (5< 6) *a* Verdadero.

Elementos de control de caudal

Las sentencias de control de flujo suelen empezar con una parte llamada *condición*, y todas van seguidas de un bloque de código llamado *cláusula*. Antes de que aprendas sobre las sentencias de control de flujo específicas de Pythoncubriré lo que son una condición y un bloque.

Condiciones

Las expresiones booleanas que ha visto hasta ahora podrían considerarse condiciones, que son lo mismo que expresiones; *condición* es sólo un nombre más específico en el contexto de las sentencias de control de flujo. Las condiciones siempre se evalúan a un valor booleano, True o False. Una sentencia de control de flujo decide qué hacer basándose en si su condición es True o False, y casi todas las sentencias de control de flujo utilizan una condición.

Bloques de código

Las líneas de código Python pueden agruparse en *bloques*. Puedes saber cuándo empieza y termina un bloque por la sangría de las líneas de código. Hay tres reglas para los bloques.

1. Los bloques comienzan cuando aumenta la sangría.
2. Los bloques pueden contener otros bloques.
3. Los bloques terminan cuando la sangría se reduce a cero o a la sangría de un bloque contenedor.

Los bloques son más fáciles de entender mirando un poco de código indentado, así que vamos a encontrar los bloques en parte de un pequeño programa de juego, que se muestra aquí:

```
si nombre== "María":
❶       print('Hola María")
        si contraseña== 'pez espada':
❷           print('Acceso
        concedido.') else:
➲           print("Contraseña incorrecta.')
```

El primer bloque de código ❶ comienza en la línea `print('Hola María")` y contiene todas las líneas que le siguen. Dentro de este bloque hay otro bloque ❷, que sólo una línea: `print('Acceso concedido.')`. El tercer bloque ❸ también tiene una sola línea: `print("Contraseña incorrecta.')`.

Ejecución del programa

En el programa *hello.py* del capítulo anterior, Python comenzó a ejecutar instrucciones en la parte superior del programa yendo hacia abajo, una tras otra. La *ejecución del programa* (o simplemente, *ejecución*) es un término para la instrucción actual que se está ejecutando. Si imprimes el código fuente en papel y pones tu dedo en cada línea mientras se ejecuta, puedes pensar en tu dedo como la ejecución del programa.

Sin embargo, no todos los programas se ejecutan simplemente yendo hacia abajo. Si utiliza el dedo para recorrer un programa con sentencias de control de flujo, es probable que se encuentre saltando por el código fuente en función de las condiciones y que se salte cláusulas enteras.

Declaraciones de control de flujo

Ahora, exploremos la pieza más importante del control de flujo: las sentencias mismas. Las sentencias representan los diamantes que viste en el diagrama de flujo de la Figura 2-1, y son las decisiones reales que tomarán tus programas.

Declaraciones if

El tipo más común de sentencia de control de flujo es la sentencia **if**. La cláusula de una sentencia if (es decir, el bloque que sigue a la sentencia if) se ejecutará si la condición de la sentencia es True. La cláusula se omite si la condición es falsa.

En inglés sencillo, una sentencia **if** podría leerse como: "Si esta condición es verdadera, ejecuta el código de la cláusula". En Python, una sentencia **if** consiste en lo siguiente:

- La palabra clave **if**
- Una condición (es decir, una expresión que se evalúa como True o False)
- Un colon
- A partir de la línea siguiente, un bloque de código sangrado (llamado cláusula **if**)

Por ejemplo, supongamos que tienes un código que comprueba si alguien se llama Alicia. (Imagina que al nombre se le asignó algún valor anteriormente).

```
if name== 'Alice':
    print('Hola, Alice.')
```

Todas las sentencias de control de flujo terminan con dos puntos y van seguidas de un nuevo bloque de código (la cláusula). La cláusula de esta sentencia **if** es el bloque con **print('Hola, Alice.')**. La Figura 2-3 muestra cómo un diagrama de flujo de este código.

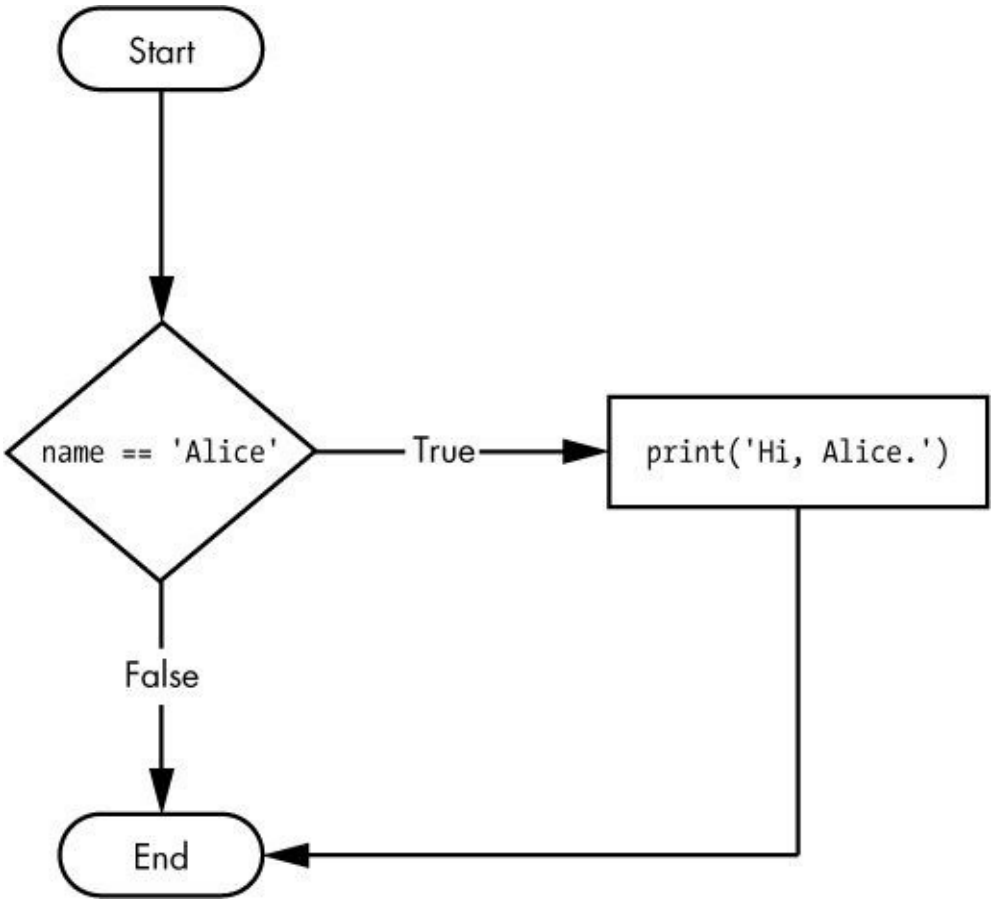


Figura 2-3. Diagrama de flujo de una sentencia if

else Declaraciones

Una cláusula `if` puede ir seguida opcionalmente de una sentencia `else`. La `else` se ejecuta sólo cuando la condición de la sentencia `if` es `False`. En inglés sencillo, una sentencia `else` podría leerse como, "Si esta condición es verdadera, ejecute este . O si no, ejecute ese ". Una sentencia `else` no tiene condición, y en código, una `else` siempre consiste en lo siguiente:

- La palabra clave `else`
- Un colon
- A partir de la siguiente línea, un bloque de código sangrado (llamado cláusula

`else`) Volviendo al ejemplo de Alice, veamos algo de código que utiliza una cláusula

`else`
declaración para ofrecer un saludo diferente si el nombre de la persona no es Alice.

```
if name== 'Alice':
    print('Hola, Alice.')
si no:
    print('Hola, forastero.')
```

La Figura 2-4 muestra cómo un diagrama de flujo de este código.

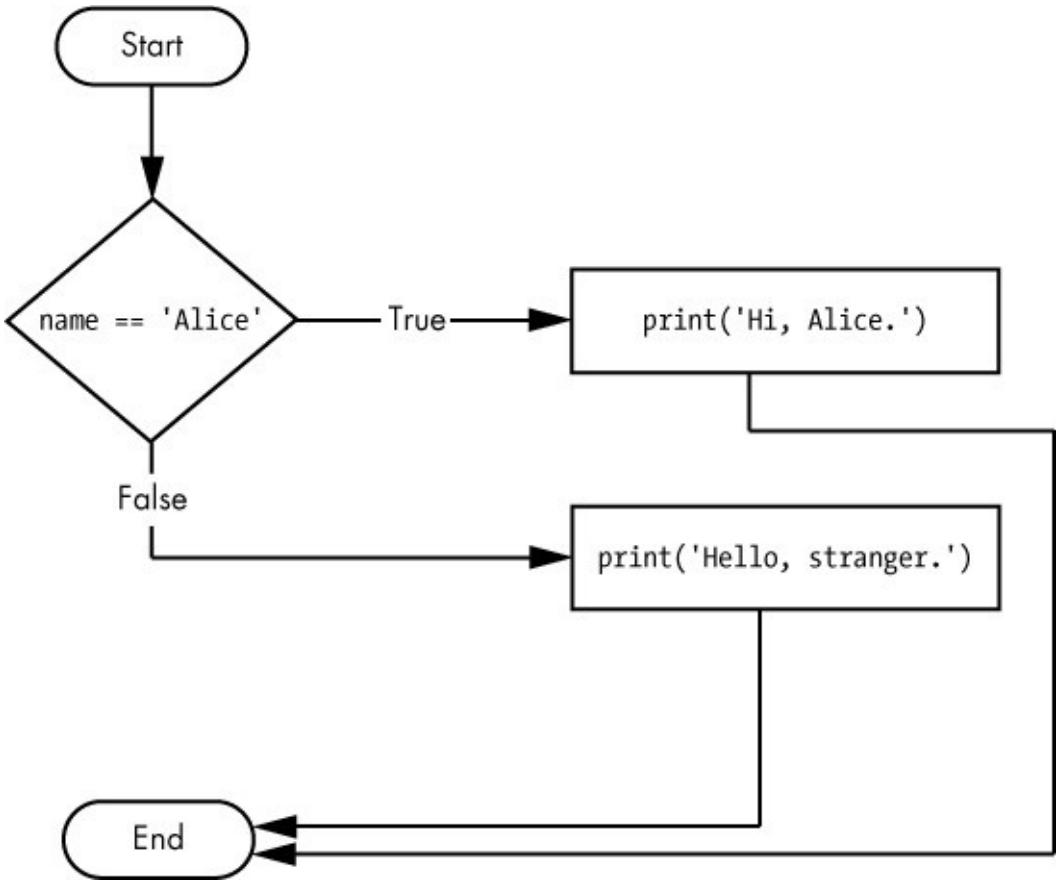


Figura 2-4. Diagrama de flujo de una sentencia else

elif Declaraciones

Aunque sólo se ejecutará una de las cláusulas **if** o **else**, puede darse el caso de que desee que se ejecute una de las *muchas* cláusulas posibles. La sentencia **elif** es una sentencia "else if" que siempre sigue a una sentencia **if** o a otra **elif**. Proporciona otra condición que se comprueba sólo si alguna de las condiciones anteriores es False. En código, una sentencia **elif** siempre consiste en lo siguiente:

- La palabra clave **elif**
- Una condición (es decir, una expresión que se evalúa como True o False)
- Un colon
- Comenzando en la siguiente línea, un bloque de código sangrado (llamado cláusula **elif**) Añadamos una **elif** al comprobador de nombres para ver esta declaración en acción.

```
if name== 'Alice':
    print('Hola, Alice.')
elif edad< 12:
    print('Tú no eres Alicia, niña')
```

Esta vez, compruebe la edad de la persona, y el programa le dirá algo diferente si es menor de 12 años. Puede ver el diagrama de flujo para esto en
Figura 2-5.

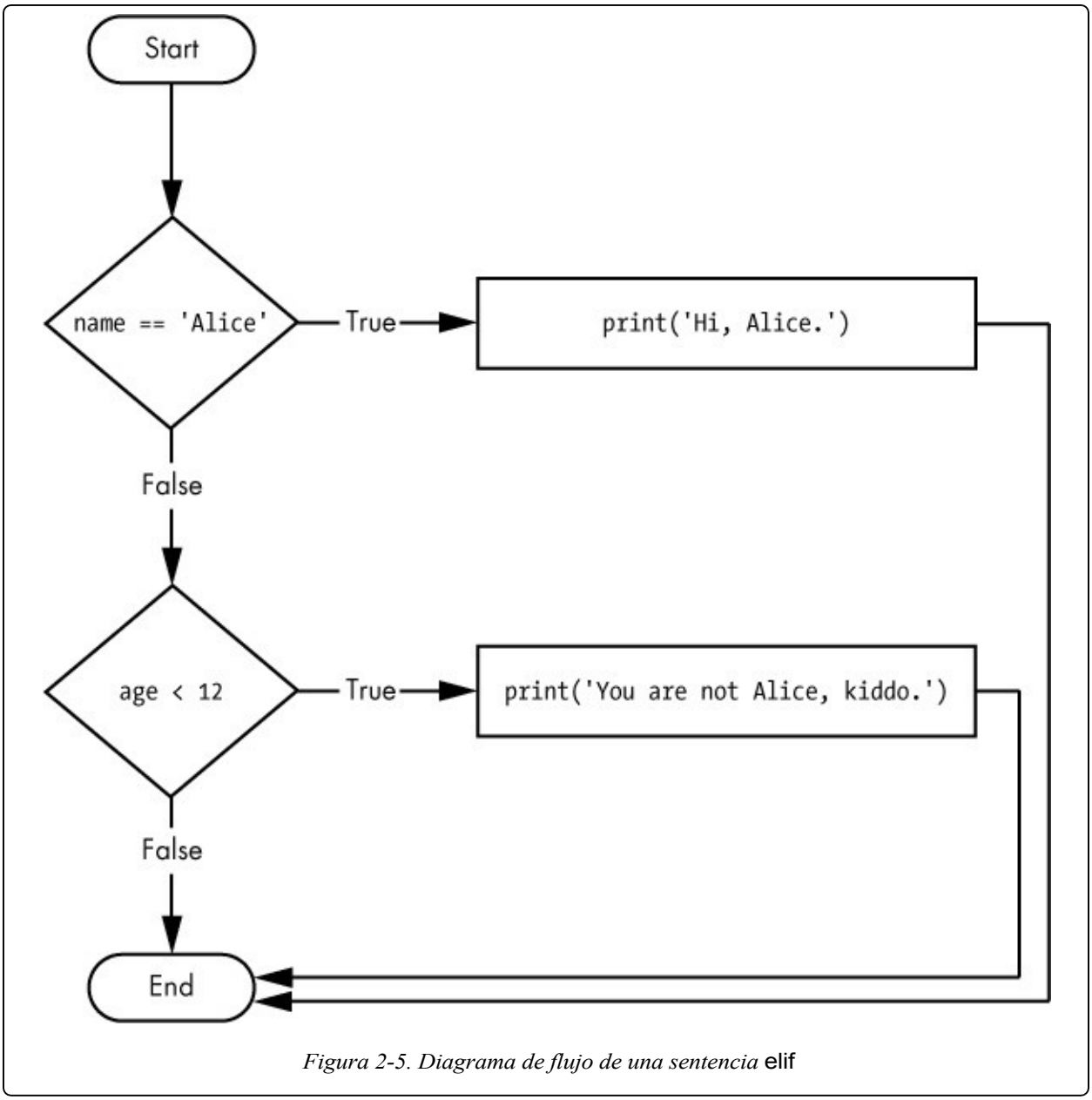


Figura 2-5. Diagrama de flujo de una sentencia elif

La cláusula `elif` se ejecuta si la edad< 12 es Verdadera y el nombre== 'Alice' es Falso. Sin embargo, si ambas condiciones son falsas, se omiten ambas cláusulas. *No* se garantiza que se ejecute al menos una de las cláusulas.

Cuando hay una cadena de sentencias `elif`, sólo se ejecutará una o ninguna de las cláusulas. Una vez que una de las condiciones de las sentencias es True, el resto de las cláusulas `elif` se saltan automáticamente. Por ejemplo, abra una nueva ventana del editor de archivos e introduzca el siguiente código, guardándolo como *vampiro.py*:

```
if name== 'Alice':
    print('Hola, Alice.')
elif edad< 12:
```

```
        print('No eres Alicia, chiquilla.') elif
    age> 2000:
        print('A diferencia de ti, Alice no es una vampiresa inmortal") elif age>
    100:
        print('No eres Alice, abuelita.')
```

Aquí he añadido dos sentencias `elif` más para hacer que el comprobador de nombres salude a una persona con diferentes respuestas basadas en la edad. La Figura 2-6 muestra el diagrama de flujo para esto.



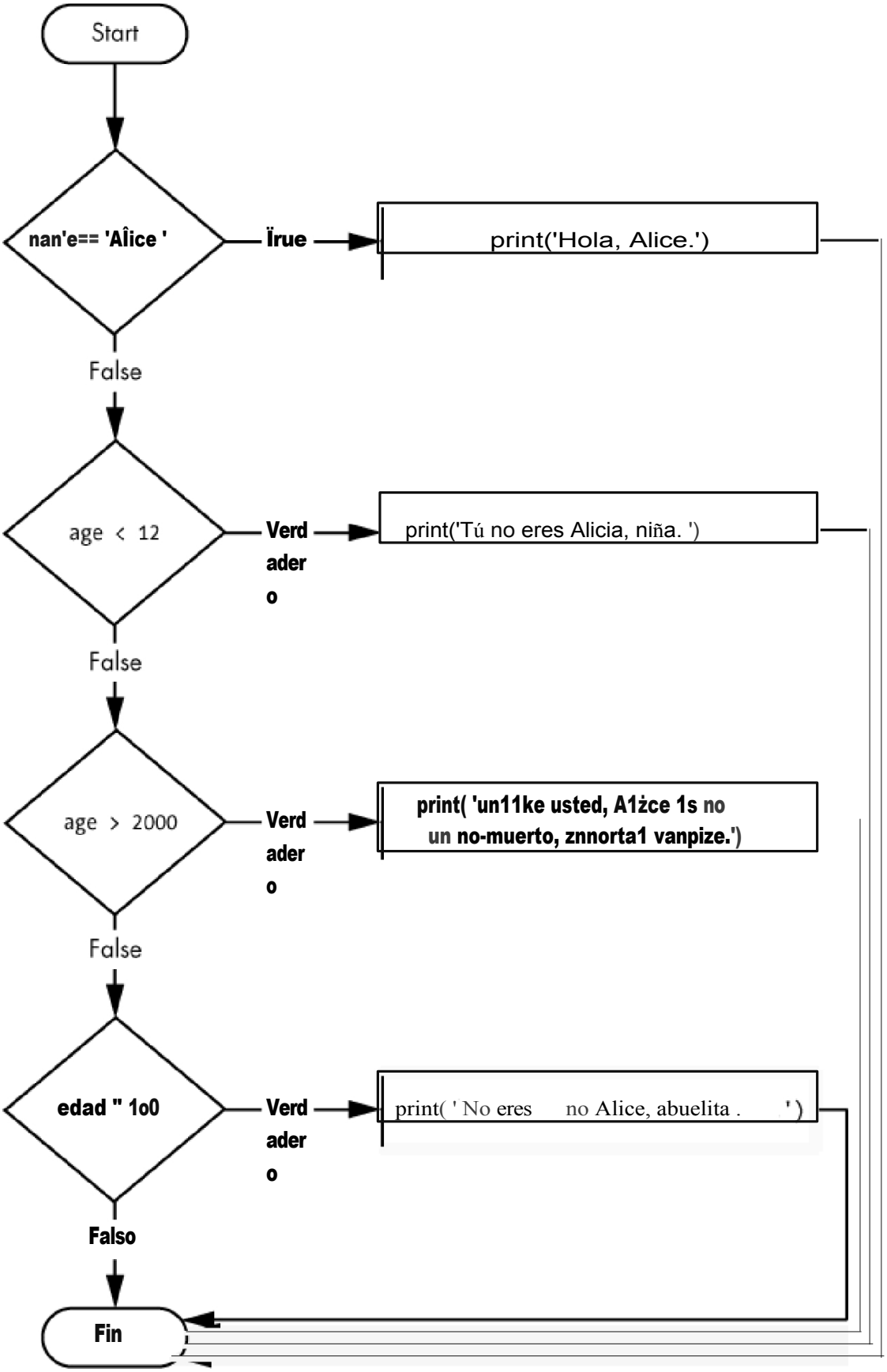

Start
nan'e== 'AÎice '
Ïrue
print('Hola, Alice.')
False
age < 12
Verd
ader
o
print('Tú no eres Alicia, niña. ')
False
age > 2000
Verd
ader
o
print('un11ke usted, A1żce 1s no
un no-muerto, znnorta1 vanpize.')
False
edad " 1o0
Verd
ader
o
print(' No eres no Alice, abuelita . ')
Falso
Fin

Figura 2-6. Diagrama de flujo para múltiples sentencias elif *en el programa vampire.py*

Sin embargo, el orden de las sentencias **elif** sí importa. Vamos a reordenarlas para introducir un error. Recuerde que el resto de las cláusulas **elif** se omiten automáticamente una vez que se ha encontrado una condición True, por lo que si cambia algunas de cláusulas en *vampire.py*, se encontrará con un problema. Cambie el código para que se parezca al siguiente, y guárdelo como *vampire2.py*:

```
si nombre== 'Alice':
        print('Hola, Alice.')
    elif age< 12:
        print('Tú no eres Alicia, niña')
❶ elif edad> 100:
        print('No eres Alicia, abuelita.') elif
    age> 2000:
        print('A diferencia de ti, Alice no es un vampiro no muerto e inmortal.')
```

Digamos que la variable edad contiene el valor 3000 antes de que se ejecute este código. Podrías esperar que el código imprimiera la cadena 'A diferencia de ti, Alicia no es un vampiro no muerto e inmortal'. Sin embargo, debido a que la condición age> 100 es True (después de todo, 3000 *es* mayor que 100)❶, la cadena 'Tú no eres Alice, grannie.', y el resto de las sentencias **elif** se saltan automáticamente. Recuerde que, como máximo, sólo se ejecutará una de las cláusulas y que, en el caso de las sentencias **elif**, el orden es importante.

La Figura 2-7 muestra el diagrama de flujo del código anterior. Observe cómo los diamantes para la edad> 100 y la edad> 2000 están intercambiados.

Opcionalmente, puede tener una sentencia else después de la última sentencia elif. En ese caso, se garantiza que al menos una (y sólo una) de las cláusulas se ejecutará. Si las condiciones de cada sentencia **if** y **elif** son False, entonces se la cláusula else. Por ejemplo, recreemos el programa Alice para usar las cláusulas **if**, **elif** y else.

```
if name== 'Alice':
    print('Hola, Alice.')
elif edad< 12:
    print('Tú no eres Alicia, niña') else:
    print('No eres Alicia ni un niño pequeño')
```

La figura 2-8 muestra el diagrama de flujo de este nuevo código, que guardaremos como

littleKid.py.

En lenguaje llano, este tipo de estructura de control de flujo sería: "Si la primera condición es cierta, haz esto. Si la segunda condición es verdadera, haz esto. Si no, haz otra cosa". Cuando utilice estas tres sentencias juntas, recuerde estas reglas sobre cómo ordenarlas para evitar errores como el de la Figura 2-7. Primero, siempre hay exactamente una sentencia `if`. Cualquier sentencia `elif` que necesite debe seguir a la sentencia `if`. En segundo lugar, si quiere estar seguro de que se ejecuta al menos una cláusula, cierre la estructura con `una else`.

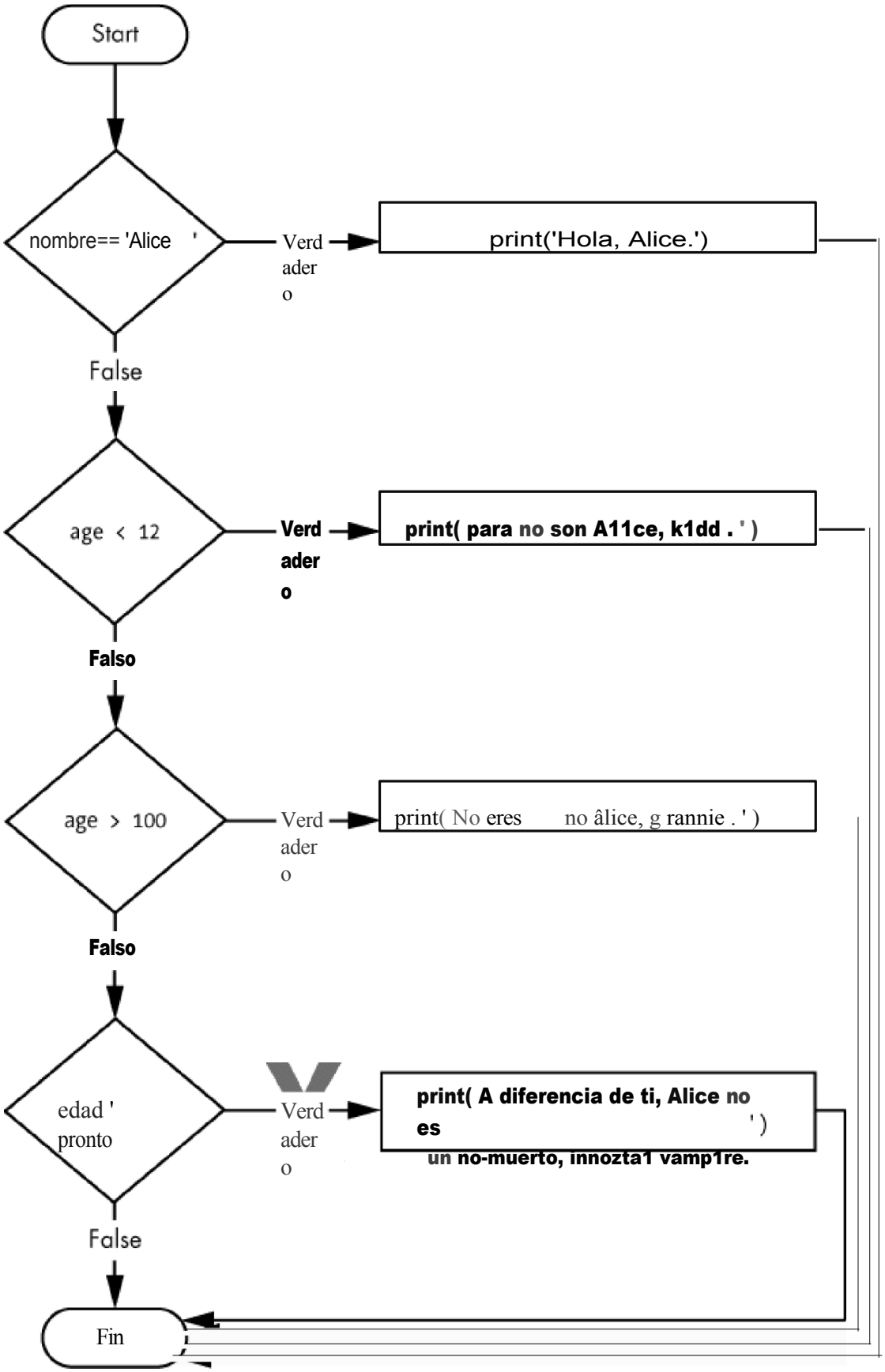

Start
nombre== 'Alice '
Verd
ader
o
print('Hola, Alice.')
False
age < 12
Verd
ader
o
print(para no son A11ce, k1dd . ')
Falso
age > 100
Verd
ader
o
print(No eres no âlice, g rannie . ')
Falso
edad '
pronto
Verd
ader
o
print(A diferencia de ti, Alice no
es
un no-muerto, innozta1 vamp1re.
')
False
Fin

Figura 2-7. Diagrama de flujo del programa vampire2.py. El camino tachado lógicamente nunca sucederá, porque si la edad *fuera mayor que* 2000*, ya habría sido mayor que* 100*.*

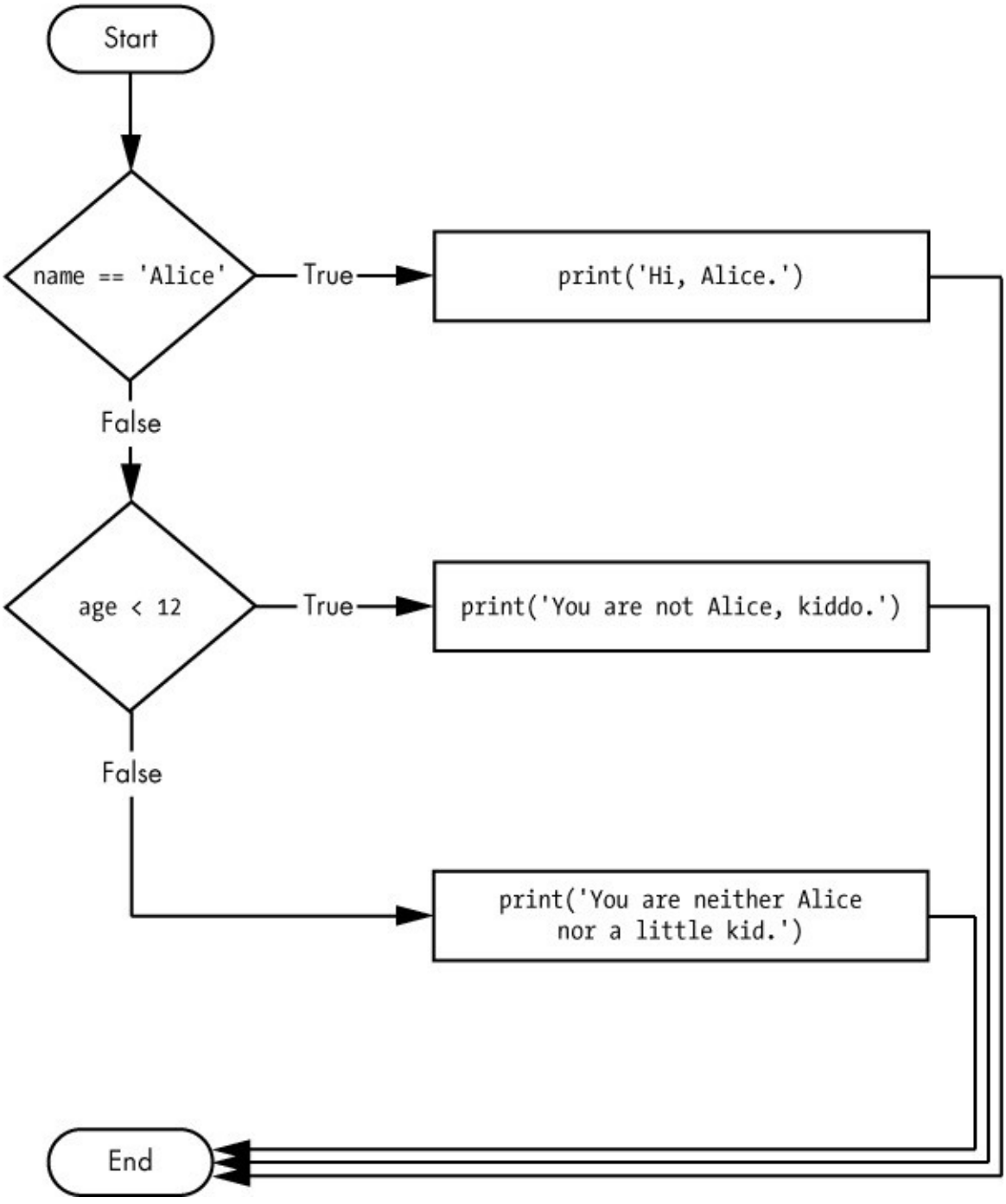


Figura 2-8. Diagrama de flujo del programa anterior littleKid.py

Declaraciones de bucle while

Puede hacer que un bloque de código se ejecute una y otra vez con una sentencia while. El código de una cláusula while se ejecutará mientras la condición de la sentencia while sea True. En código, una sentencia while siempre consiste en lo siguiente:

- La palabra clave while
- Una condición (es decir, una expresión que se evalúa como True o False)
- Un colon
- A partir de la línea siguiente, un bloque de código sangrado (llamado cláusula while)

Puede ver que una sentencia while se parece a una sentencia if. La diferencia está en cómo se comportan. Al final de una cláusula if, la ejecución del programa continúa después de la if. Pero al final de una cláusula while, ejecución del programa vuelve al principio de la sentencia while. La cláusula while a menudo se denomina *bucle while* o simplemente *bucle*.

Veamos una sentencia if y un bucle while que usan la misma condición y toman las mismas acciones basadas en esa condición. Aquí está el código con una sentencia if:

```
spam= 0
si spam< 5:
    print('Hola, mundo.')
    spam= spam+ 1
```

Aquí está el código con una sentencia while:

```
spam= 0
while spam< 5:
    print('Hola, mundo.')
    spam= spam+ 1
```

Estas sentencias son similares: tanto if como while comprueban el valor de spam y, si es inferior a cinco, imprimen un mensaje. Pero cuando ejecutas estos dos fragmentos de código, ocurre algo muy diferente en cada uno de ellos. Para la sentencia if, la salida es simplemente "Hola, mundo". Pero para la sentencia while, es "Hola, mundo" repetido cinco veces. Echa un vistazo a los diagramas de flujo de estos dos trozos de , Figura 2-9 y Figura 2-10, para ver por qué ocurre esto.

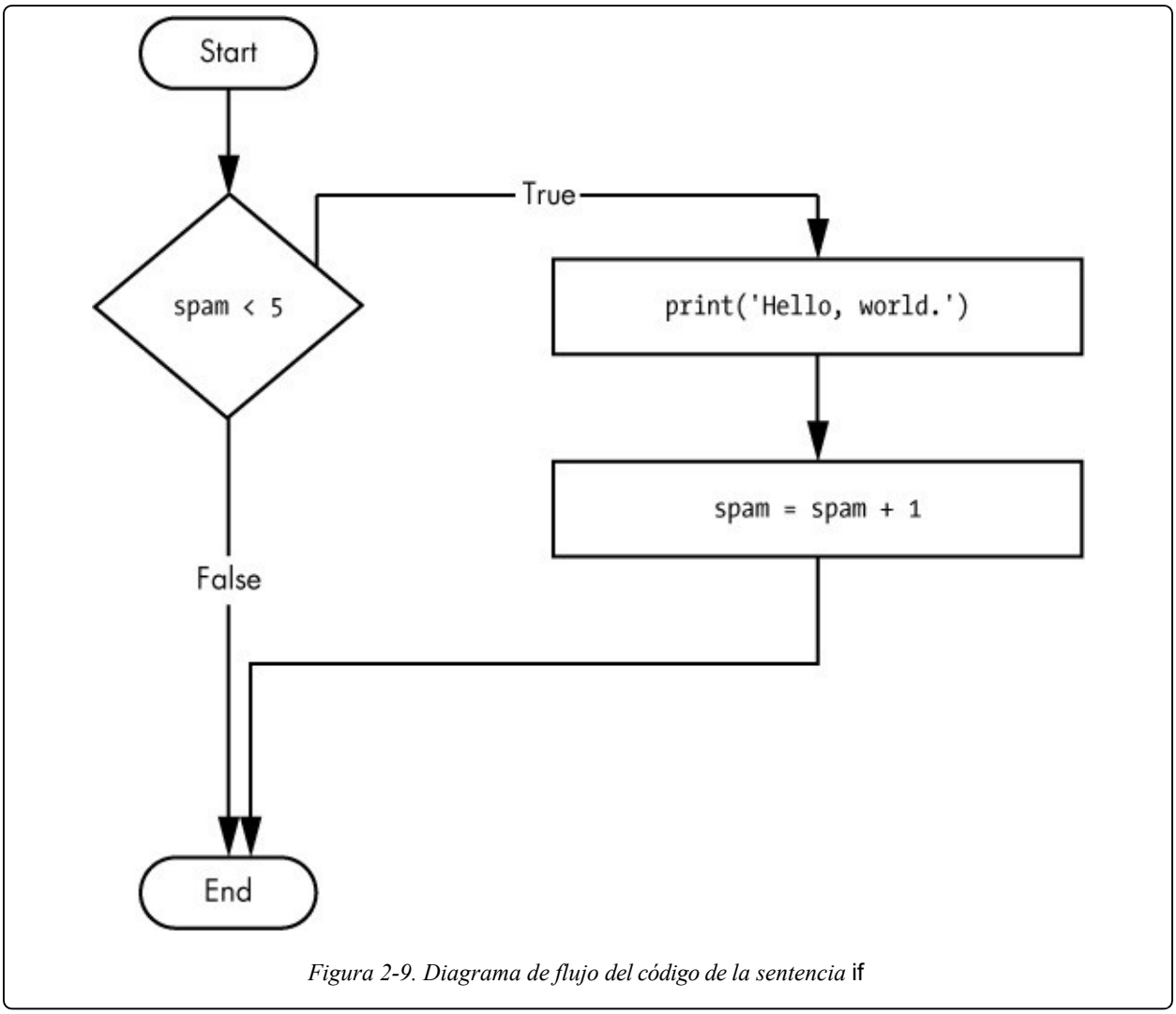


Figura 2-9. Diagrama de flujo del código de la sentencia if

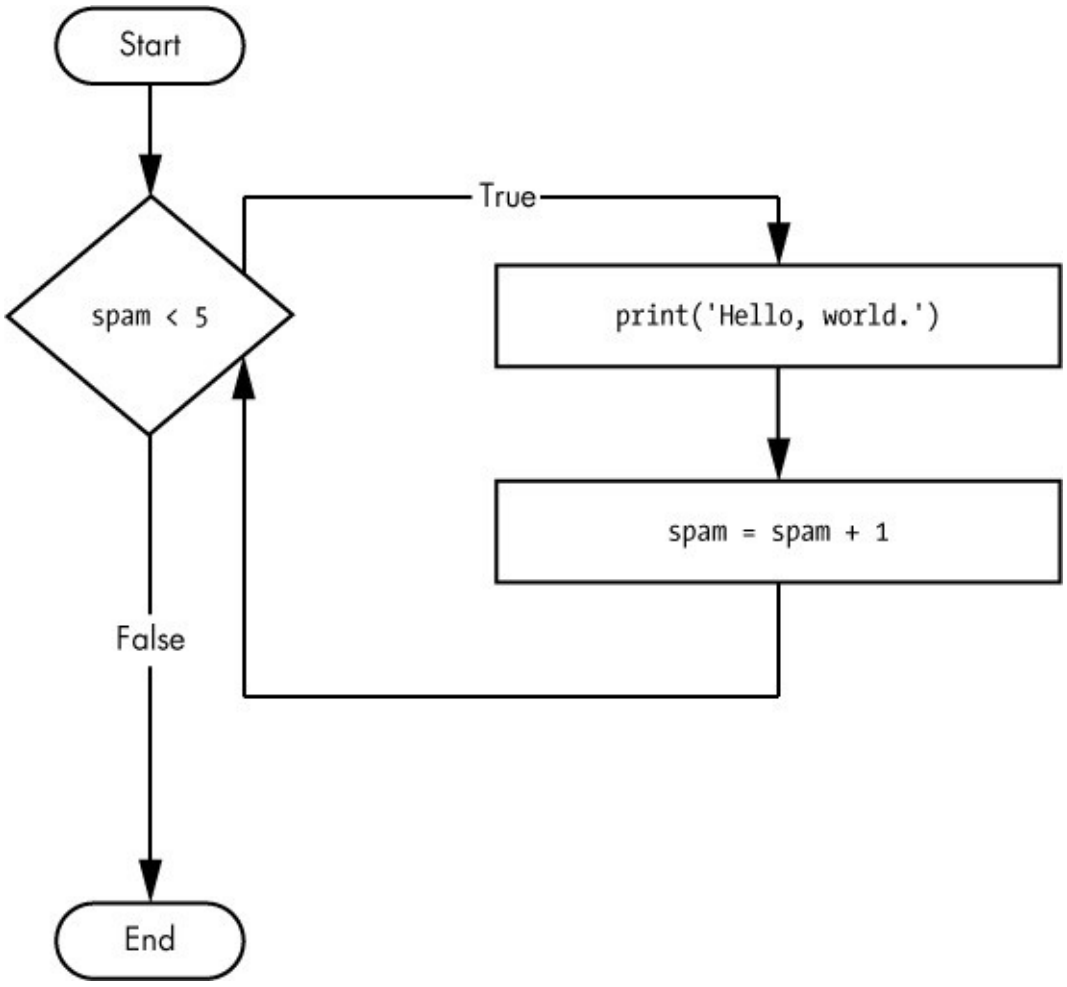


Figura 2-10. Diagrama de flujo del código de la sentencia while

El código con la sentencia **if** comprueba la condición, e imprime Hola, mundo. sólo una vez si esa condición es verdadera. El código con el while, por otro , lo imprimirá cinco veces. Se detiene después de cinco impresiones porque el entero en spam se incrementa en uno al final de cada iteración del bucle, lo que significa que el bucle se ejecutará cinco veces antes de que spam< 5 sea Falso.

En el bucle while, la condición se comprueba siempre al principio de cada *iteración* (es decir, cada vez que se ejecuta el bucle). Si la condición es Verdadera, entonces ejecuta la cláusula, y después, la condición se comprueba de nuevo. La primera vez que se comprueba que la condición es Falsa, se omite la cláusula while.

Un molesto bucle while

He aquí un pequeño programa de ejemplo que le pedirá que escriba, literalmente**, su**

nombre. Seleccione **Archivo► Nueva ventana** para abrir una nueva ventana de editor de archivos, introduzca el siguiente código y guarde el archivo como *tuNombre.py*:

```
mbre= ""
❷ while nombre != 'tu nombre': print('Por
        favor, escribe tu nombre.')
➲       nombre= entrada()
❹ print('!')
```

En primer lugar, el programa establece la variable nombre ❶ a una cadena vacía. Esto es para que la condición nombre != 'tu nombre' se evalúe a True y la ejecución del programa entre en la cláusuladel bucle while.

El código dentro de esta cláusula pide al usuario que escriba su nombre, que se asigna a la variable de nombre ❸. Como ésta es la última línea del bloque, la ejecución retrocede al inicio del bucle while y reevalúa la condición. Si el valor en nombre *no es igual* a la cadena 'tu nombre', entonces la condición es True, y la ejecución entra de nuevo en la cláusula while.

Pero una vez que el usuario escribe **tu nombre**, la condición del bucle while será 'tu nombre' != 'tu nombre', que se evalúa como False. La condición ahora es False, y en lugar de que la ejecución del programa vuelva a entrar en la cláusula del bucle while, se la salta y continúa ejecutando el resto del programa❹. La Figura 2- 11 muestra un diagrama de flujo para el programa *tuNombre.py*.

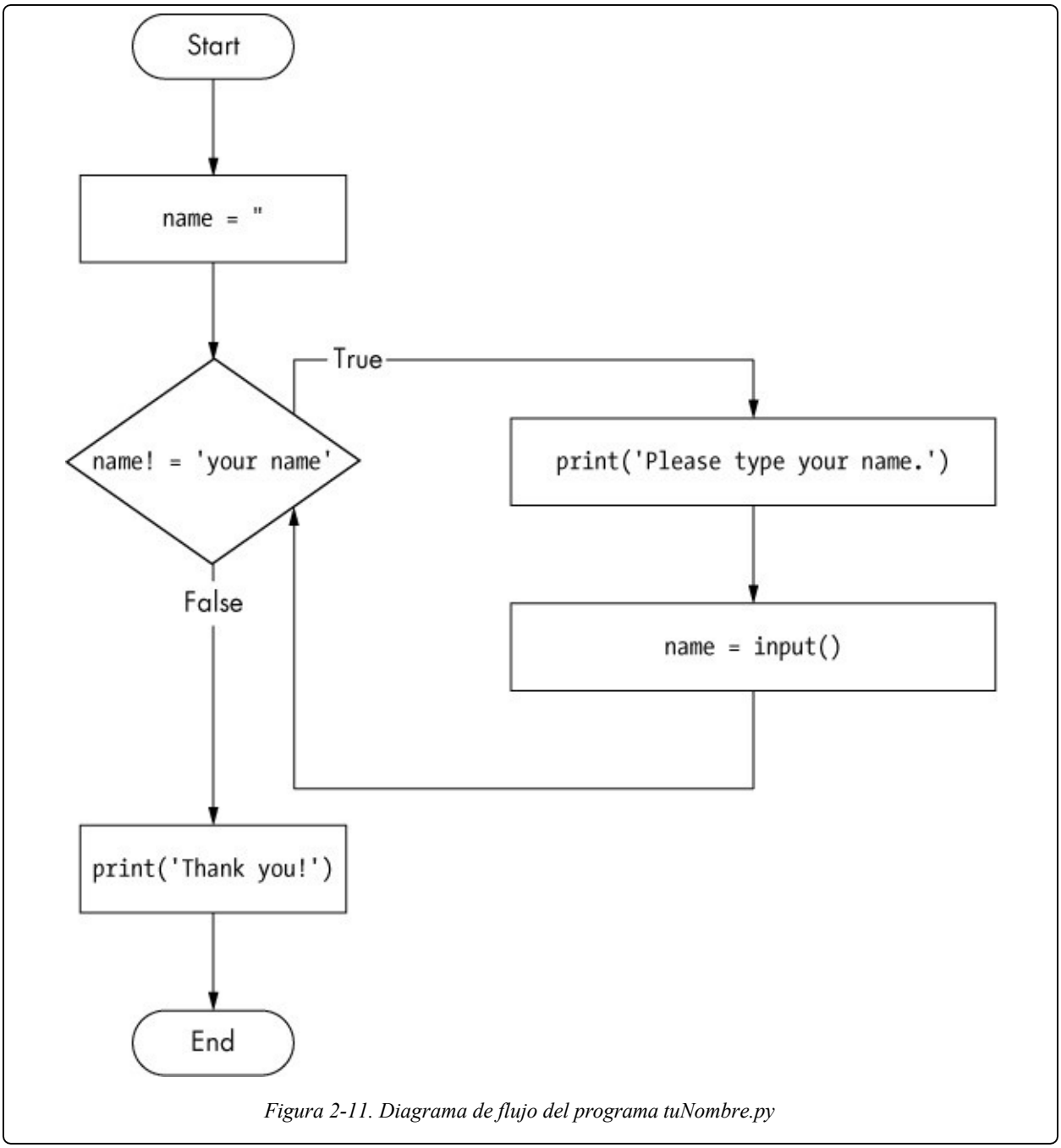


Figura 2-11. Diagrama de flujo del programa tuNombre.py

Ahora, veamos *tuNombre.py* en acción. Pulsa **F5** para ejecutarlo, e introduce algo que no sea **tu nombre** unas cuantas veces antes de darle al programa lo que quiere.

```
Por favor, escriba su nombre.
Al
Por favor, escriba su nombre.
Albert
Por favor, escriba su nombre.
¡¡¡%#@#%*(^&!!!
```

```
Por favor, escriba su nombre.
tu nombre
Gracias.
```

Si nunca introduce **su nombre**, entonces la condición del bucle while nunca será `False`, y el programa seguirá preguntando eternamente. Aquí, la llamada a `input()` permite al usuario introducir la cadena correcta para que el programa siga adelante. En otros programas, la condición puede no cambiar nunca, y eso puede ser un problema. Veamos cómo salir de un bucle `while`.

romper Declaraciones

Existe un atajo para conseguir que la ejecución del programa salga antes de tiempo de la cláusula de un bucle while. Si la ejecución alcanza una sentencia break, sale inmediatamente de la cláusula del bucle while. En código, una sentencia break simplemente contiene la palabra clave break.

Bastante sencillo, ¿verdad? Aquí tienes un programa que hace lo mismo que el anterior, pero utiliza una sentencia break para salir del bucle. Introduce el siguiente código, y guarda el archivo como *tuNombre2.py*:

```
❶ mientras True:
        print('Escriba su nombre.")
❷       nombre= entrada()
➲       si nombre== 'tu nombre":
❹           romper
❺ print('!')
```

La primera línea ❶ crea un *bucle infinito*; es un bucle while cuya condición es siempre True. (La expresión True, después de todo, siempre se evalúa al valor True.) La ejecución del programa siempre entrará en el bucle y saldrá de él sólo cuando se ejecute una sentencia break. (Un bucle infinito que *nunca* sale es un error de programación común).

Al igual que antes, este programa pide al usuario que escriba **su nombre** ❷. Ahora, sin embargo, mientras la ejecución está todavía dentro del bucle while, se ejecuta una if ❸ para comprobar si name es igual a tu nombre. Si esta condición es True, se ejecuta la sentencia break ❹, y la ejecución sale del bucle para imprimir('¡Gracias!')❺. En caso contrario, se salta la cláusula de la sentencia if con la sentencia break, lo que sitúa la ejecución al final del bucle while. En este punto, la ejecución del programa salta de nuevo al inicio de la sentencia while
❶ para volver a comprobar la condición. Dado que esta condición es simplemente el valor booleano True, la ejecución entra en el bucle para pedir al usuario que escriba **su nombre** de nuevo. Vea la Figura 2-12 para el diagrama de flujo de este programa.

Ejecute *suNombre2.py*, e introduzca el mismo texto que introdujo para *suNombre.py*. El programa reescrito debería responder de la misma manera que el original.

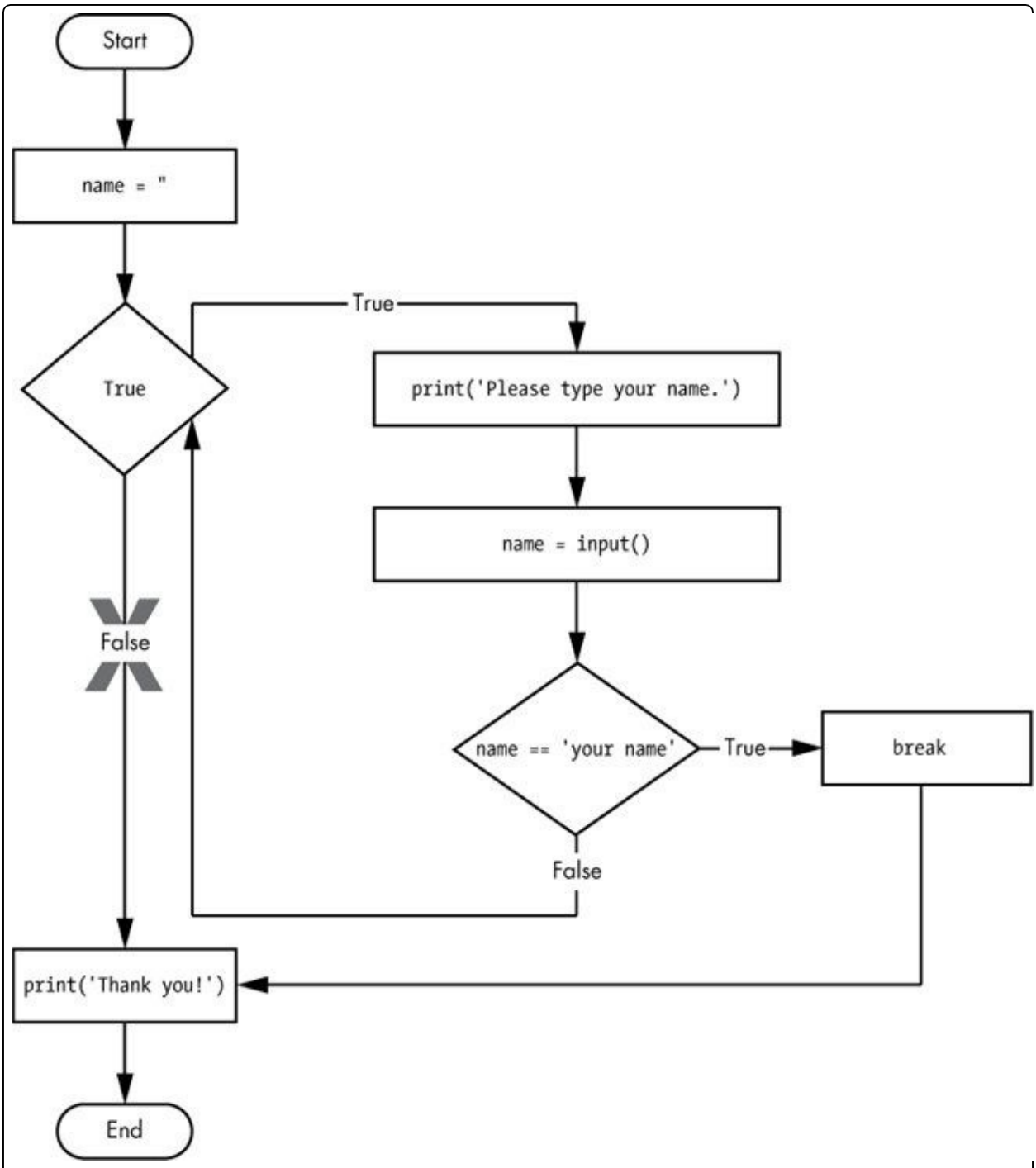


Figura 2-12. El diagrama de flujo para el programa tuNombre2.py con un bucle infinito. Observa que la ruta X lógicamente nunca ocurrirá porque la condición del bucle siempre es True.

continuar Declaraciones

Al igual que las sentencias break, las sentencias continue se utilizan dentro de los bucles. Cuando la ejecución del programa alcanza una sentencia continue, la ejecución del programa salta inmediatamente al inicio del bucle y reevalúa la condición del bucle. (Esto es también lo que ocurre cuando la ejecución alcanza el final del bucle).

¿ATRAPADO EN UN BUCLE INFINITO?

Si alguna vez ejecutas un programa que tiene un error que hace que se atasque en un bucle infinito, pulsa CTRL-C. Esto enviará un error KeyboardInterrupt a tu programa y hará que se detenga inmediatamente. Para probarlo, crea un simple bucle infinito en el editor de archivos, y guárdalo como *infiniteloop.py*.

```
mientras sea verdad:
    print('¡Hola mundo!')
```

Cuando ejecutes este programa, imprimirá ¡Hola mundo! en la pantalla para siempre, porque la condición de la sentencia while es siempre True. En la ventana del shell interactivo de IDLE, sólo hay dos formas de detener este programa: pulsar CTRL-C o seleccionar **Shell▸ restart Shell** en el menú. CTRL-C es útil si alguna vez quieres terminar tu programa inmediatamente, incluso si no está atascado en un bucle infinito.

Usemos continue para escribir un programa que pida un nombre y una contraseña. Introduce el siguiente código en una nueva ventana del editor de archivos y guarda el programa como *pezespada.py*.

```
mientras sea verdad:
        print('¿Quién eres?')
        name= input()
❶       si nombre != 'Joe':
❷           continuar
        print('Hola, Joe. ¿Cuál es la contraseña? (Es un pez.)')
➲       contraseña= input()
        si contraseña== 'pez espada':
❹           romper
❺ print('Acceso concedido.')
```

Si el usuario introduce cualquier nombre que no sea Joe ❶, la sentencia continue ❷ hace que la ejecución del programa salte de nuevo al inicio del bucle. Cuando reevalúe la condición, la ejecución siempre entrará en el bucle, ya que la condición es simplemente el valor True. Una vez que consiguen pasar esa sentencia if, se le pide usuario una contraseña❸. Si la contraseña introducida es swordfish, entonces se ejecuta la sentencia break ❹, y la ejecución salta fuera del bucle while para imprimir Access granted ❺. En caso contrario, la ejecución continúa hasta el final del bucle while, donde se

luego salta de nuevo al inicio del bucle. Consulte la Figura 2-13 para ver el diagrama de flujo de este programa.



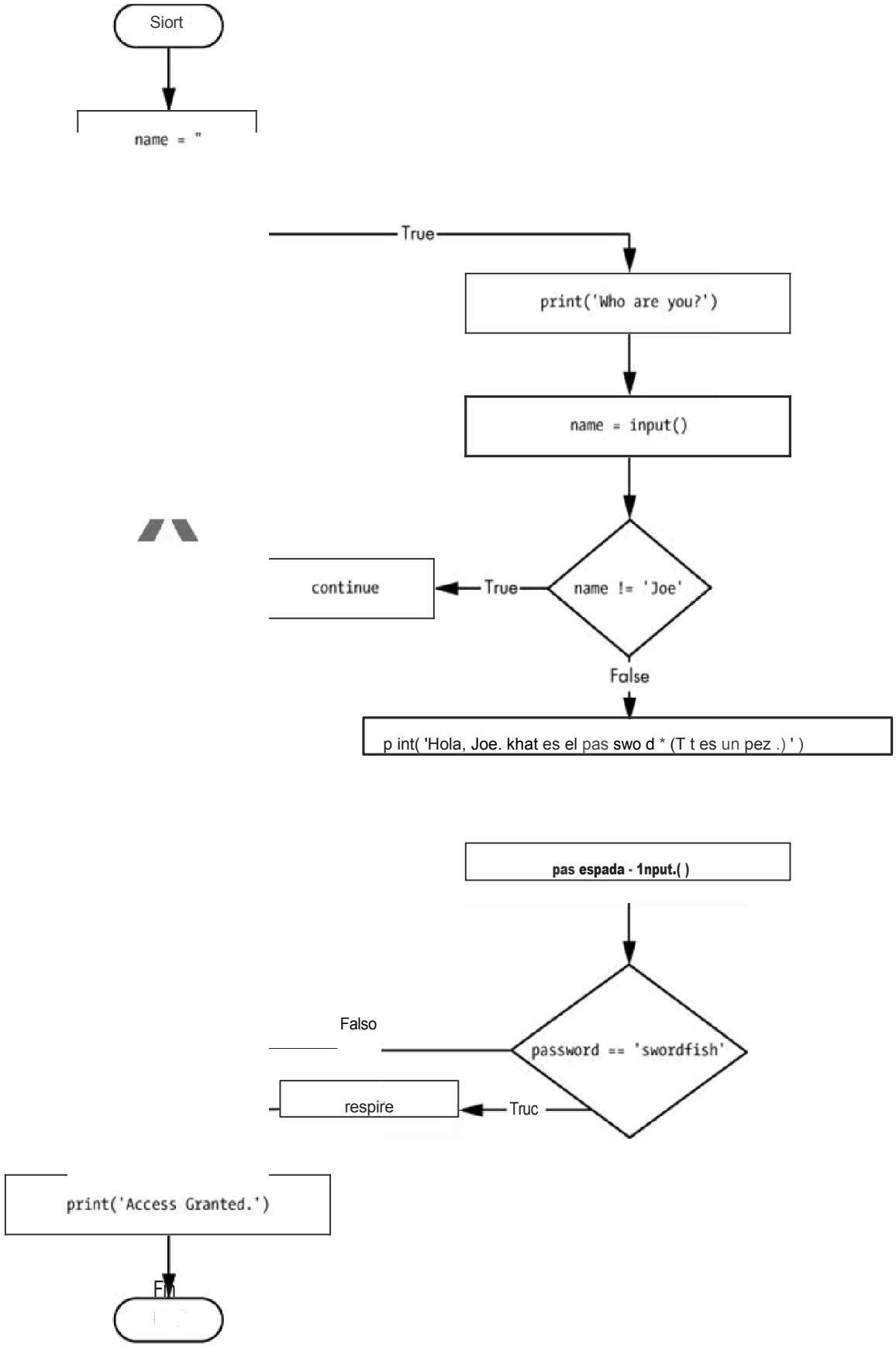

Siort
name = ''
True
print('Who are you?')
name = input()
continue
True
name != 'Joe'
False
p int('Hola, Joe. khat es el pas swo d * (T t es un pez .) ')
pas espada - 1nput.( )
Falso
password == 'swordfish'
respire
Truc
print('Access Granted.')

Figura 2-13. Diagrama de flujo de swordfish.py. La ruta X lógicamente nunca ocurrirá porque la condición del bucle es siempre True.

"VALORES VERDADERO" Y "FALSO

Hay algunos valores en otros tipos de datos que las condiciones considerarán equivalentes a Verdadero y Falso. Cuando se utilizan en condiciones, 0, 0.0 y '' (la cadena vacía) se consideran Falso, mientras que todos los demás valores se consideran Verdadero. Por ejemplo, mire el siguiente programa:

```
nombre= ''
while not name:❶
    print('Introduce tu
    nombre:') name= input()
print('¿Cuántos invitados tendrá?')
numOfGuests= int(input())
if numOfGuests:❷
    print('Asegúrate de tener suficiente espacio para todos tus
invitados.')❸ print('Hecho')
```

Si el usuario introduce una cadena en blanco para el nombre, entonces la condición de la sentencia while será True ❶, y el programa continúa preguntando por un nombre. Si el valor de numOfGuests no es 0❷, entonces se considera que la condición es True, y el programa imprimirá un recordatorio para el usuario❸.

Podría haber escrito not name != '' en lugar de not name, y numOfGuests != 0 en lugar de numOfGuests, pero usar los valores truthy y falsey puede hacer que tu código sea más fácil de leer.

Ejecuta este programa y dale alguna entrada. Hasta que no digas que eres Joe, no debería una contraseña, y una vez que introduzcas la contraseña correctadebería salir.

```
¿Quién es usted?
Estoy bien, GRACIAS. ¿Quién es usted?
¿Quién es usted?
Joe
Hola, Joe. ¿Cuál es la contraseña? (Es un pez.)
Mary
¿Quién es usted?
Joe
Hola, Joe. ¿Cuál es la contraseña? (Es un pez.)
pez espada
Acceso concedido.
```

Bucles for y la función range()

El bucle while se mantiene en bucle mientras su condición es True (que es la razón de su nombre), pero ¿qué pasa si desea ejecutar un bloque de código sólo un cierto número de veces? Puedes hacerlo con una sentencia de bucle for y la función range().

En código, una sentencia for tiene un aspecto similar a for i in range(5): y siempre incluye lo siguiente:

- La palabra clave for
- Un nombre de variable
- La palabra clave in
- Una llamada al método range() con hasta tres enteros pasados a él
- Un colon
- A partir de la línea siguiente, un bloque de código sangrado (llamado cláusula for)

Vamos a crear un nuevo programa llamado *cincoVeces.py* para ayudarte a ver un bucle for en acción.

```
print('Mi nombre
es') for i in
range(5):
    print('Jimmy cinco veces ('+ str(i)+ ')')
```

El código de la cláusula del bucle for se ejecuta cinco veces. La primera vez que se ejecuta, la variable i se pone a 0. La llamada print() en la cláusula imprimirá Jimmy cinco veces (0). Después de que Python termina una iteración a través de todo el código dentro de la cláusula del bucle for, la ejecución vuelve al principio del bucle, y la sentencia for incrementa i en uno. Esta es la razón por la que range(5) da como resultado cinco iteraciones a través de la cláusula, con i poniéndose a 0, luego a 1, luego a 2, luego a 3, y luego a 4. La variable i irá hasta, pero no incluirá, el entero pasado a range().
La Figura 2-14 muestra un diagrama de flujo del programa *fiveTimes.py*.

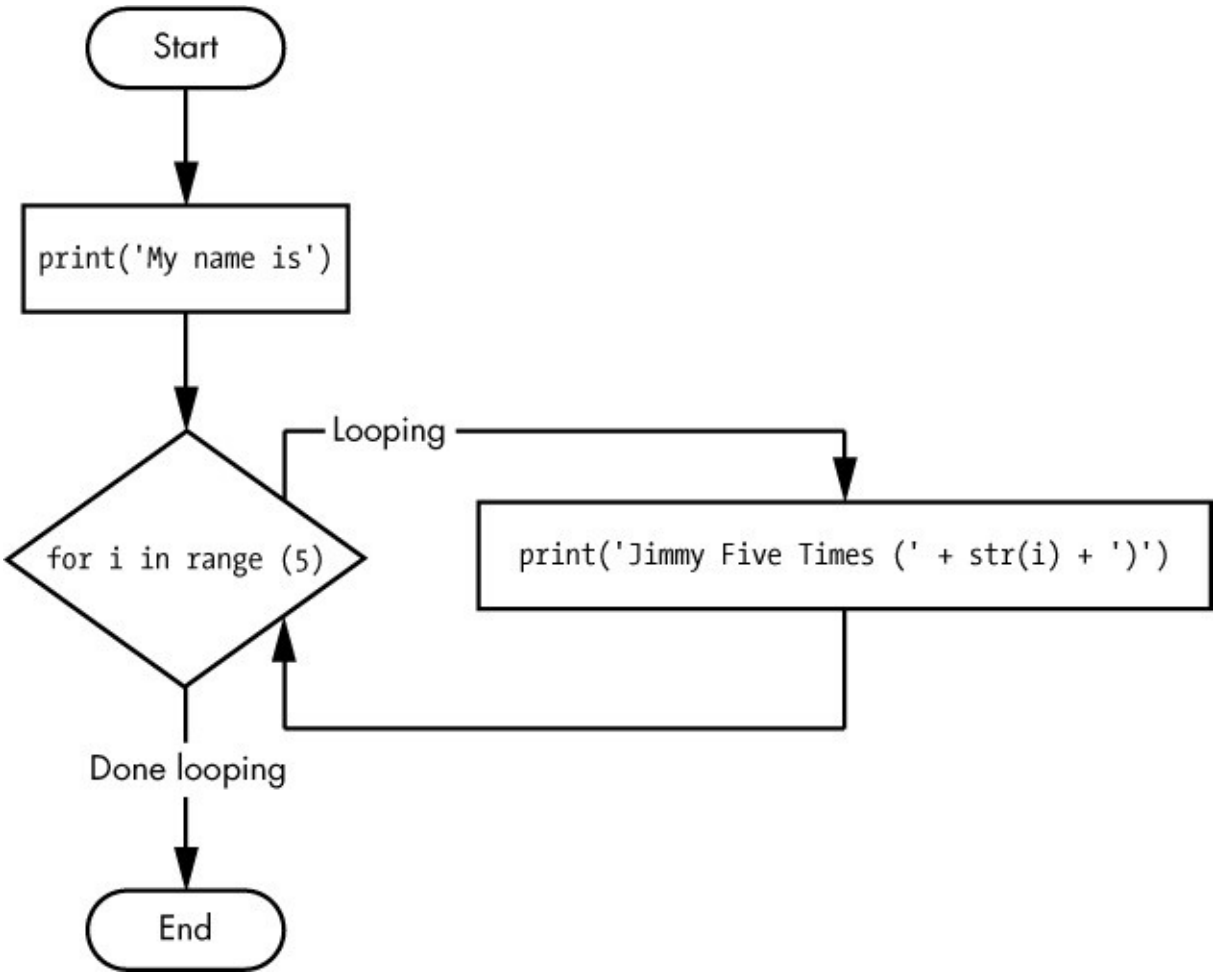


Figura 2-14. Diagrama de flujo de fiveTimes.py

Cuando ejecutes este programa, debería imprimir Jimmy cinco veces seguido del valor de **i** cinco veces antes de salir del bucle for.

```
Me llamo
Jimmy Cinco Veces (0)
Jimmy Cinco Veces (1)
Jimmy Cinco Veces (2)
Jimmy Cinco Veces (3)
Jimmy Cinco Veces (4)
```

NOTA

También puede utilizar las sentencias break *y* continue *dentro de los bucles* for*. La sentencia* continue *continuará hasta el siguiente valor del contador del bucle for, como si la ejecución del programa hubiera llegado al final del bucle y volviera al principio. De hecho, sólo puede utilizar las sentencias* continue *y* break *dentro de los bucles* while *y* for*. Si intentas usar estas sentencias en otro lugar, Python te dará un error.*

Como otro ejemplo de bucle for, considere esta historia sobre el matemático Karl Friedrich Gauss. Cuando Gauss era niño, un profesor quería dar a la clase un poco de trabajo. El profesor les dijo que sumaran todos los números del 0 al 100.
El joven Gauss ideó un ingenioso truco para averiguar la respuesta en unos segundos, pero tú puedes escribir un programa en Python con un bucle for que haga este cálculo por ti.

```
❶ total= 0
❷ for num in range(101):
➲       total= total+ num
❹ print(total)
```

El resultado debe ser 5.050. Cuando el programa se inicia por primera vez, la variable total se establece en 0❶. A continuación, el bucle for ❷ ejecuta total= total+ num ❸ 100 veces. En el momento en que el bucle ha terminado todas sus 100 iteraciones, cada entero de 0 a 100 se habrá añadido al total. En este punto, el total se imprime en la pantalla
❹. Incluso en los ordenadores más lentos, este programa tarda menos de un segundo en completo.

(El joven Gauss dedujo que había 50 pares de números que sumaban 100: 1 + 99, 2 + 98, 3 + 97, y así sucesivamente, hasta 49 + 51.) Como 50 × 100 son 5.000, al sumar ese 50 del medio, la suma de todos los números del 0 al 100 es 5.050.) Niño listo!)

Un bucle while equivalente

Puedes usar un bucle while para hacer lo mismo que un bucle for; los bucles for son más concisos. Reescribamos *cincoVeces.py* para usar un bucle while equivalente a un bucle for.

```
print('Mi nombre
es') i= 0
mientras i< 5:
    print('Jimmy cinco veces ('+ str(i)+ ')')
    i= i+ 1
```

Si ejecutas este programa, la salida debería ser la misma que la *de fiveTimes.py* que utiliza un bucle for.

Argumentos de inicio, parada y paso de range()

Algunas funciones pueden invocarse con varios argumentos separados por una coma, y range() es una de ellas. Esto le permite cambiar el entero pasado a range()
para seguir cualquier secuencia de números enteros, incluso comenzando en un número distinto de

cero.

```
para i en rango(12, 16):
    print(i)
```

El primer argumento será donde empieza la variable del bucle for, y el segundo argumento será hasta, pero sin incluir, el número en el que parar.

```
12
13
14
15
```

La función range() también se puede llamar con tres argumentos. Los dos primeros argumentos serán los valores inicial y final, y el tercero será el *argumento paso*. El paso es la cantidad en que se incrementa la variable después de cada iteración.

```
para i en rango(0, 10,
    2): print(i)
```

Así, llamando a range(0, 10, 2) contará de cero a ocho por intervalos de dos.

```
0
2
4
6
8
```

La función range() es flexible en la secuencia de números que produce para los bucles for. Por ejemplo (nunca me disculpo por mis juegos de palabras), puede incluso utilizar un número negativo para el argumento paso para hacer que el bucle for cuente hacia abajo en lugar de hacia arriba.

```
for i in range(5, -1, -1):
    print(i)
```

Ejecutando un bucle for para imprimir i con range(5, -1, -1) debería imprimir desde cinco hasta cero.

```
5
4
3
2
1
0
```

Importación de módulos

Todos los programas Python pueden llamar a un conjunto básico de funciones llamadas *funciones incorporadas*, incluyendo las funciones print(), input() y len() que has visto antes.
Python también viene con un conjunto de módulos llamados *biblioteca estándar*. Cada es un programa Python que contiene un grupo de funciones relacionadas que pueden ser incrustadas en tus programas. Por ejemplo, el módulo math tiene funciones relacionadas con las matemáticas, el módulo random tiene funciones relacionadas con los números aleatorios, etc.

Antes de poder utilizar las funciones de un módulo, debe importar el módulo con una sentencia import. En código, una sentencia import consiste en lo siguiente:

- La palabra clave de importación
- El nombre del módulo
- Opcionalmente, más nombres de módulos, siempre que estén separados por comas

Una vez que importas un módulo, puedes utilizar todas las funciones interesantes de ese módulo. Vamos a intentarlo con el módulo random, que nos dará acceso a la función random.randint().

Introduce este código en el editor de archivos y guárdalo como *printRandom.py*:

```
importar aleatorio
for i in range(5):
    print(random.randint(1, 10))
```

Cuando ejecutes este programa, la salida será algo parecido a esto:

```
4
1
8
4
1
```

La llamada a la función random.randint() evalúa a un valor entero aleatorio entre los dos enteros que le pases. Como randint() está en el módulo random, primero debes escribir **random.** delante del nombre de la función para decirle a Python que busque esta función dentro del módulo random.

Este es un ejemplo de una sentencia import que importa cuatro módulos diferentes:

```
import random, sys, os, math
```

Ahora podemos utilizar cualquiera de las funciones de estos cuatro módulos. Aprenderemos más sobre ellas más adelante en el libro.

de la importación Statements

Una forma alternativa de la sentencia `import` se compone de la palabra clave `from`, seguida del nombre del módulo, la palabra clave `import` y una estrella; por ejemplo, `from random import *`.

Con esta forma de sentencia `import` las llamadas a funciones en `random` no necesitarán el prefijo `random.` Sin embargo, el uso del nombre completo hace que el código sea más legible, por lo que es mejor utilizar la forma normal de la sentencia `import`.

Finalización anticipada de un programa con sys.exit()

El último concepto de control de flujo a cubrir es cómo terminar el programa. Esto siempre ocurre si la ejecución del programa llega al final de las instrucciones. Sin embargo, puedes hacer que el programa termine, o salga, llamando a la función sys.exit(). Como esta función está en el módulo sys, tienes que importar sys antes de que tu programa pueda usarla.

Abra una nueva ventana del editor de archivos e introduzca el siguiente código, guardándolo como *exitEjemplo.py*:

```
importar sistema

mientras sea verdad:
    print('Escriba exit para salir.')
    response= input()
    if response== 'exit':
        sys.exit()
    print('Has escrito '+ respuesta+ '.')
```

Ejecute este programa en IDLE. Este programa tiene un bucle infinito sin una sentencia break en su interior. La única manera en que este programa terminará es si el usuario ingresa exit, causando que sys.exit() sea llamado. Cuando response es igual a exit, el programa termina. Dado que la variable respuesta es establecida por la función input(), el usuario debe ingresar exit para detener el .

Resumen

Utilizando expresiones que se evalúan como Verdadero o Falso (también llamadas *condiciones*), puedes escribir programas que toman decisiones sobre qué código ejecutar y qué código omitir. También puedes ejecutar código una y otra vez en un bucle mientras una determinada condición se evalúa como True. Las sentencias break y continue son útiles si necesitas salir de un bucle o volver al principio.

Estas sentencias de control de flujo te permitirán escribir programas mucho más inteligentes. Hay otro tipo de control de flujo que puedes lograr escribiendo tus propias funciones, que es el tema del próximo capítulo.

Preguntas prácticas

P: 1. ¿Cuáles son los dos valores del tipo de datos booleano? ¿Cómo se ?

P: 2. ¿Cuáles son los tres operadores booleanos?

P: 3. Escribe las tablas de verdad de cada operador booleano (es decir, todas las combinaciones posibles de valores booleanos para el operador y a qué se evalúan).

P: 4. ¿A cuánto se evalúan las siguientes expresiones?

```
(5> 4) y (3== 5)
no (5> 4)
(5> 4) o (3== 5)
not ((5> 4) or (3== 5))
(Verdadero y Verdadero) y (Verdadero ==
Falso) (no Falso) o (no Verdadero)
```

P: 5. ¿Cuáles son los seis operadores de comparación?

P: 6. ¿Cuál es la diferencia entre el operador igual a y el operador de asignación?

P: 7. Explique qué es una condición y dónde utilizaría una.

P: 8. Identifique los tres bloques de este código:

```
spam= 0
si spam == 10:
    print('huevos'
    ) si spam> 5:
        print('bacon') else:
        print("jamón")
    print("spam")
print("spam")
```

P: 9. Escribe un código que imprima Hola si 1 está almacenado en spam, que imprima Hola si 2 está almacenado en spam, y que imprima
Saludos! si algo más se almacena en spam.

P: 10. ¿Qué puedes pulsar si tu programa está atascado en un bucle infinito?

P: 11. ¿Cuál es la diferencia entre break y continue?

Q: 12. ¿Cuál es la diferencia entre range(10), range(0, 10), y range(0, 10, 1) en un bucle for?

Q: 13. Escribe un programa corto que imprima los números del 1 al 10 utilizando un bucle for. Luego escriba un programa equivalente

programa que imprime los números del 1 al 10 utilizando un bucle while.

P: 14. Si tuvieras una función llamada bacon() dentro de un módulo llamado spam, ¿cómo la llamarías después de importar spam?

Crédito extra: Busca las funciones round() y abs() en Internet y averigua lo que hacen. Experimenta con ellas en el intérprete de comandos interactivo.

Capítulo 3. Funciones Funciones

Ya estás familiarizado con las funciones print(), input() y len() de los capítulos anteriores. Python proporciona varias funciones incorporadas como éstas, pero también puedes escribir tus propias funciones. Una *función* es como un mini-programa dentro de un programa.

Para entender mejor cómo funcionan las funciones, vamos a crear una. Escribe este programa en el editor de archivos y guárdalo como *holaFunc.py*:

```
❶ def hola():
❷       print('¡Hola!')
        print('¡¡¡Hola!!!')
        print('.')
➲  hola()
   hola()
   hola()
```

La primera línea es una sentencia def ❶, que define una función llamada hola(). El código del bloque que sigue a la sentencia def ❷ es el cuerpo de la función. Este código se ejecuta cuando se llama a la función, no cuando la función se define por primera vez.

Las líneas hello() después de la función ❸ son llamadas a funciones. En código, una llamada a función es simplemente el nombre de la función seguido de paréntesis, posiblemente con algún número de argumentos entre los paréntesis. Cuando la ejecución del programa llega a estas , saltará a la línea superior de la función y comenzará a ejecutar el código allí. Cuando llega al final de la función, la ejecución vuelve a la línea que llamó a la función y continúa moviéndose por el código como antes.

Como este programa llama a hola() tres veces, el código de la función hola() se ejecuta tres veces. Cuando se ejecuta este , la salida tiene este aspecto:

```
¡Hola! ¡Hola!
Hola que tal.
¡Hola!
¡¡¡Hola!!! Hola
que tal. ¡Hola!
¡¡¡Hola!!!
Hola.
```

Uno de los principales objetivos de las funciones es agrupar código que se ejecuta varias veces.

Sin una función definida, tendrías que copiar y pegar este código cada vez, y el programa se vería así:

```
print('¡Hola!')
print('!!!') print('Hola.')
print('¡Hola!')
print('!!!') print('Hola.')
print('¡Hola!')
print('¡¡¡Hola!!!') print('.')
```

En general, siempre querrás evitar duplicar código, porque si alguna vez decides actualizar el código -si, por ejemplo, encuentras un error que necesitas corregir- tendrás que acordarte de cambiar el código en todos los sitios donde lo copiaste.

A medida que adquieras más experiencia en programación, a menudo te encontrarás *deduplicando* código, lo que significa deshacerte de código duplicado o copiado y pegado. La deduplicación hace que tus programas sean más cortos, más fáciles de leer y más fáciles de actualizar.

def Declaraciones con parámetros

Cuando se llama a la función print() o len(), se pasan valores, llamados *argumentos* en este contexto, escribiéndolos entre paréntesis. También puedes definir tus propias funciones que acepten argumentos. Escribe este ejemplo en el editor de archivos y guárdalo como *holaFunc2.py*:

```
❶ def hola(nombre):
❷       print('Hola '+ nombre)

➲  hola('Alice')
   hola('Bob')
```

Cuando ejecutas este programa, la salida tiene este aspecto:

```
Hola Alice
Hola Bob
```

La definición de la función hola() en este programa tiene un parámetro llamado nombre❶. Un *parámetro* es una variable en la que se almacena un argumento cuando se llama a una función. La primera vez que se llama a la función hola(), es con el argumento❸. La ejecución del programa entra en la función, y el nombre de la variable se establece automáticamente a 'Alice', que es lo que se imprime por la sentencia print() ❷.

Una cosa especial a tener en cuenta sobre los parámetros es que el valor almacenado en un parámetro se olvida cuando la función retorna. Por ejemplo, si añadieras print(nombre) después de hello('Bob') en el programa anterior, el programa te daría un NameError porque no hay ninguna variable llamada nombre. Esta variable fue destruida después de que la llamada a la función hola('Bob') hubiera , por lo que print(nombre) haría referencia a una variable nombre que no existe.

Esto es similar a cómo las variables de un programa se olvidan cuando el programa termina. Hablaré más sobre por qué ocurre esto más adelante en el capítulo, cuando hable de lo que es el ámbito local de una función.

Valores de retorno y sentencias return

Cuando se llama a la función `len()` y se le pasa un argumento como `"Hola"`, la llamada a la función se evalúa con el valor entero `5`, que es la longitud de la cadena se le ha pasado. En general, el valor al que se evalúa una llamada a función se denomina *valor de retorno* de la función.

Al crear una función con la sentencia `def`, puede especificar el valor de retorno con una sentencia `return`. Una sentencia `return` consiste en lo siguiente:

- La palabra clave `return`
- El valor o expresión que debe devolver la función

Cuando se utiliza una expresión con una sentencia `return`, el valor de retorno es aquello a lo que se evalúa esta expresión. Por ejemplo, el siguiente programa define una función que devuelve una cadena diferente dependiendo del número que se le pase como argumento. Escriba este código en el editor de archivos y guárdelo como *magic8Ball.py*:

```
❶ importar aleatorio
❷ def obtenerRespuesta(numeroRespuesta):
➲       si answerNumber== 1:
            return 'Es cierto' elif
        answerNumber== 2:
            return 'Es decididamente
        así' elif answerNumber== 3:
            devolver "Sí
        elif answerNumber== 4:
            return 'Respuesta confusa
        inténtalo de nuevo' elif
        answerNumber== 5:
            return 'Vuelve a
        preguntar más tarde' elif
        answerNumber== 6:
            return 'Concéntrate y vuelve a preguntar'
        elif answerNumber== 7:
            return "Mi respuesta es
        no' elif answerNumber== 8:
            return 'Outlook no tan bueno'
        elif answerNumber== 9:
            return "Muy dudoso

❹ r= random.randint(1, 9)
❺ fortuna= getAnswer(r)
❻ print(fortuna)
```

Cuando se inicia este programa, Python importa primero el módulo random ❶. Luego se define la función getAnswer() ❷. Debido a que la función está siendo definida (y no llamada), la ejecución se salta el código en ella. A continuación, se llama a la función random.randint() con dos argumentos, 1 y 9 ❹. Se evalúa a un entero aleatorio entre 1 y 9 (incluyendo el propio 1 y 9), y este valor se almacena en una variable llamada r.

Se llama a la función getAnswer() con r como argumento❺. La ejecución del programa se mueve a la parte superior de la función getAnswer()❸, y el valor r se almacena en un parámetro llamado answerNumber. Entonces, dependiendo de este valor en answerNumber, la función devuelve uno de los muchos posibles valores de cadena. La ejecución del programa vuelve a la línea en la parte inferior del programa que originalmente llamó a getAnswer()❺. La cadena devuelta se asigna a una variable llamada fortuna, que luego se pasa a una llamada a print() ❻ y se imprime en la pantalla.

Tenga en cuenta que como puede pasar valores de retorno como argumento a otra llamada de función, podría acortar estas tres líneas:

```
r= random.randint(1, 9) fortuna
= getAnswer(r) print(fortuna)
```

a esta única línea equivalente:

```
print(getAnswer(random.randint(1, 9)))
```

Recuerde que las expresiones se componen de valores y operadores. Una llamada a una función puede utilizarse en una expresión porque se evalúa a su valor de retorno.

El valor Ninguno

En Python existe un valor llamado None, que representa la ausencia de un valor. Ninguno es el único valor del tipo de datos NoneType. (Otros lenguajes de programación podrían llamar a este valor null, nil o undefined.) Al igual que los valores booleanos True y False, None debe escribirse con *N* mayúscula.

Este valor-sin-valor puede ser útil cuando se necesita almacenar algo que no se confunda con un valor real en una variable. Un lugar donde se utiliza None es como valor de retorno de print(). La función print() muestra texto en la pantalla, pero no necesita devolver nada de la misma manera que len() o input(). Pero como todas las llamadas a funciones necesitan evaluarse a un valor de retorno, print() devuelve None. Para ver esto en acción, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam = print('¡Hola!')
¡Hola!
>>> Ninguno== spam
Verdadero
```

Entre bastidores, Python añade return None al final de cualquier definición de función sin return. Esto es similar a cómo un bucle while o for termina implícitamente con una sentencia continue. Además, si usas una sentencia return sin un valor (es decir, sólo la palabra clave return por sí misma), entonces se devuelve None.

Palabra clave Argumentos e print()

La mayoría de los argumentos se identifican por su posición en la llamada a la función. Por ejemplo, random.randint(1, 10) es diferente de random.randint(10, 1). La llamada a la función random.randint(1, 10) devolverá un entero aleatorio entre 1 y 10, porque el primer argumento es el extremo inferior del rango y el segundo argumento es el extremo superior (mientras que random.randint(10, 1) provoca un error).

Sin embargo, *los argumentos de palabra clave* identifican por la palabra clave antepuesta en la llamada a la función. Los argumentos de palabra clave se utilizan a menudo para parámetros opcionales. Por ejemplo, la función print() tiene los parámetros opcionales end y sep para especificar lo que debe imprimirse al final de sus argumentos y entre sus argumentos (separándolos), respectivamente.

Si ejecutas el siguiente programa:

```
print('Hola')
print("Mundo")
```

el resultado sería el siguiente:

```
Hola
Mundo
```

Las dos cadenas aparecen en líneas separadas porque la función print() añade automáticamente un carácter de nueva línea al final de la cadena que se le pasa. Sin embargo, puede establecer el argumento de la palabra clave end para cambiar esto a una cadena diferente. Por ejemplo, si el programa fuera este

```
print('Hola', end='')
print("Mundo")
```

el resultado sería el siguiente:

```
HolaMundo
```

La salida se imprime en una sola línea porque ya no se imprime una nueva línea después de 'Hola'. En su lugar, se imprime la cadena en blanco. Esto es útil si necesita desactivar la nueva línea que se añade al final de cada llamada a la función print().

Del mismo modo, cuando pase varios valores de cadena a print(), la función los separará automáticamente con un solo espacio. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> print('GATOS', 'PERROS', 'ratones')
gatos perros ratones
```

Pero puede reemplazar la cadena de separación predeterminada pasando el argumento de la palabra clave sep. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> print('GATOS', 'PERROS', 'ratones', sep=',')
gatos,perros,ratones
```

También puedes añadir argumentos de palabra clave a las funciones que escribas, pero primero tendrás que aprender sobre los tipos de datos lista y diccionario en los dos próximos capítulos. Por ahora, sólo tienes que saber que algunas funciones tienen argumentos de palabra clave opcionales que se pueden especificar cuando se llama a la función.

Ámbito local y mundial

Los parámetros y variables que se asignan en una función llamada se dice que existen en el *ámbito local* de esa función. Las variables que se asignan fuera de todas las funciones se dice que existen en el *ámbito global*. Una variable que existe en un ámbito local se denomina *variable local*, mientras que una variable que existe en el ámbito global se denomina *variable global*. Una variable debe ser una u ; no puede ser tanto local como global.

Piense en un *ámbito* como un contenedor de variables. Cuando se destruye un ámbito, todos los valores almacenados en las variables del ámbito se olvidan. Sólo hay un ámbito global, y se crea cuando se inicia el programa. Cuando tu programa termina, el ámbito global se destruye, y todas sus variables se olvidan.
De lo contrario, la próxima vez que ejecute su programa, las variables recordarán sus valores de la última vez que lo ejecutó.

Cada vez que se llama a una función se crea un ámbito local. Cualquier variable asignada en esta función existe dentro del ámbito local. Cuando la función retorna, el ámbito local se destruye, y estas variables se olvidan. La próxima vez que se llame a esta función, las variables locales no recordarán los valores almacenados en ellas desde la última vez que se llamó a la función.

Los visores son importantes por varias razones:

- El código en el ámbito global no puede utilizar ninguna variable local.
- Sin embargo, un ámbito local puede acceder a variables globales.
- El código en el ámbito local de una función no puede utilizar variables en ningún otro ámbito local.
- Puedes usar el mismo nombre para diferentes variables si están en diferentes ámbitos. Es decir, puede haber una variable local llamada spam y una variable global también llamada spam.

La razón por la que Python tiene diferentes ámbitos en lugar de hacer que todo sea una variable global es para que cuando las variables sean modificadas por el código en una llamada particular a una función, la función interactúe con el resto del programa sólo a través de sus parámetros y el valor de retorno. Esto reduce la lista de líneas de código que pueden estar causando un error. Si su programa no contiene nada más que variables globales y tiene un error debido a que una variable se ha establecido a un valor incorrecto, entonces sería difícil averiguar dónde se estableció este valor incorrecto. Podría haberse establecido en cualquier parte del programa, ¡y tu programa podría cientos o miles de líneas! Pero si el fallo se debe a una variable local con un incorrecto, sabrás que sólo el código de esa función podría haberla establecido incorrectamente.

Aunque el uso de variables globales en programas pequeños está bien, es un mal hábito depender de variables globales a medida que tus programas se hacen más y más grandes.

Las variables locales no pueden utilizarse en el ámbito global

Considere este programa, que provocará un error cuando lo ejecute:

```
def spam():
    huevos= 31337
spam()
print(huevos)
```

Si ejecutas este programa, la salida tendrá este aspecto:

```
Traceback (llamada más reciente):
  File "C:/test3784.py", line 4, in <module>
    print(huevos)
NameError: el nombre 'eggs' no está definido
```

El error se produce porque la variable eggs sólo existe en el ámbito local creado cuando se llama a spam(). Una vez que la ejecución del programa vuelve de spam, ese ámbito local se destruye, y ya no existe una variable llamada eggs. Así que cuando tu programa intenta ejecutar print(eggs), Python te da un error diciendo
que los huevos no están definidos. Esto tiene sentido si lo piensas; cuando la ejecución del programa está en el ámbito global, no existen ámbitos locales, por lo que no puede haber variables locales. Por eso en el ámbito global sólo se pueden utilizar variables globales.

Los ámbitos locales no pueden utilizar variables de otros ámbitos locales

Se crea un nuevo ámbito local cada vez que se llama a una función, incluso cuando se llama a una función desde otra función. Considere este programa:

```
def spam():
❶       huevos= 99
❷       bacon()
➲       print(huevos)

   def bacon():
         jamón= 101
❹       huevos= 0

❺ spam()
```

Cuando se inicia el programa, se llama a la función spam() ❺, y se crea un ámbito local. La variable local eggs ❶ se establece en 99. A continuación se llama a la función bacon() ❷, y se crea un segundo ámbito local. Múltiples ámbitos locales pueden existir al mismo tiempo. En este nuevo ámbito local, la variable local jamón se establece en 101, y se crea una función
la variable local eggs-que es diferente de la del ámbito local de spam()-también se crea ❹ y se establece en 0.

Cuando vuelve bacon(), se destruye el ámbito local de esa llamada. La ejecución del programa continúa en la función spam() para imprimir el valor de huevos ❸, y como el ámbito local para la llamada a spam() todavía existe aquí, la variable huevos se establece en 99. Esto es lo que imprime el programa.

El resultado es que las variables locales de una función están completamente separadas de las variables locales de otra función.

Las variables globales pueden leerse desde un ámbito local

Considere el siguiente programa:

```
def spam():
    print(huevos)
huevos= 42
spam()
print(huevos)
```

Como no hay ningún parámetro llamado eggs ni ningún código que asigne un valor a eggs en la función spam(), cuando se usa eggs en spam(), Python lo considera una referencia a la variable global eggs. Esta es la razón por la que se imprime 42 cuando se ejecuta el programa anterior.

Variables locales y globales con el mismo nombre

Para simplificarte la vida, evita usar variables locales que tengan el mismo nombre que una variable global u otra variable local. Pero técnicamente, es perfectamente legal hacerlo en Python. Para ver qué ocurre, escribe el siguiente código en el editor de archivos y guárdalo *como sameName.py*:

```
def spam():
❶       huevos= 'spam local'
        print(eggs) # imprime 'spam local'
    def bacon():

❷       huevos= 'bacon local'
        print(huevos) # imprime 'bacon local'
        spam()
        print(huevos) # imprime 'bacon local'

➲   huevos= 'global'
    bacon()
    print(eggs) # imprime 'global'
```

Cuando ejecutas este programa, sale lo siguiente:

```
bacon local
spam local
bacon local
global
```

En realidad hay tres variables diferentes en este programa, pero confusamente todas se llaman huevos. Las variables son las siguientes:

❶ Una variable llamada huevos que existe en un ámbito local cuando se llama a spam().

❷ Una variable llamada huevos que existe en un ámbito local cuando se llama a bacon().

➲ Una variable llamada huevos que existe en el ámbito global.

Dado que estas tres variables tienen el mismo nombre, puede resultar saber cuál se está utilizando en cada momento. Por este motivo, debes evitar utilizar el mismo nombre de variable en diferentes ámbitos.

La Declaración global

Si necesitas modificar una variable global desde dentro de una función, usa la sentencia global. Si tienes una línea como global eggs en la parte superior de una función, le dice a Python, "En esta función, eggs se refiere a la variable global, así que no crees una variable local con este nombre". Por ejemplo, escribe el siguiente código en el editor de archivos y guárdalo *como sameName2.py*:

```
def spam():
❶     huevos globales
❷     huevos= "spam"

  huevos= 'global'
  spam()
  print(huevos)
```

Cuando ejecutes este programa, la llamada final a print() mostrará lo siguiente:

```
spam
```

Debido a que eggs se declara global en la parte superior de spam()❶, cuando eggs se establece a "spam"❷, esta asignación se realiza a eggs de ámbito global. No se crea ninguna variable local eggs.

Existen cuatro reglas para saber si una variable está en un local o global:

1. Si una variable se utiliza en el ámbito global (es decir, fuera de todas las funciones), siempre es una variable global.
2. Si existe una declaración global para esa variable en una función, se trata de una variable global.
3. De lo contrario, si la variable se utiliza en una sentencia de asignación en la función, es una variable local.
4. Pero si la variable no se utiliza en una sentencia de asignación, es una variable global.

Para conocer mejor estas reglas, aquí tienes un programa de ejemplo. Escriba el siguiente código en el editor de archivos y guárdelo como *sameName3.py*:

```
def spam():
❶   huevos globales
     eggs= "spam" # this is the global def

  bacon():
```

```
❷   eggs= 'bacon' # this is a local def
    ham():
➲   print(huevos) # este es el global

    eggs= 42 # este es el spam() global
    print(huevos)
```

En la función spam(), eggs es la variable global eggs, porque hay una sentencia global para eggs al principio de la función ❶. En bacon(), eggs es una variable local, porque hay una sentencia de asignación para ella en esa función

❷. En ham() ❸, eggs es la variable global, porque no hay ninguna sentencia de asignación o sentencia global para ella en esa función. Si ejecutas *sameName3.py*, la salida tendrá este aspecto:

```
spam
```

En una función, una variable siempre será global o siempre será local. No hay forma de que el código de una función pueda usar una variable local llamada eggs y luego en esa misma función usar la variable global eggs.

NOTA

Si alguna vez desea modificar el valor almacenado en una variable global desde dentro de una función, debe utilizar una directiva
sobre esa .

Si intentas usar una variable local en una función antes de asignarle un valor, como en el siguiente programa, Python te dará un error. Para comprobarlo, escriba lo siguiente en el editor de archivos y guárdelo como *sameName4.py*:

```
def spam():
        print(huevos) # ¡ERROR!
❶       huevos= 'spam local'

❷ huevos= 'global'
   spam()
```

Si ejecuta el programa anterior, aparece un mensaje de error.

```
Traceback (llamada más reciente):
  File "C:/test3784.py", line 6, in <module> spam()
  File "C:/test3784.py", line 2, in spam print(huevos)
    # ¡ERROR!
UnboundLocalError: variable local 'eggs' referenciada antes de la asignación
```

Este error ocurre porque Python ve que hay una sentencia de asignación para eggs en la función spam() ❶ y por tanto considera que eggs es local. Pero como print(eggs) se ejecuta antes de asignar nada a eggs, la variable local eggs no existe. Python *no* volverá a utilizar la variable global eggs ❷.

FUNCIONES COMO "CAJAS NEGRAS"

A menudo, todo lo que necesita saber sobre una función son sus entradas (los parámetros) y su valor de salida; no siempre tiene que preocuparse de cómo funciona realmente el código de la función. Cuando piensas en las funciones de esta manera de alto nivel, es común decir que estás tratando la función como una "caja negra".

Esta idea es fundamental para la programación moderna. En capítulos posteriores de este libro te mostraremos varios módulos con funciones que fueron escritas por otras personas. Aunque puedes echar un vistazo al código fuente si tienes curiosidad, no necesitas saber cómo funcionan estas funciones para usarlas. Y como se recomienda escribir funciones sin variables globales, normalmente no tendrás que preocuparte de que el código de la función interactúe con el resto de tu programa.

Gestión de excepciones

Ahora mismo, obtener un error, o *excepción*, en tu programa Python significa que todo el programa se bloqueará. No quieres que esto suceda en los programas del mundo real. En lugar de eso, quieres que el programa detecte errores, los maneje y continúe ejecutándose.

Por ejemplo, considere el siguiente programa, que tiene un error de "división por cero". Abra una nueva ventana del editor de archivos e introduzca el siguiente código, guardándolo como *ceroDivide.py*:

```
def spam(dividePor):
    return 42 / dividePor

print(spam(2))
print(spam(12))
print(spam(0))
print(spam(1))
```

Hemos definido una función llamada spam, le hemos dado un parámetro y luego hemos impreso el valor de esa función con varios parámetros para ver qué pasa. Esta es la salida que se obtiene al ejecutar el código anterior:

```
21.0
3.5
Traceback (llamada más reciente):
  File "C:/zeroDivide.py", line 6, in <module>
    print(spam(0))
  File "C:/divideCero.py", line 2, in spam
    return 42 / dividePor
ZeroDivisionError: división por cero
```

Un ZeroDivisionError ocurre siempre que se intenta dividir un número por cero. Por el número de línea que aparece en el mensaje de error, se sabe que la sentencia return de spam() está causando un error.

Los errores pueden tratarse con sentencias try y except. El código que potencialmente podría tener un error se coloca en una cláusula try. Si se un error, la ejecución del programa se desplaza al inicio de la siguiente cláusula except.

Puedes poner el código anterior de dividir por cero en una cláusula try y tener un comando except contienen código para manejar lo que ocurre cuando se produce este error.

```
def spam(dividePor):
    try:
```

```
        return 42 / divideBy
    except ZeroDivisionError:
        print('Error: Argumento no válido.')

print(spam(2))
print(spam(12))
print(spam(0))
print(spam(1))
```

Cuando el código de una cláusula try provoca un error, la ejecución del programa pasa inmediatamente al código de la cláusula except. Después de ejecutar ese código, la ejecución continúa normalmente. La salida del programa anterior es la siguiente:

```
21.0
3.5
Error: Argumento no válido.
Ninguno
42.0
```

Tenga en cuenta que cualquier error que se produzca en las llamadas a funciones en un bloque try también se detectará. Considere el siguiente programa, que en su lugar tiene las llamadas a spam() en el bloque try:

```
def spam(dividePor):
    return 42 / dividePor

Inténtalo:
    print(spam(2))
    print(spam(12))
    print(spam(0))
    print(spam(1))
except ZeroDivisionError: print('Error:
    Argumento no válido.")
```

Cuando se ejecuta este programa, la salida tiene este aspecto:

```
21.0
3.5
Error: Argumento no válido.
```

La razón por la que print(spam(1)) nunca se ejecuta es porque una vez que la ejecución salta al código de la cláusula except, no vuelve a la cláusula try.
En lugar de eso, sigue bajando con normalidad.

Un programa corto: Adivina el número

Los ejemplos de juguete que te he mostrado hasta ahora son útiles para introducir conceptos básicos, pero ahora vamos a ver cómo todo lo que has aprendido se une en un programa más completo. En esta sección, te mostraré un simple juego de "adivinar el número". Cuando ejecutes este programa, la salida será algo parecido a esto:

```
Estoy pensando en un número entre 1 y 20.
Adivínalo.
10
Tu suposición es demasiado
baja. Adivina.
15
Tu suposición es demasiado
baja. Adivina.
17
Tu suposición es
demasiado alta. Adivina.
16
¡Buen trabajo! ¡Adivinaste mi número en 4 intentos!
```

Escriba el siguiente código fuente en el editor de archivos y guarde el archivo como *adivinaelnumero.py*:

```
# Esto es un juego de adivinar el número. import
random
númeroSecreto= random.randint(1, 20)
print('Estoy pensando en un número entre 1 y 20.')

# Pide al jugador que adivine 6 veces.
for adivinanzasTomadas in range(1, 7):
    print('Adivina.') adivina=
    int(input())

    if guess< secretNumber: print('Tu
        suposición es demasiado baja.')
    elif guess> secretNumber: print('Tu
        suposición es demasiado alta.')
    si no:
        romper    # ¡Esta condición  la correcta!

si adivina== secretNumber:
    print('¡Buen trabajo! Has adivinado mi número en '+ str(guessesTaken)+ ' guesses!')
else:
    print('No. El número en el que estaba pensando era '+ str(secretNumber))
```

Veamos este código línea por línea, empezando por arriba.

```
# Esto es un juego de adivinar el número. import
random
númeroSecreto= random.randint(1, 20)
```

Primero, un comentario en la parte superior del código explica lo que hace el programa. A continuación, el programa importa el módulo random para poder utilizar la función random.randint() para generar un número que el usuario debe adivinar. El valor de retorno, un número entero aleatorio entre 1 y 20, se almacena en la variable secretNumber.

```
print('Estoy pensando en un número entre 1 y 20.')

# Pide al jugador que adivine 6 veces.
for adivinanzasTomadas in range(1, 7):
    print('Adivina.') adivina=
    int(input())
```

El programa le dice al jugador que ha encontrado un número secreto y le da seis oportunidades para adivinarlo. El código que permite al jugador introducir un número y comprobarlo se encuentra en un for que se repite un máximo de seis veces. Lo primero que ocurre en el bucle es que el jugador teclea una suposición. Como input() devuelve una cadena, su valor de retorno se pasa directamente a int(), que traduce la cadena en un valor entero. Esto se almacena en una variable llamada guess.

```
if guess< secretNumber: print('Tu
    suposición es demasiado baja.')
elif guess> secretNumber: print('Tu
    suposición es demasiado alta.')
```

Estas pocas líneas de código comprueban si el número adivinado es mayor o menor que el número secreto. En cualquier caso, se imprime una pista en la pantalla.

```
si no:
    romper    # ¡Esta condición la correcta!
```

Si la suposición no es ni mayor ni menor que el número secreto, entonces debe ser igual al número secreto, en cuyo caso se desea que la ejecución del programa salga del bucle for.

```
si adivina== secretNumber:
    print('¡Buen trabajo! Has adivinado mi número en '+ str(guessesTaken)+ ' guesses!')
else:
    print('No. El número en el que estaba pensando era '+ str(secretNumber))
```

Tras el bucle for, la sentencia if...else anterior comprueba si el jugador

ha adivinado correctamente el número e imprime un mensaje apropiado en la pantalla. En ambos casos, el programa muestra una variable que contiene un valor entero (guessesTaken y secretNumber). Como debe concatenar estos valores enteros a cadenas, pasa estas variables a la función **str()**, que devuelve la forma de valor de cadena de estos enteros. Ahora estas cadenas concatenarse con los operadores+ antes de pasarlas finalmente a la llamada a la función **print()**.

Resumen

Las funciones son la principal forma de compartimentar el código en grupos lógicos. Dado que las variables de las funciones existen en sus propios ámbitos locales, el código de una función no puede afectar directamente a los valores de las variables de otras funciones. Esto limita qué código podría estar cambiando los valores de tus variables, lo que puede ser útil cuando se trata de depurar tu código.

Las funciones son una gran herramienta para ayudarte a organizar tu código. Puedes pensar en ellas como si fueran cajas negras: Tienen entradas en forma de parámetros y salidas en forma de valores de retorno, y el código que contienen no afecta a las variables de otras funciones.

En capítulos anteriores, un solo error podía hacer que tus programas se bloquearan. En este capítulo, has aprendido sobre las sentencias `try` y `except`, que pueden ejecutar código cuando se detecta un error. Esto puede hacer que tus programas sean más resistentes a casos de error comunes.

Preguntas prácticas

P: 1. ¿Por qué es ventajoso tener funciones en sus programas?

P: 2. ¿Cuándo se ejecuta el código de una función: cuando se define la función o cuando se llama a la función?

P: 3. ¿Qué enunciado crea una función?

P: 4. ¿Cuál es la diferencia entre una función y una llamada a función?

P: 5. ¿Cuántos ámbitos globales hay en un programa Python? ¿Cuántos ámbitos locales?

P: 6. ¿Qué ocurre con las variables de ámbito local cuando retorna la llamada a la función?

P: 7. ¿Qué es un valor de retorno? ¿Puede un valor de retorno formar parte de una expresión?

P: 8. Si una función no tiene una sentencia return, ¿cuál es el valor de retorno de una llamada a esa función?

P: 9. ¿Cómo se puede forzar una variable en una función para que haga referencia a la variable global?

P: 10. ¿Cuál es el tipo de datos de `None`?

P: 11. ¿Qué hace la declaración `import areallyourpetsnamederic`?

P: 12. Si tuvieras una función llamada `bacon()` en un módulo llamado `spam`, ¿cómo la llamarías después de importar `spam`?

P: 13. ¿Cómo se puede evitar que un programa se bloquee cuando recibe un error?

P: 14. ¿Qué va en la cláusula `try`? ¿Qué va en la cláusula `except`?

Proyectos prácticos

Para practicar, escribe programas que realicen las siguientes tareas.

La secuencia Collatz

Escribe una función llamada collatz() que tenga un parámetro llamado número. Si number es par, collatz() debe imprimir number // 2 y devolver este . Si el número es impar, entonces collatz() debe imprimir y devolver 3 * número+ 1.

A continuación, escriba un programa que permita al usuario teclear un número entero y que siga llamando a collatz() sobre ese número hasta que la función devuelva el valor 1. (Aunque parezca increíble, esta secuencia funciona para cualquier número entero: tarde o temprano, utilizando esta secuencia, llegará a 1.) Ni siquiera los matemáticos saben por qué. Tu
programa está explorando lo que se denomina la *secuencia de Collatz*, a veces llamada "el problema matemático imposible más simple").

Recuerde convertir el valor de retorno de input() a un entero con la función int(); de lo contrario, será un valor de cadena.

Pista: Un número entero es par si number % 2== 0, y es impar si number % 2== 1.

La salida de este programa podría ser algo parecido a esto:

```
Introduzca el
número: 3
10
5
16
8
4
2
1
```

Validación de entradas

Añada sentencias try y except al proyecto anterior para detectar si el usuario teclea una cadena no entera. Normalmente, la función int() generará un error ValueError si se le pasa una cadena no entera, como en int('cachorro'). En la cláusula except, imprima un mensaje al usuario diciendo que debe introducir un entero.

Capítulo 4. Listas Listas

Otro tema que tendrás que entender antes de empezar a escribir programas en serio es el tipo de datos lista y su primo, la tupla. Las listas y tuplas pueden contener múltiples valores, lo que facilita la escritura de programas que manejen grandes cantidades de datos. Y como las propias listas pueden contener otras listas, puedes utilizarlas para organizar datos en estructuras jerárquicas.

En este capítulo, discutiré los fundamentos de las listas. También te enseñaré sobre métodos, que son funciones que están ligadas a valores de un cierto tipo de datos. A continuación, trataré brevemente los tipos de datos de lista tuple y string y cómo se comparan con los valores de lista. En el próximo capítulote presentaré el tipo de datos diccionario.

El tipo de datos List

Una *lista* es un valor que contiene varios valores en una secuencia ordenada. El término *valor* de *lista* se refiere a la lista en sí (que es un valor que puede almacenarse en una variable o pasarse a una función como cualquier otro valor), no a los valores dentro del valor de lista. Un valor de lista tiene este aspecto **['gato', 'murciélago', 'rata', 'elefante']**. Del mismo modo que los valores de cadena se escriben con comillas para marcar el inicio de la cadena y el final.
una lista comienza con un corchete de apertura y termina con un corchete de cierre, **[]**. Los valores de la lista también se denominan *elementos*. Los elementos se separan con comas (es decir, *están delimitados por comas*). Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> [1, 2, 3]
[1, 2, 3]
>>> ['gato', 'murciélago', 'rata', 'elefante']
['gato', 'murciélago', 'rata', 'elefante']
>>> ['hola', 3.1415, Verdadero, Ninguno, 42]
['hola', 3.1415, Verdadero, Ninguno, 42]
```

```
>>> ❶spam= ['gato', 'murciélago', 'rata', 'elefante']
>>> spam
['gato', 'murciélago', 'rata', 'elefante']
```

A la variable spam ❶ se le sigue asignando un único valor: el valor de lista. Pero el propio valor de lista contiene otros valores. El valor **[]** es una lista vacía que no contiene valores, similar a **''**, la cadena vacía.

Obtención de valores individuales en una lista con índices

Digamos que tienes la lista ['gato', 'murciélago', 'rata', 'elefante'] almacenada en una variable llamada spam. El código de Python spam[0] se evaluaría como 'gato', y spam[1] se evaluaría como 'murciélago', y así sucesivamente. El número entero dentro de los corchetes que sigue a la lista se llama *índice*. El primer valor de la lista está en el índice 0, el segundo valor está en el índice 1, el tercer valor está en el índice 2, y así sucesivamente.

La Figura 4-1 muestra un valor de lista asignado a spam, junto con el valor que evaluarían las expresiones de índice.

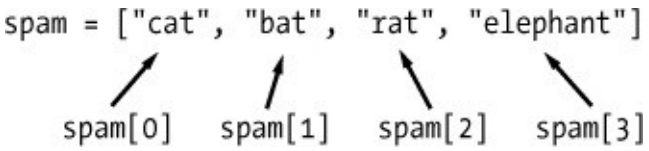


Figura 4-1. Un valor de lista almacenado en la variable spam*, mostrando a qué valor se refiere cada índice*

Por ejemplo, escriba las siguientes expresiones en el intérprete de comandos interactivo. Comience asignando una lista a la variable spam.

```
>>> spam= ['gato', 'murciélago', 'rata', 'elefante']
    >>> spam[0]
    gato
    >>> spam[1]
    murciélago
    >>> spam[2]
    "rata
    >>> spam[3]
    elefante
    >>> ['gato', 'murciélago', 'rata', 'elefante'][3]
    elefante

>>> ❶'Hola '+ spam[0]
❷ 'Hola gato'
    >>> 'El '+ spam[1]+ ' se comió al '+ spam[0]+ '.'
    'El murciélago se comió al gato'.
```

Observa que la expresión 'Hola '+ spam[0] ❶ se evalúa como 'Hola '+ 'gato' porque spam[0] se evalúa como la cadena 'gato'. Esta expresión a su vez se evalúa al valor de cadena 'Hola gato'❷.

Python te dará un mensaje de error IndexError si utilizas un índice que excede el número de valores de tu lista.

```
>>> spam= ['gato', 'murciélago', 'rata', 'elefante']
```

```
>>> spam[10000]
Traceback (llamada más reciente):
  File "<pyshell#9>", line 1, in <module> spam[10000]
IndexError: índice de lista fuera de rango
```

Los índices sólo pueden ser valores enteros, no flotantes. El siguiente ejemplo provocará un error TypeError:

```
>>> spam= ['gato', 'murciélago', 'rata', 'elefante']
>>> spam[1]
murciélago
>>> spam[1.0]
Traceback (llamada más reciente):
  File "<pyshell#13>", line 1, in <module> spam[1.0]
TypeError: los índices de la lista deben ser enteros, no flotantes
>>> spam[int(1.0)]
murciélago
```

Las listas también pueden contener otros valores de listas. Se puede acceder a los valores de estas listas de listas utilizando índices múltiples, de esta forma:

```
>>> spam= [['gato', 'murciélago'], [10, 20, 30, 40, 50]]
>>> spam[0]
['gato', 'murciélago']
>>> spam[0][1]
murciélago
>>> spam[1][4]
50
```

El primer índice indica el valor de la lista que se va a utilizar, y el segundo indica el valor dentro del valor de la lista. Por ejemplo, spam[0][1] imprime 'bat', el segundo valor de la primera lista. Si sólo utiliza un índice, el programa imprimirá el valor completo de la lista en ese índice.

Índices negativos

Aunque los índices empiezan en 0 y van subiendo, también puedes utilizar enteros negativos para el índice. El valor entero -1 se refiere al último índice de una lista, el valor -2 se refiere al penúltimo índice de una lista, y así sucesivamente. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= ['gato', 'murciélago', 'rata', 'elefante']
>>> spam[-1]
elefante
>>> spam[-3]
murciélago
>>> 'El '+ spam[-1]+ ' tiene miedo del '+ spam[-3]+ '.'
'El elefante tiene miedo del murciélago'.
```

Obtención de sublistas con Slices

Del mismo modo que un índice puede obtener un único valor de una lista, un *corte* puede obtener varios valores de una lista, en forma de una nueva lista. Un slice se escribe entre corchetes, como un índice, pero tiene dos enteros separados por dos puntos. Fíjate en la diferencia entre índices y trozos.

- spam[2] es una lista con un índice (un entero).
- spam[1:4] es una lista con un slice (dos enteros).

En un corte, el primer número entero es el índice donde comienza el corte. El segundo entero es el índice en el que termina el corte. Un corte llega hasta el valor del segundo índice, pero no lo incluye. Un corte se evalúa como un nuevo valor de lista. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= ['gato', 'murciélago', 'rata', 'elefante']
>>> spam[0:4]
['gato', 'murciélago', 'rata', 'elefante']
>>> spam[1:3]
['murciélago', 'rata']
>>> spam[0:-1]
['gato', 'murciélago', 'rata']
```

Como método abreviado, puede omitir uno o ambos índices a ambos lados de los dos puntos de la porción. Omitir el primer índice equivale a utilizar 0, o el principio de la lista. Omitir el segundo índice equivale a utilizar la longitud de la lista, que se cortará hasta el final de la lista. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= ['gato', 'murciélago', 'rata', 'elefante']
>>> spam[:2]
['gato', 'murciélago']
>>> spam[1:]
['murciélago', 'rata', 'elefante']
>>> spam[:]
['gato', 'murciélago', 'rata', 'elefante']
```

Obtener la longitud de una lista con len()

La función `len()` devolverá el número de valores que hay en un valor de lista que se le haya pasado, del mismo modo que puede contar el número de caracteres de un valor de cadena. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= ['gato', 'perro', 'ALCE']
>>> len(spam)
3
```

Modificación de valores en una lista con índices

Normalmente un nombre de variable va en el lado izquierdo de una sentencia de asignación, como spam= 42. Sin embargo, también puedes usar un índice de una lista para cambiar el valor en ese índice. Por ejemplo, spam[1]= 'aardvark' significa "Asignar el valor en el índice 1 de la lista spam a la cadena 'aardvark'". Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= ['gato', 'murciélago', 'rata', 'elefante']
>>> spam[1]= 'aardvark'
>>> spam
['gato', 'oso hormiguero', 'rata', 'elefante']
>>> spam[2]= spam[1]
>>> spam
['gato', 'oso hormiguero', 'oso hormiguero', 'elefante']
>>> spam[-1]= 12345
>>> spam
['gato', 'oso hormiguero', 'oso hormiguero', 12345]
```

Concatenación de listas y replicación de listas

El operador+ puede combinar dos listas para crear un nuevo valor de lista del mismo modo que combina dos cadenas para crear un nuevo valor de cadena. El operador * también puede utilizarse con una lista y un valor entero para replicar la lista. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> [1, 2, 3]+ ['A', 'B', 'C']
[1, 2, 3, 'A', 'B', 'C']
>>> ['X', 'Y', 'Z'] * 3
['X', 'Y', 'Z', 'X', 'Y', 'Z', 'X', 'Y', 'Z']
>>> spam= [1, 2, 3]
>>> spam= spam+ ['A', 'B', 'C']
>>> spam
[1, 2, 3, 'A', 'B', 'C']
```

Eliminar valores de listas con sentencias del

La sentencia del borrará valores en un índice de una lista. Todos los valores de la lista posteriores al valor eliminado se desplazarán un índice hacia arriba. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= ['gato', 'murciélago', 'rata', 'elefante']
>>> del spam[2]
>>> spam
['gato', 'murciélago', 'elefante']
>>> del spam[2]
>>> spam
['gato', 'murciélago']
```

La sentencia del también se puede utilizar en una variable simple para borrarla, como si fuera una sentencia "unassignment". Si intenta utilizar la variable después de borrarla, obtendrá un error NameError porque la variable ya no existe.

En la práctica, casi nunca es necesario borrar variables simples. La sentencia del se utiliza sobre todo para borrar valores de listas.

Trabajar con listas

Cuando empiezas a escribir programas, es tentador crear muchas variables individuales para almacenar un grupo de valores similares. Por ejemplo, si quisiera almacenar los nombres de mis gatos, podría tener la tentación de escribir código como éste:

```
catName1= 'Zophie'
catName2= 'Pooka'
catName3=  "Simon"
catName4= 'Lady Macbeth"
catName5= 'Fat-tail'
catName6=  "Miss Cleo'
```

(En realidad no tengo tantos gatos, lo juro.) Resulta que es una mala forma de escribir código. Por un lado, si el número de gatos cambia, tu programa nunca será capaz de almacenar más gatos de los que tienes en variables. Este tipo de programas también tienen mucho código duplicado o casi idéntico. Considere cuánto código duplicado hay en el siguiente programa, que debería introducir en el editor de archivos y guardar como *todosMisGatos1.py*:

```
print('Introduzca el nombre del gato 1:')
catName1= input()
print('Introduzca el nombre del gato 2:')
catName2= input()
print('Introduzca el nombre del gato 3:')
catName3= input()
print('Introduzca el nombre del gato 4:')
catName4= input()
print('Introduzca el nombre del gato 5:')
catName5= input()
print('Introduzca el nombre del gato 6:')
catName6= input()
print('Los nombres de los gatos son:')
print(catName1+ ' '+ catName2+ ' '+ catName3+ ' '+ catName4+ ' ' + catName5+ ' '+
catName6)
```

En lugar de utilizar múltiples variables repetitivas, puede utilizar una única variable que contenga un valor de lista. Por ejemplo, aquí hay una versión nueva y mejorada del programa *allMyCats1.py*. Esta nueva versión utiliza una única lista y puede almacenar cualquier número de gatos que introduzca el usuario. En una nueva ventana del editor de archivos, escriba el siguiente código fuente y guárdelo como *todosMisGatos2.py*:

```
catNames= []
while True:
    print('Introduzca el nombre del gato '+ str(len(catNames)+ 1+
```

```
        (O no introduzca nada para
    parar.):') nombre= input()
    si nombre== '':
        break
    catNames= catNames+ [nombre] # concatenación de listas
print('Los nombres de los gatos son:')
para nombre en
    catNames: print('
            '+ nombre)
```

Cuando ejecutes este programa, la salida será algo parecido a esto:

```
Introduzca el nombre del gato 1 (O no introduzca nada para parar):
Zophie
Introduzca el nombre del gato 2 (O no introduzca nada para parar):
Pooka
Introduzca el nombre del gato 3 (O no introduzca nada para parar):
Simon
Introduzca el nombre del gato 4 (O no introduzca nada para parar):
Lady Macbeth
Introduzca el nombre de la cat. 5 (O no introduzca nada para parar):
Cola gorda
Introduzca el nombre de la cat. 6 (O no introduzca nada para parar):
SEÑORITA Cleo
Introduzca el nombre del gato 7 (O no introduzca nada para parar):

Los nombres de los
  gatos son: Zophie
  Pooka
  Simon
  Lady Macbeth
  Cola
  Gorda
  Miss Cleo
```

La ventaja de utilizar una lista es que los datos están ahora en una estructura, por lo que el programa es mucho más flexible en el procesamiento de los datos de lo que sería con varias variables repetitivas.

Uso de bucles for con listas

En el Capítulo 2, aprendiste sobre el uso de bucles for para ejecutar un bloque de código un cierto número de veces. Técnicamente, un bucle `for` repite el bloque de código una vez por cada valor de una lista o valor similar a una lista. Por ejemplo, si ejecutas este código

```
para i en rango(4):
    print(i)
```

la salida de este programa sería la siguiente:

```
0
1
2
3
```

Esto se debe a que el valor devuelto por `range(4)` es un valor similar a una lista que Python considera similar a `[0, 1, 2, 3]`. El siguiente programa tiene la misma salida que el anterior:

```
para i en [0, 1, 2,
    3]: print(i)
```

Lo que en realidad hace el bucle `for` anterior es recorrer su cláusula con la variable `i` fijada a un valor sucesivo de la lista `[0, 1, 2, 3]` en cada iteración.

NOTA

En este libro, utilizo el término tipo lista *para referirme a los tipos de datos que técnicamente se denominan* secuencias. *Sin embargo, no es necesario que conozcas las definiciones técnicas de este término.*

Una técnica habitual en Python es utilizar `range(len(`*`algunaLista`*`))` con un bucle `for` para iterar sobre los índices de una lista. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> SUMINISTROS= ['BOLÍGRAFOS', 'GRAPADORAS', LANZALLAMAS', 'CARPETAS']
>>> para i en rango(len(SUMINISTROS)):
    print('Índice '+ str(i)+ ' en SUMINISTROS ES: '+ SUMINISTROS[i])

El índice 0 en suministros es:
bolígrafos El índice 1 en suministros
es: grapadoras
El índice 2 de suministros es:
lanzallamas El índice 3 de suministros
es: carpetas
```

El uso de `range(len(supplies))` en el bucle `for` mostrado anteriormente es práctico

porque el código del bucle puede acceder al índice (como la variable i) y al valor de ese índice (como suministros[i]). Lo mejor de todo es que range(len(suministros)) iterará a través de todos los índices de suministros, sin importar cuántos elementos contenga.

Los operadores in y not in

Puede determinar si un valor está o no en una lista con los operadores `in` y `not in`. Al igual que otros operadores, `in` y `not in` se utilizan en expresiones y conectan dos valores: un valor a buscar en una lista y la lista donde puede encontrarse. Estas expresiones se evaluarán a un valor booleano. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> 'howdy' en ['hello', 'hi', 'howdy', 'HEYAS']
Verdadero
>>> spam= ['hello', 'hi', 'howdy', 'HEYAS']
>>> gato" en spam
Falso
>>> howdy' no en spam
Falso
>>> 'gato' no en spam
Verdadero
```

Por ejemplo, el siguiente programa permite al usuario escribir el nombre de una mascota y luego comprueba si el nombre está en una lista de mascotas. Abra una nueva ventana del editor de archivos, introduzca el siguiente código y guárdelo como *misMascotas.py*:

```
myPets= ['Zophie', 'Pooka', 'Fat-tail']
print('Introduce el nombre de tu mascota:')
nombre= entrada()
si nombre no está en misMascotas:
    print('No tengo una mascota llamada '+ name)
else:
    print(nombre+ ' es mi mascota.')
```

El resultado puede ser algo parecido a esto:

```
Introduzca un nombre de mascota:
Pie
No tengo una mascota llamada Pie
```

El truco de la asignación múltiple

El *truco de la asignación múltiple* es un atajo que te permite asignar múltiples variables con los valores de una lista en una sola línea de código. Así que en lugar de hacer esto

```
>>> cat= ['gordo', 'negro', 'ruidoso']
>>> tamaño= cat[0]
>>> color= cat[1]
>>> disposición= cat[2]
```

podrías escribir esta línea de :

```
>>> cat= ['gordo', 'negro', 'ruidoso']
>>> tamaño, color, disposición= cat
```

El número de variables y la longitud de la lista deben ser exactamente iguales, o Python te dará un ValueError:

```
>>> cat= ['gordo', 'negro', 'ruidoso']
>>> tamaño, color, DISPOSICIÓN, nombre= cat
Traceback (llamada más reciente):
  File "<pyshell#84>", line 1, in <module> size,
    color, disposition, name= cat
ValueError: necesito más de 3 valores para desempaquetar
```

Operadores de asignación aumentados

Al asignar un valor a una variable, con frecuencia utilizará la propia variable. Por ejemplo, después de asignar 42 a la variable spam, aumentaría el valor de spam en 1 con el siguiente código:

```
>>> spam= 42
>>> spam= spam+ 1
>>> spam
43
```

Como atajo, puede utilizar el operador de asignación aumentada**+=** para hacer lo mismo:

```
>>> spam= 42
>>> spam+= 1
>>> spam
43
```

Existen operadores de asignación aumentados para los operadores**+** , -, *, /, y %, descritos en la Tabla 4-1.

Tabla 4-1. Operadores de asignación aumentados

Declaración de asignación aumentada	Declaración de asignación equivalente
spam+= 1	spam= spam+ 1
spam -= 1	spam= spam - 1
spam *= 1	spam= spam * 1
spam /= 1	spam= spam / 1
spam %= 1	spam= spam % 1

El operador**+=** también puede concatenar cadenas y listas, y el operador ***=** puede replicar cadenas y listas. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= 'Hola'
>>> spam+= ' ¡mundo!
>>> spam
"¡Hola mundo!

>>> bacon= ['Zophie']
>>> bacon *= 3
```

```
>>> bacon
['Zophie', 'Zophie', 'Zophie']
```

Métodos

Un *método* es lo mismo que una función, salvo que se "llama" sobre un valor. Por ejemplo, si el valor de una lista estuviera almacenado en `spam`, llamarías al método de lista `index()` (que explicaré a continuación) en esa lista así: `spam.index("hola")`. La parte del método viene después del , separada por un punto.

Cada tipo de datos tiene su propio conjunto de métodos. El tipo de datos lista, por ejemplo, tiene varios métodos útiles para encontrar, añadir, eliminar y manipular valores de una lista.

Encontrar un valor en una lista con el método index()

Las listas de valores tienen un método index() al que se le puede pasar un valor, y si ese valor existe en la lista, se devuelve el índice del valor. Si el valor no está en la lista, entonces Python produce un error ValueError. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= ['hello', 'hi', 'howdy', 'HEYAS']
>>> spam.index('hola')
0
>>> spam.index('HEYAS')
3
>>> spam.index('hola hola hola')
Traceback (llamada más reciente):
  File "<pyshell#31>", line 1, in <module>
    spam.index('howdy howdy howdy')
ValueError: 'howdy howdy howdy' no está en la lista
```

Cuando hay duplicados del valor en la lista, se devuelve el índice de su primera aparición. Introduzca lo siguiente en el intérprete de comandos interactivo y observe que index() devuelve 1, no 3:

```
>>> spam= ['Zophie', 'Pooka', 'Fat-tail', 'Pooka']
>>> spam.index('Pooka')
1
```

Añadir valores a listas con los métodos append() e insert()

Para añadir nuevos valores a una lista, utilice los métodos append() e insert(). Introduzca lo siguiente en el shell interactivo para llamar al método append() sobre un valor de lista almacenado en la variable spam:

```
>>> spam= ['gato', 'perro', 'murciélago']
>>> spam.append('ALCE')
>>> spam
['gato', 'perro', 'murciélago', 'alce']
```

La llamada anterior al método append() añade el argumento al final de la lista. El método insert() puede insertar un valor en cualquier índice de la lista. El primer argumento de insert() es el índice para el nuevo valor, y el segundo argumento es el nuevo valor a insertar. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= ['gato', 'perro', 'murciélago']
>>> spam.insert(1, 'pollo')
>>> spam
['gato', 'pollo', 'perro', 'murciélago']
```

Observe que el código es spam.append("alce") y spam.insert(1, 'pollo'), no spam= spam.append("alce") y spam= spam.insert(1, 'pollo'). Ni append() ni insert() devuelven el nuevo valor de spam. (De hecho, el valor de retorno de append() e
insert() es None, por lo que definitivamente no querrá almacenar esto como el nuevo valor de la variable). En su *lugar*, la lista se modifica. La modificación de una lista en su lugar se trata con más detalle más adelante en Tipos de datos mutables e inmutables.

Los métodos pertenecen a un único tipo de datos. Los métodos append() e insert() son métodos de lista y sólo pueden invocarse sobre valores de lista, no sobre otros valores cadenas o enteros. Introduzca lo siguiente en el intérprete de comandos interactivo y observe los mensajes de error AttributeError que aparecen:

```
>>> HUEVOS= 'hola'
>>> EGGS.append('mundo')
Traceback (llamada más reciente):
  File "<pyshell#19>", line 1, in <module>
    eggs.append('mundo')
AttributeError: 'str' object has no attribute 'append'
>>> bacon= 42
```

```
>>> bacon.insert(1, 'mundo')
Traceback (llamada más reciente):
  File "<pyshell#22>", line 1, in <module>
    bacon.insert(1, 'world')
AttributeError: 'int' object has no attribute 'insert'
```

Eliminar valores de las listas con remove()

Al método remove() se le pasa el valor a eliminar de la lista sobre la que se llama. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= ['gato', 'murciélago', 'rata', 'elefante']
>>> spam.eliminar('bat')
>>> spam
['gato', 'rata', 'elefante']
```

Si se intenta eliminar un valor que no existe en la lista, se producirá un error ValueError. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo y observe el error que se muestra:

```
>>> spam= ['gato', 'murciélago', 'rata', 'elefante']
>>> spam.eliminar('pollo')
Traceback (llamada más reciente):
  File "<pyshell#11>", line 1, in <module>
    spam.remove('pollo')
ValueError: list.remove(x): x no está en la lista
```

Si el valor aparece varias veces en la lista, sólo se eliminará la primera instancia del valor. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= ['gato', murciélago', 'rata', 'gato', 'sombrero', 'gato']
>>> spam.eliminar('gato')
>>> spam
['murciélago', 'rata', 'gato', 'sombrero', 'gato']
```

La sentencia del es buena cuando se conoce el índice del valor se quiere eliminar de la lista. El método remove() es bueno cuando conoces el valor que quieres eliminar de la lista.

Ordenación de los valores de una lista con el método sort()

Las listas de valores numéricos o las listas de cadenas pueden ordenarse con el método sort(). Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= [2, 5, 3.14, 1, -7]
>>> spam.ordenar()
>>> spam
[-7, 1, 2, 3.14, 5]
>>> spam= ['HORMIGAS', 'GATOS', 'PERROS', 'TEJONES', 'ELEFANTES']
>>> spam.ordenar()
>>> spam
['hormigas', 'tejones', 'gatos', 'perros', 'elefantes']
```

También puede pasar True como argumento de la palabra clave reverse para que sort() ordene los valores en orden inverso. Introduzca lo siguiente en el shell interactivo:

```
>>> spam.sort(reverse=True)
>>> spam
['elefantes', perros', 'gatos', 'tejones', 'hormigas']
```

Hay tres cosas que debes tener en cuenta sobre el sort(). En primer lugar, el método sort() ordena la lista en su lugar; no intentes capturar el valor de retorno escribiendo código como spam= spam.sort().

En segundo lugar, no puedes ordenar listas que contengan tanto valores numéricos *como* valores de cadena, ya que Python no sabe cómo comparar estos valores. Escriba lo siguiente en el intérprete de comandos interactivo y observe el error TypeError:

```
>>> spam= [1, 3, 2, 4, 'Alice', 'Bob']
>>> spam.ordenar()
Traceback (llamada más reciente):
  File "<pyshell#70>", line 1, in <module> spam.sort()
TypeError: tipos no ordenables: str()< int()
```

En tercer lugar, sort() utiliza el "orden ASCIIbético" en lugar del orden alfabético real para ordenar las cadenas. Esto significa que las mayúsculas van antes que las minúsculas. Por lo tanto, la *a* minúscula se ordena de modo que venga *después* de la *Z* mayúscula. Para ver un ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= ['Alice', 'HORMIGAS', 'Bob', 'TEJONES', 'Carol', 'GATOS']
>>> spam.ordenar()
>>> spam
```

```
['Alice', 'Bob', 'Carol', 'hormigas', 'tejones', 'gatos']
```

Si necesita ordenar los valores en orden alfabético regular, pase str. lower como argumento de palabra clave en la llamada al método sort().

```
>>> spam= ['a', 'z', 'A', 'Z']
>>> spam.sort(KEY=STR.lower)
>>> spam
['a', 'A', 'z', 'Z']
```

Esto hace que la función sort() trate todos los elementos de la lista como si estuvieran en minúsculas sin cambiar realmente los valores de la lista.

Programa de ejemplo: Bola 8 mágica con una lista

Utilizando listas, puede escribir una versión mucho más elegante del programa Bola 8 Mágica del capítulo anterior. En lugar de varias líneas de sentencias `elif` casi idénticas, puede crear una única lista con la que funcione el código. Abra una nueva ventana del editor de archivos e introduzca el siguiente código. Guárdalo como *magic8Ball2.py*.

```
importar aleatorio

mensajes= ['Es cierto', 'Es
    decididamente así',
    'Sí definitivamente',
    'Respuesta confusa
    inténtalo de nuevo',
    'Pregunta de nuevo
    más tarde',
    'Concéntrate y vuelve a
    preguntar', "Mi respuesta es
    no',
    Perspectivas no muy ", "Muy
    dudoso"].

print(mensajes[random.randint(0,    len(mensajes) -      1)])
```

EXCEPCIONES A LAS REGLAS DE INDENTACIÓN EN PYTHON

En la mayoría de los casos, la cantidad de sangría de una línea de código indica a Python en qué bloque se encuentra. Sin embargo, hay algunas excepciones a esta regla. Por ejemplo, las listas pueden abarcar varias líneas en el archivo de código fuente. La sangría de estas líneas no importa; Python sabe que hasta que no ve el corchete final, la lista no ha terminado. Por ejemplo, puedes tener un código como este:

```
spam=
    ['manzanas',
    'naranjas',
                    "plátanos",
gatos']
print(spam)
```

Por supuesto, en la práctica, la mayoría de la gente utiliza el comportamiento de Python para que sus listas tengan un aspecto bonito y legible, como la lista de mensajes del programa Magic 8 Ball.

También puede dividir una única instrucción en varias líneas utilizando el carácter de continuación de línea `\` al final. Piense que `\` dice: "Esta instrucción continúa en la línea siguiente". La `sangría` en la línea después de una continuación de línea no es significativa. Por ejemplo, lo siguiente es código Python válido:

```
print('Hace cuatro veintenas y siete '+ \
      'años...')
```

Estos trucos son útiles cuando quieres reorganizar largas líneas de código Python para que sean un poco más legibles.

Cuando ejecutes este programa, verás que funciona igual que el anterior programa *magic8Ball.py*.

Fíjate en la expresión que utiliza como índice en los mensajes: random.randint(0, len(mensajes) - 1). Esto produce un número aleatorio para usar como índice, independientemente del tamaño de los mensajes. Es decir, obtendrás un número aleatorio entre 0 y el valor de len(mensajes) - 1. La ventaja de este método es que puede añadir y eliminar fácilmente cadenas de la lista de mensajes sin cambiar otras líneas de código. Si más adelante actualizas el código, habrá menos líneas que tengas que cambiar y menos posibilidades de que introduzcas errores.

Tipos de lista: Cadenas y tuplas

Las listas no son los únicos tipos de datos que representan secuencias ordenadas de valores. Por ejemplo, las cadenas y las listas son similares, si se considera que una cadena es una "lista" de caracteres de texto individuales. Muchas de las cosas que se pueden hacer con las listas también se pueden hacer con las cadenas: indexarlas, trocearlas y utilizarlas con bucles `for`, con `len()` y con los operadores `in` y `not in`. Para , introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> nombre= 'Zophie'
>>> nombre[0]
'Z'
>>> nombre[-2]
'i'
>>> nombre[0:4]
Zoph
>>> Zo" en el nombre
Verdadero
>>> z' en el nombre
Falso
>>> 'p' no está en el nombre
Falso
>>> para i en nombre:
        print('  * '+ i+ '       *')

  * Z     *
  * o     *
  * p     *
  * h     *
  * i     *
  * e     *
```

Tipos de datos mutables e inmutables

Pero las listas y las cadenas son diferentes en un aspecto importante. Un valor de lista es un tipo de dato *mutable*: Se le pueden añadir, eliminar o cambiar valores. Sin embargo, una cadena es *inmutable*: No puede modificarse. Intentar reasignar un único carácter en una cadena da como resultado un error TypeError, como se puede ver introduciendo lo siguiente en el intérprete de comandos interactivo:

```
>>> nombre= 'Zophie un gato'
>>> nombre[7]= 'el'
Traceback (llamada más reciente):
  File "<pyshell#50>", line 1, in <module> name[7]=
    'the'
TypeError: El objeto 'str' no admite la asignación de elementos
```

La forma correcta de "mutar" una cadena es utilizar el corte y la concatenación para construir una *nueva* cadena copiando partes de la cadena anterior. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> nombre= 'Zophie un gato'
>>> newName= nombre[0:7]+ 'el'+ nombre[8:12]
>>> nombre
Zophie un gato
>>> nuevoNombre
El gato Zophie
```

Utilizamos `[0:7]` y `[8:12]` para referirnos a los caracteres que no deseamos sustituir. Observa que la cadena original `'Zophie un gato'` no se modifica porque las cadenas son inmutables.

Aunque un valor de lista *es* mutable, la segunda línea del código siguiente no modifica los huevos de lista:

```
>>> HUEVOS= [1, 2, 3]
>>> HUEVOS= [4, 5, 6]
>>> HUEVOS
[4, 5, 6]
```

El valor de lista de los huevos no se modifica aquí, sino que un valor de lista completamente nuevo y diferente (`[4, 5, 6]`) sobrescribe el valor de lista antiguo (`[1, 2, 3]`). Esto se muestra en la Figura 4-2.

Si quisieras modificar realmente la lista original en huevos para que contuviera `[4, 5, 6]`, tendrías que hacer algo así:

```
>>> HUEVOS= [1, 2, 3]
```

```
>>> del huevos[2]
>>> del huevos[1]
>>> del huevos[0]
>>> huevos.append(4)
>>> huevos.append(5)
>>> huevos.append(6)
>>> huevos
[4, 5, 6]
```

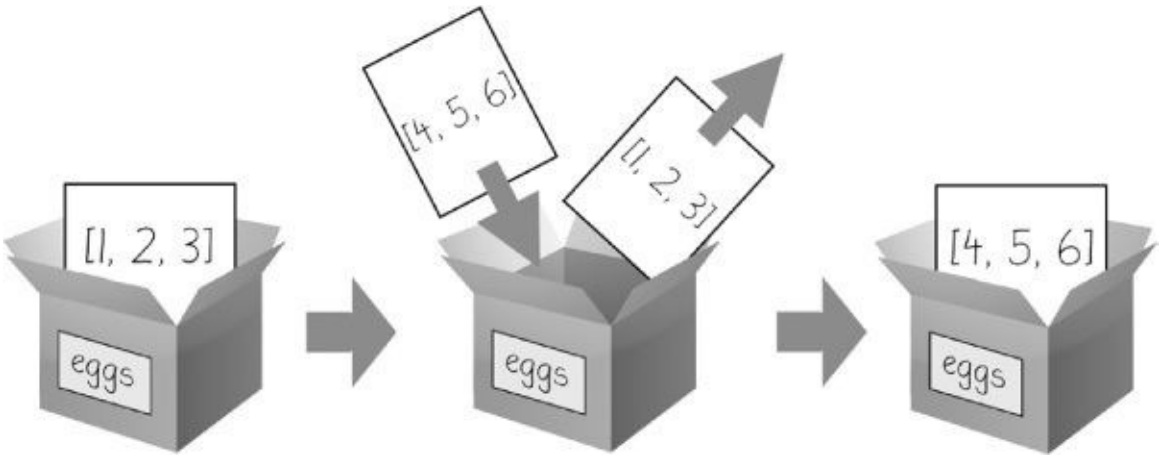


Figura 4-2. Cuando se ejecuta eggs= [4, 5, 6]*, el contenido de* eggs *se sustituye por un nuevo valor de la lista.*

En el primer ejemplo, el valor de lista con el que acaban los huevos es el mismo valor de lista con el que empezó. Sólo que esta lista se ha modificado, en lugar de sobrescribirse. La Figura 4-3 muestra los siete cambios realizados por las siete primeras líneas del ejemplo de shell interactivo anterior.

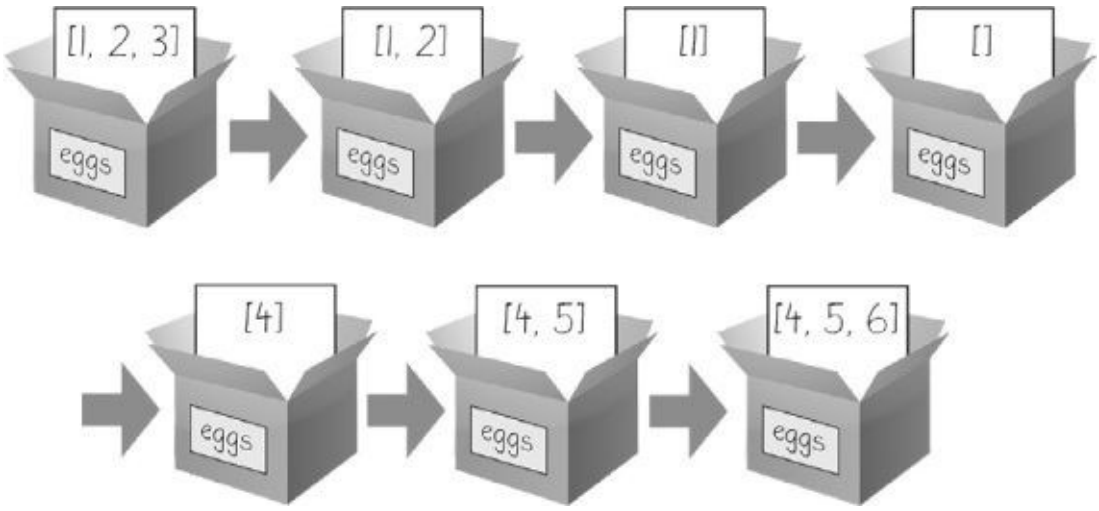


Figura 4-3. La sentencia del *y el método* append() *modifican el mismo valor de lista en su lugar.*

Cambiar un valor de un tipo de dato mutable (como lo que hacen la sentencia `del` y el método `append()` en el ejemplo anterior) cambia el valor en su lugar, ya que el valor de la variable no es reemplazado por un nuevo valor de la lista.

Los tipos mutables frente a los inmutables pueden parecer una distinción sin sentido, pero Pasando Referencias explicará el diferente comportamiento al llamar a funciones con argumentos mutables frente a argumentos inmutables. Pero primero, conozcamos el tipo de datos tupla, que es una forma inmutable del tipo de datos lista.

El tipo de datos tupla

El tipo de datos *tupla* es casi idéntico al tipo de datos lista, excepto en dos aspectos. En primer lugar, las tuplas se escriben con paréntesis, **(** y **)**, en lugar de corchetes, **[** y **]**. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> HUEVOS= ('hola', 42, 0.5)
>>> HUEVOS[0]
Hola.
>>> HUEVOS[1:3]
(42, 0.5)
>>> len(HUEVOS)
3
```

Pero la principal diferencia entre las tuplas y las listas es que las tuplas, como las cadenas, son inmutables. Los valores de las tuplas no pueden ser modificados, añadidos o eliminados. Introduzca lo siguiente en el intérprete de comandos interactivo y observe el mensaje de error TypeError:

```
>>> HUEVOS= ('hola', 42, 0.5)
>>> HUEVOS[1]= 99
Traceback (llamada más reciente):
  File "<pyshell#5>", line 1, in <module>
    eggs[1] = 99
TypeError: El objeto 'tuple' no admite la asignación de elementos
```

Si sólo tienes un valor en tu tupla, puedes indicarlo colocando una coma después del valor dentro del paréntesis. De lo contrario, Python pensará que acabas de escribir un valor dentro de un paréntesis normal. La coma es lo que permite a Python saber que se trata de un valor de tupla. (A diferencia de otros lenguajes de programación, en Python está bien poner una coma después del último elemento de una lista).

o tupla). Introduzca las siguientes llamadas a la función type() en el shell interactivo para ver la distinción:

```
>>> type(('hola',))
<clase 'tuple'>
>>> type(('hola'))
<clase 'str'>
```

Puedes utilizar tuplas para transmitir a cualquiera que lea tu código que no pretendes que esa secuencia de valores cambie. Si necesitas una secuencia ordenada de valores que nunca cambie, utiliza una tupla. Una segunda ventaja de usar tuplas en lugar de listas es que, como son inmutables y su contenido no cambia, Python puede implementar algunas optimizaciones que hacen que el código que usa tuplas sea ligeramente más rápido que el que usa listas.

Conversión de tipos con las funciones list() y tuple()

Al igual que `str(42)` devuelve `'42'`, la representación en cadena del entero `42`, las funciones `list()` y `tuple()` devuelven versiones en lista y tupla de los valores que se pasan. Introduzca lo siguiente en el intérprete de comandos interactivo y observe que el valor devuelto es de un tipo de datos diferente al valor pasado:

```
>>> tupla(['gato', 'perro', 5])
('gato', 'perro', 5)
>>> list(('gato', 'perro', 5))
['gato', 'perro', 5]
>>> list('hola')
['h', 'e', 'l', 'l', 'o']
```

Convertir una tupla en una lista es útil si necesitas una versión mutable de un valor de tupla.

Referencias

Como has visto, las variables almacenan cadenas y valores enteros. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= 42
>>> queso= spam
>>> spam= 100
>>> spam
100
>>> queso
42
```

Asignas 42 a la spam, y luego copias el valor en spam y lo asignas a la variable queso. Cuando más tarde cambias el valor de spam a 100, esto no afecta al valor de cheese. Esto se debe a que spam y cheese son variables diferentes que almacenan valores diferentes.

Pero las listas no funcionan así. Cuando asignas una lista a una variable, en realidad estás asignando una *referencia de* lista a la variable. Una referencia es un valor que apunta a algún dato, y una referencia a una lista es un valor que apunta a una lista. A continuación se muestra un código que facilitará la comprensión de esta distinción. Introduce esto en el intérprete de comandos interactivo:

```
>>> ❶spam= [0, 1, 2, 3, 4, 5]

>>> ❷queso= spam

➲  >>> cheese[1]= '¡Hola!'
   >>> spam
   [0, '¡Hola!', 2, 3, 4, 5]
   >>> queso
   [0, '¡Hola!', 2, 3, 4, 5]
```

Esto puede parecerte extraño. El código cambió sólo la lista de queso, pero parece que tanto la lista de queso como la de spam han cambiado.

Cuando creas la lista ❶, le asignas una referencia en la variable spam. Pero la siguiente línea ❷ copia sólo la referencia de la lista en spam a cheese, no el valor de la lista en sí. Esto significa que los valores almacenados en spam y cheese ahora hacen referencia a la misma lista. Sólo hay una lista subyacente porque la lista en sí no se copiado nunca. Así que cuando modificas el primer elemento de cheese ❸, estás modificando la misma lista a la que hace referencia spam.

Recuerde que las variables son como cajas que contienen valores. Las cifras anteriores

en este capítulo muestran que las listas en cajas no son exactamente exactas porque las variables de lista no contienen listas, sino *referencias* a listas. (Estas referencias tendrán números ID que Python usa internamente, pero puedes ignorarlos). Usando las cajas como metáfora de las variables, la Figura 4-4 muestra lo que ocurre cuando se asigna una lista a la variable spam.

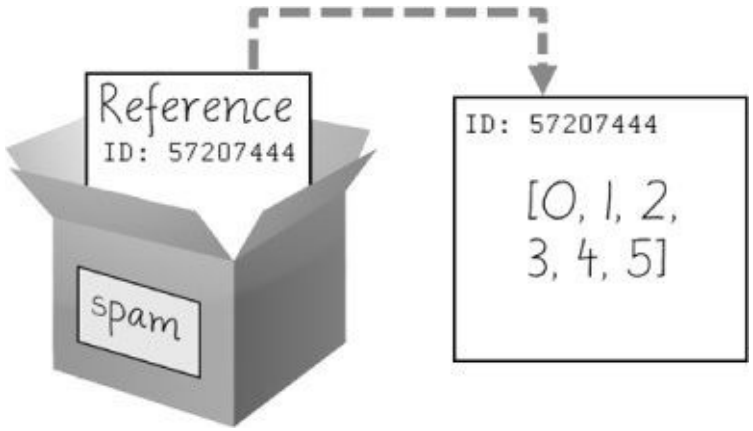


Figura 4-4. spam= [0, 1, 2, 3, 4, 5] *almacena una referencia a una lista, no la lista real.*

A continuación, en la Figura 4-5, la referencia en spam se copia en cheese. Sólo se ha creado y almacenado una nueva referencia en cheese, no una nueva lista. Observa cómo ambas referencias se refieren a la misma lista.

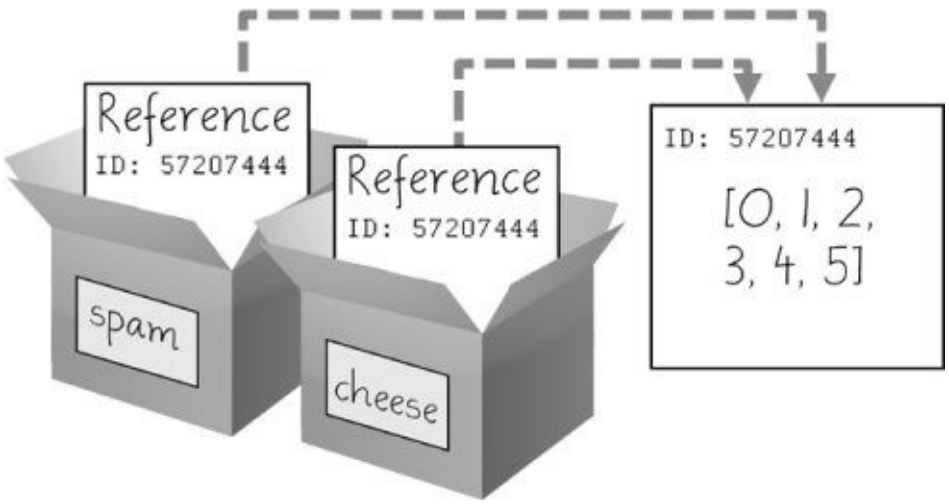


Figura 4-5. spam= *cheese copia la referencia, no la lista.*

Cuando se modifica la lista a la que hace referencia el queso, también se modifica la lista a la que hace referencia el spam, porque tanto el queso como el spam hacen referencia a la misma lista. Esto se puede ver en la Figura 4-6.

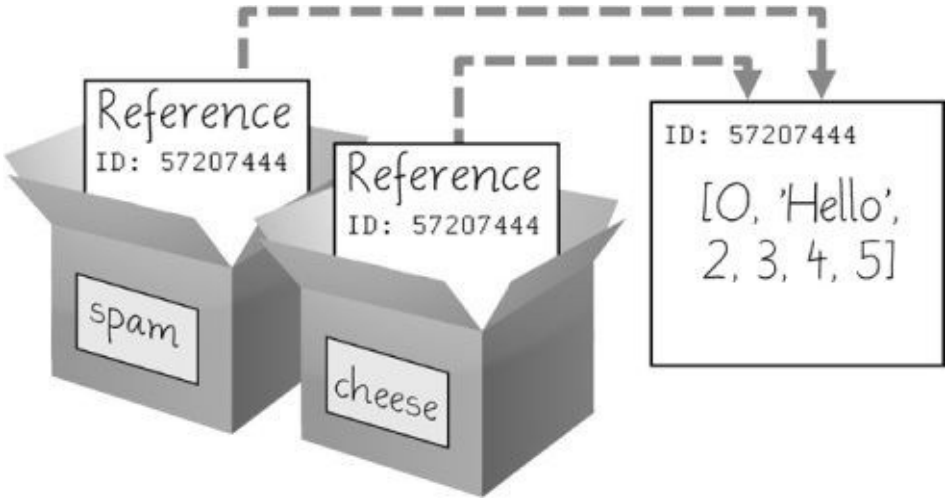


Figura 4-6. cheese[1]= '¡Hola!' *modifica la lista la que se refieren ambas variables.*

Las variables contendrán referencias a los valores de la lista en lugar de los propios valores de la lista. Pero para cadenas y valores enteros, las variables simplemente contienen la cadena o el valor entero. Python utiliza referencias cuando las variables deben almacenar valores de tipos de datos mutables, como listas o diccionarios. Para valores de tipos de datos inmutables cadenas, enteros o tuplas, las variables de Python almacenarán el valor en sí.

Aunque las variables de Python técnicamente contienen referencias a valores de listas o diccionarios, la gente suele decir casualmente que la variable contiene la lista o el diccionario.

Pasar referencias

Las referencias son especialmente importantes para entender cómo se pasan los argumentos a las funciones. Cuando se llama a una función, los valores de los argumentos se copian en las variables de los parámetros. Para las listas (y los diccionarios, que describiré en el próximo capítulo), esto significa que se utiliza una copia de la referencia para el parámetro. Para ver las consecuencias de esto, abre una nueva ventana del editor de archivos, introduce el siguiente código y guárdalo como *pasandoReferencia.py*:

```
def huevos(algúnParámetro):
    algúnParámetro.append("Hola")

spam= [1, 2, 3]
eggs(spam)
print(spam)
```

Observe que cuando se llama a eggs(), no se utiliza un valor de retorno para asignar un nuevo valor a spam. En su lugar, modifica la lista en su sitio, directamente. Cuando se ejecuta, este programa produce la siguiente salida:

```
[1, 2, 3, 'Hola']
```

Aunque spam y someParameter contienen referencias separadas, ambas se refieren a la misma lista. Esta es la razón por la que la llamada al método append('Hola') dentro de la función afecta a la lista incluso después de que la llamada a la función haya retornado.

Ten en cuenta este comportamiento: Olvidar que Python maneja las variables de lista y diccionario de esta manera puede llevar a errores confusos.

Funciones copy() y deepcopy() del módulo copy

Aunque pasar referencias es a menudo la forma más práctica de tratar con listas y diccionarios, si la función modifica la lista o diccionario que se pasa, puede que no quieras estos cambios en el valor original de la lista o diccionario. Para ello, Python proporciona un módulo llamado copy que proporciona las funciones copy() y deepcopy(). La primera de ellas, copy.copy(), se puede utilizar para hacer una copia duplicada de un valor mutable como una lista o diccionario, no sólo una copia de una referencia. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> copia de importación
>>> spam= ['A', 'B', 'C', 'D']
>>> queso= copia.copia(spam)
>>> queso[1]= 42
>>> spam
['A', 'B', 'C', 'D']
>>> queso
['A', 42, 'C', 'D']
```

Ahora las spam y queso se refieren a listas independientes, por lo que sólo se modifica la lista de queso al asignar 42 en el índice 1. Como puede ver en la Figura 4-7, los números de identificación de referencia ya no son los mismos para ambas variables porque las variables se refieren a listas independientes.

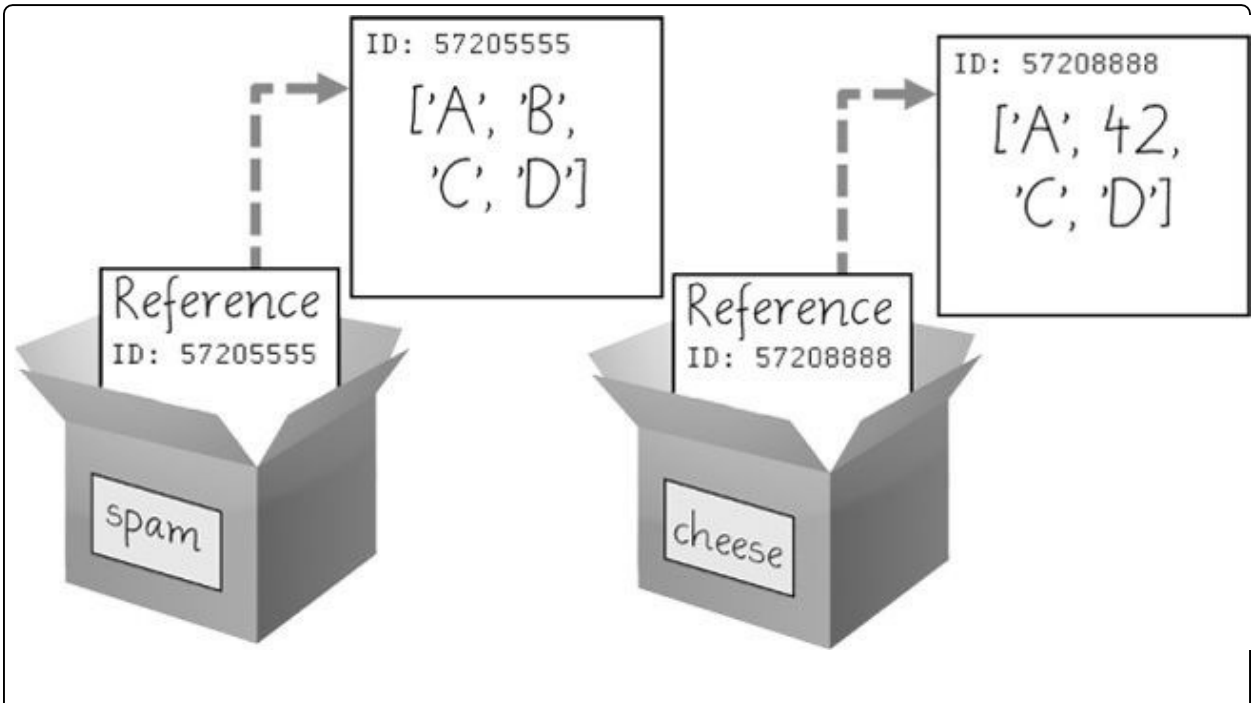


Figura 4-7. cheese= copy.copy(spam) *crea una segunda lista que puede modificarse independientemente de la primera.*

Si la lista que necesita copiar contiene listas, utilice la función copy.deepcopy() en lugar de copy.copy(). La función deepcopy() copiará también estas listas internas.

Resumen

Las listas son tipos de datos útiles, ya que permiten escribir código que trabaja con un número modificable de valores en una única variable. Más adelante en este libro, verás programas que utilizan listas para hacer cosas que serían difíciles o imposibles de hacer sin ellas.

Las listas son mutables, lo que significa que su contenido puede cambiar. Las tuplas y las cadenas, aunque se parecen a las listas en algunos aspectos, son inmutables y no pueden modificarse. Una variable que contiene un valor de tupla o cadena puede sobrescribirse con un nuevo valor de tupla o cadena, pero esto no es lo mismo que modificar el valor existente en su lugar, como hacen, por ejemplo, los métodos append() o remove() en las listas.

Las variables no almacenan valores de listas directamente, sino *referencias* a listas. Esta es una distinción importante cuando se copian variables o se pasan listas como argumentos en llamadas a funciones. Como el valor que se copia es la referencia a la lista, tenga en cuenta que cualquier cambio que haga en la lista puede afectar a otra variable de su programa. Puedes usar copy() o deepcopy() si quieres hacer cambios
a una lista en una variable sin modificar la lista original.

Preguntas prácticas

P: 1. ¿Qué es []?

P: 2. ¿Cómo asignarías el valor "hola" como tercer valor de una lista almacenada en una variable llamada spam? (Supongamos que spam contiene [2, 4, 6, 8, 10]).

Para las tres preguntas siguientes, supongamos que el spam contiene la lista ['a', 'b', 'c', 'd'].

P: 3. ¿A cuánto se evalúa spam[int(int('3' * 2) / 11)]?

P: 4. ¿A cuánto se evalúa spam[-1]?

P: 5. ¿A qué se evalúa spam[:2]?

Para las tres preguntas siguientes, supongamos que bacon contiene la lista [3.14, 'gato', 11, 'gato', Verdadero].

P: 6. ¿A cuánto se evalúa bacon.index('cat')?

P: 7. ¿Qué aspecto tiene el valor de la lista en bacon.append(99)?

P: 8. ¿Qué tiene el valor de la lista en bacon.remove('cat')?

P: 9. ¿Cuáles son los operadores para la concatenación de listas y la replicación de listas?

P: 10. ¿Cuál es la diferencia entre los métodos de lista append() e insert()?

P: 11. ¿Cuáles son las dos formas de eliminar valores de una lista?

Q: 12. Nombra algunas formas en que los valores de lista son similares a los valores de cadena.

P: 13. ¿Cuál es la diferencia entre listas y tuplas?

P: 14. ¿Cómo se escribe el valor de la tupla que sólo el valor entero 42?

P: 15. ¿Cómo se puede obtener la forma de tupla de un valor de lista? ¿Cómo se puede obtener la forma de lista de un valor de tupla?

Q: 16. Las variables que "contienen" valores de listas en realidad no contienen listas directamente. ¿Qué contienen en su lugar?

P: 17. ¿Cuál es la diferencia entre copy.copy() y copy.deepcopy()?

Proyectos prácticos

Para practicar, escribe programas que realicen las siguientes tareas.

Código Coma

Digamos que tienes un valor de lista como este

```
spam= ['manzanas', 'plátanos', 'tofu', 'gatos']
```

Escriba una función que tome un valor de lista como argumento y devuelva una cadena con todos los elementos separados por una coma y un espacio, con *y* insertado antes del último elemento. Por ejemplo, si pasas la lista de spam anterior a la función, obtendrás "manzanas, plátanos, tofu y gatos". Pero tu función debería poder trabajar con cualquier valor de lista que se le pase.

Cuadrícula de personajes

Digamos que tienes una lista de listas en la que cada valor de las listas interiores es una cadena de un carácter, como ésta:

```
cuadrícula= [['.', '.', '.', '.', '.', '.'],
            ['.', 'O', 'O', '.', '.', '.'],
            ['O', 'O', 'O', 'O', '.', '.'],
             ['O', 'O', 'O', 'O', 'O', '.'],
            ['.', 'O', 'O', 'O', 'O', 'O'],
             ['O', 'O', 'O', 'O', 'O', '.'],
            ['O', 'O', 'O', 'O', '.', '.'],
            ['.', 'O', 'O', '.', '.', '.'],
            ['.', '.', '.', '.', '.', '.']]
```

Puedes pensar en `grid[x][y]` como el carácter en las coordenadas x e y de una "imagen" dibujada con caracteres de texto. El origen `(0, 0)` estará en la esquina superior izquierda, las coordenadas x aumentan hacia la derecha y las coordenadas y aumentan hacia abajo.

Copie el valor de la cuadrícula anterior y escriba código que lo utilice para imprimir la imagen.

```
..OO.OO..
.OOOOOOO.
.OOOOOOO.
..OOOOO..
...OOO...
....O....
```

Pista: Tendrás que usar un bucle dentro de otro bucle para imprimir la `rejilla[0][0]`, luego `la rejilla[1][0]`, luego `la rejilla[2][0]`, y así sucesivamente, hasta `la rejilla[8][0]`. Esto terminará la primera fila, así que imprime una nueva línea. Luego tu programa debería imprimir `grid[0][1]`, luego `grid[1][1]`, luego `grid[2][1]`, y así sucesivamente. Lo último que imprimirá tu programa será `grid[8][5]`.

Además, recuerde pasar el argumento de la palabra clave `end` a `print()` si no desea que se imprima automáticamente una nueva línea después de cada llamada a `print()`.

Capítulo 5. Diccionarios y estructuración de datos
Diccionarios y estructuración de datos

En este capítulo, cubriré el tipo de datos diccionario, que proporciona una forma flexible de acceder y organizar datos. Luego, combinando diccionarios con tus conocimientos de listas del capítulo anterior, aprenderás a crear una estructura de datos para modelar un tablero de tres en raya.

El tipo de datos diccionario

Al igual que una lista, un *diccionario* es una colección de muchos valores. Pero, a diferencia de los índices de las listas, los índices de los diccionarios pueden utilizar muchos tipos de datos diferentes, no sólo números enteros. Los índices de los diccionarios se denominan *claves*, y una clave con su valor asociado se denomina *par clave-valor*.

En código, un diccionario se escribe con llaves, **{}**. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> myCat= {'size': 'fat', 'color': 'gray', 'DISPOSITION': 'loud'}
```

Esto asigna un diccionario a la variable myCat. Las claves de este diccionario son "tamaño", "color" y "disposición". Los valores de estas claves son "gordo", "gris" y "fuerte", respectivamente. Puedes acceder a estos valores a través de sus claves:

```
>>> myCat['tamaño']
grasa
>>> 'Mi gato TIENE '+ myCat['color']+ ' pelaje.'
'Mi gato tiene el pelo gris'.
```

Los diccionarios pueden seguir utilizando valores enteros como claves, al igual que las listas utilizan enteros para los índices, pero no tienen que empezar en 0 y pueden ser cualquier número.

```
>>> spam= {12345: 'Combinación de maletas', 42: 'La respuesta'}
```

Diccionarios frente a listas

A diferencia de las listas, los elementos de los diccionarios no están ordenados. El primer elemento de una lista llamada spam sería spam[0]. Pero en un diccionario no hay un "primer" elemento. Mientras que el orden de los elementos es importante para determinar si dos listas son iguales, no importa en qué orden se escriben los pares clave-valor en un diccionario. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= ['GATOS', 'PERROS', 'ALCES']
>>> bacon= ['PERROS', 'ALCES', 'GATOS']
>>> spam== bacon
Falso
>>> EGGS= {'name': 'Zophie', 'SPECIES': 'cat', 'age': '8'}
>>> ham= {'ESPECIE': 'gato', 'edad': '8', 'nombre': 'Zophie'}
>>> HUEVOS== jamón
Verdadero
```

Como los diccionarios no están ordenados, no se pueden trocear como las listas.

Si se intenta acceder a una clave que no existe en un diccionario, se producirá

un error del tipo

Mensaje de error KeyError, similar al error IndexError de una lista "fuera de rango mensaje. Introduzca lo siguiente en el intérprete de comandos interactivo, y observe el mensaje de error que aparece porque no hay tecla 'color':

```
>>> spam= {'nombre': 'Zophie', 'edad': 7}
>>> spam['color']
Traceback (llamada más reciente):
  File "<pyshell#1>", line 1, in <module> spam["color"]
KeyError: 'color'
```

Aunque los diccionarios no están ordenados, el hecho de que puedas tener valores arbitrarios para las claves te permite organizar tus datos de formas muy potentes. Digamos que quieres que tu programa almacene datos sobre los cumpleaños de tus amigos. Puedes utilizar un diccionario con los nombres como claves y los cumpleaños como valores. Abre una nueva ventana del editor de archivos e introduce el siguiente código. Guárdalo como *cumpleaños.py*.

```
❶ cumpleaños= {'Alice': 'Apr 1', 'Bob': 'Dec 12', 'Carol': "Mar 4'}

    mientras sea verdad:
        print('Introduce un nombre: (en blanco para
        salir)') name= input()
        si nombre== '':
            break
```

```
❷       si nombre en cumpleaños:
➲           print(cumpleaños[nombre]+ ' es el cumpleaños de '+
        nombre) else:
            print('No tengo información del cumpleaños de '+ nombre)
            print('¿Cuál es su cumpleaños?')
            bday= input()
❹           birthdays[name]= bday print('Base de
            datos de cumpleaños actualizada.')
```

Creas un diccionario inicial y lo almacenas en `cumpleaños`❶. Puedes ver si el nombre introducido existe como clave en el diccionario con la palabra clave `in`❷, igual que hiciste con las listas. Si el nombre está en el diccionario, accedes al valor asociado usando corchetes ❸; si no, puedes añadirlo usando la misma sintaxis de corchetes combinada con el operador de asignación ❹.

Cuando ejecutes este programa, tendrá el siguiente aspecto:

```
Introduzca un nombre: (en blanco para salir)
Alice
El 1 de abril es el cumpleaños
de Alice Introduce un nombre:
(en blanco para salir) Eve
No tengo información del cumpleaños de Eve
¿Cuál es su cumpleaños?
5 de diciembre
Base de datos de cumpleaños
actualizada. Introduzca un
nombre: (en blanco para salir)
Eve
El 5 de diciembre es el
cumpleaños de Eva Introduce
un nombre: (en blanco para
salir)
```

Por supuesto, todos los datos que introduzca en este programa se olvidan cuando el programa termina. Aprenderá cómo guardar datos en archivos del disco duro en el Capítulo 8.

Los métodos keys(), values() y items()

Existen tres métodos de diccionario que devuelven valores en forma de lista de las claves, valores o ambas claves y valores del diccionario: keys(), values() y items(). Los valores devueltos por estos métodos no son verdaderas listas: No pueden modificarse y no tienen un método append(). Pero estos tipos de datos (dict_keys, dict_values, y dict_items, respectivamente) *pueden* utilizarse en for bucles. Para ver cómo funcionan estos métodos, introduzca lo siguiente en el programa interactivo concha:

```
>>> spam= {'color': 'rojo', 'edad': 42}
>>> para v en spam.VALORES():
        print(v)

rojo
42
```

Aquí, un bucle for itera sobre cada uno de los valores del diccionario de spam. Un bucle for también puede iterar sobre las claves o sobre ambas claves y valores:

```
>>> para k en spam.KEYS():
        print(k)

color
edad
>>> para i en spam.ITEMS():
        print(i)

('color', 'rojo')
('edad', 42)
```

Utilizando los métodos keys(), values() y items(), un bucle for puede iterar sobre las claves, valores o pares clave-valor de un , respectivamente. Observe que los valores del valor dict_items devuelto por el método items() son tuplas de clave y valor.

Si quieres una lista verdadera de uno de estos métodos, pasa su valor de retorno tipo lista a la función list(). Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= {'color': 'rojo', 'edad': 42}
>>> spam.KEYS()
dict_keys(['color', 'edad'])
>>> list(spam.KEYS())
['color', 'edad']
```

La línea list(spam.keys()) toma el valor dict_keys devuelto por keys()
y lo pasa a list(), que devuelve un valor de lista de ['color', 'edad'].

También puede utilizar el truco de asignación múltiple en un bucle for para asignar la clave y el valor a variables separadas. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= {'color': 'rojo', 'edad': 42}
>>> para k, v en spam.ITEMS():
        print('Clave: '+ k+ ' Valor: '+ str(v))

Clave: edad Valor: 42
Clave: color Valor: rojo
```

Comprobación de la existencia de una clave o un valor en un diccionario

Recuerde del capítulo anterior que los operadores `in` y `not in` pueden comprobar si un valor existe en una lista. También puede utilizar estos operadores para comprobar si una determinada clave o valor existe en un diccionario. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= {'nombre': 'Zophie', 'edad': 7}
>>> 'nombre' en spam.KEYS()
Verdadero
>>> 'Zophie' en spam.VALUES()
Verdadero
>>> 'color' en spam.KEYS()
Falso
>>> 'color' no está en spam.KEYS()
Verdadero
>>> color' en spam
Falso
```

En el ejemplo anterior, observe que `'color' en spam` es esencialmente una versión abreviada de escribir `'color' en spam.keys()`. Esto es siempre: alguna vez desea comprobar si un valor es (o no es) una clave del , sólo tiene que utilizar la palabra clave `in` (o `not in`) con el propio valor del diccionario.

El método get()

Es tedioso comprobar si una clave existe en un diccionario antes de acceder al valor de esa clave. Afortunadamente, los diccionarios tienen un método get() que toma dos argumentos: la clave del valor a recuperar y un valor alternativo a devolver si esa clave no existe.

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> picnicItems= {'manzanas': 5, 'tazas': 2}
>>> 'Traigo '+ str(picnicItems.get('cups', 0))+ ' tazas.'
'Traigo 2 tazas'.
>>> 'Traigo '+ str(picnicItems.get('huevos', 0))+ ' huevos.'
'Traigo 0 huevos.'
```

Dado que no existe la clave 'eggs' en el diccionario picnicItems, el método get() devuelve el valor 0 por defecto. Sin utilizar get(), el código habría provocado un mensaje de error, como en el ejemplo siguiente:

```
>>> picnicItems= {'manzanas': 5, 'tazas': 2}
>>> 'Traigo '+ str(picnicItems['huevos'])+ ' huevos.'
Traceback (llamada más reciente):
  File "<pyshell#34>", line 1, in <module>
    'Traigo '+ str(picnicItems["huevos"])+  '  huevos.' KeyError:
'huevos'
```

El método setdefault()

A menudo tendrá que establecer un valor en un diccionario para una determinada clave sólo si esa clave no tiene ya un valor. El código es algo parecido a esto:

```
spam= {"nombre": 'Pooka', 'edad': 5} if
'color' not in spam:
    spam["color"]= 'negro'
```

El método setdefault() ofrece una manera de hacer esto en una línea de . El primer argumento pasado al método es la clave a , y el segundo argumento es el valor a establecer en esa clave si la clave no existe. Si la clave existe, el método setdefault() devuelve el valor de la clave. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= {'nombre': 'Pooka', 'edad': 5}
>>> spam.setdefault('color', 'negro')
negro
>>> spam
{'color': 'negro', 'edad': 5, "name": 'Pooka'}
>>> spam.setdefault('color', 'blanco')
negro
>>> spam
{'color': 'negro', 'edad': 5, "name": 'Pooka'}
```

La primera vez que se llama a setdefault(), el diccionario en spam cambia a {'color': 'negro', 'edad': 5, "nombre": 'Pooka'}. El método devuelve el valor "negro" porque éste es ahora el valor establecido para la clave "color". Cuando a continuación se llame a spam.setdefault("color", 'white'), el valor de esa clave *no* cambiará a 'white' porque el spam ya tiene una clave llamada 'color'.

El método setdefault() es un buen atajo para asegurarse de que una clave existe. He aquí un breve programa que cuenta el número de ocurrencias de cada letra en una cadena. Abra la ventana del editor de archivos e introduzca el siguiente código, guardándolo como *characterCount.py*:

```
mensaje= 'Era un día luminoso y frío de abril, y los relojes marcaban las trece'.
count= {}

para carácter en mensaje:
    count.setdefault(carácter, 0) count[carácter]=
    count[carácter]+ 1

print(count)
```

El programa hace un bucle sobre cada carácter de la cadena de la variable mensaje, contando cuántas veces aparece cada carácter. La llamada al método setdefault() asegura que la clave está en el diccionario count (con un valor por defecto de 0) para el programa no lance un error KeyError cuando count[character= count[character]+ 1 se ejecuta. Cuando ejecutes este programa, la salida tendrá este aspecto:

```
{' ': 13, ',': 1, '.': 1, 'A': 1, 'I': 1, 'a': 4, 'c': 3, 'b': 1, 'e': 5, 'd': 3,
'g': 2, 'i':
6, 'h': 3, 'k': 2, 'l': 3, 'o': 2, 'n': 4, 'p': 1, 's': 3, 'r': 5, 't': 6, 'w': 2,
'y': 1}
```

En la salida, puede ver que la letra minúscula *c* aparece 3 veces, el espacio aparece 13 veces y la letra mayúscula *A* aparece 1 vez. Este programa funcionará independientemente de la cadena que contenga la variable de mensaje, ¡incluso si la cadena tiene millones de caracteres!

Impresión bonita

Si importa el módulo pprint en sus programas, tendrá acceso a las funciones pprint() y pformat() que "imprimen bonito" los valores de un . Esto es útil cuando se desea una visualización más limpia de los elementos de un diccionario que la que proporciona print(). Modifique el programa *characterCount.py* anterior y guárdelo como *prettyCharacterCount.py*.

```
importar pprint
mensaje= 'Era un día luminoso y frío de abril, y los relojes marcaban las trece'.
count= {}

para carácter en mensaje:
    count.setdefault(carácter, 0) count[carácter]=
    count[carácter]+ 1

pprint.pprint(recuento)
```

Esta vez, al ejecutar el programa, la salida tiene un aspecto mucho más limpio, con las claves ordenadas.

```
{' ': 13,
 ',': 1,
 '.': 1,
'A': 1,
 'I': 1,
'a': 4,
'b': 1,
'c': 3,
'd': 3,
'e': 5,
'g': 2,
'h': 3,
 'i': 6,
'k': 2,
 'l': 3,
'n': 4,
'o': 2,
'p': 1,
 'r': 5,
's': 3,
 't': 6,
'w': 2,
 'y': 1}
```

La función pprint.pprint() es especialmente útil cuando el propio diccionario contiene listas o diccionarios anidados.

Si desea obtener el texto embellecido como un valor de cadena en lugar de en pantalla, llame en su lugar a pprint.pformat(). Estas dos líneas son equivalentes entre sí:

```
pprint.pprint(algúnValorDiccionario)
print(pprint.pformat(algúnValorDiccionario))
```

Estructuras de datos para modelar cosas del mundo real

Incluso antes de , era posible jugar una partida de ajedrez con alguien al otro lado del mundo. Cada jugador instalaba un tablero en su casa y se enviaba por turnos una postal describiendo cada jugada. Para ello, los jugadores necesitaban una forma de describir sin ambigüedades el estado del tablero y sus jugadas.

En *la notación algebraica del ajedrez*, los espacios del tablero se identifican mediante una coordenada de número y letra, como en la figura 5-1.

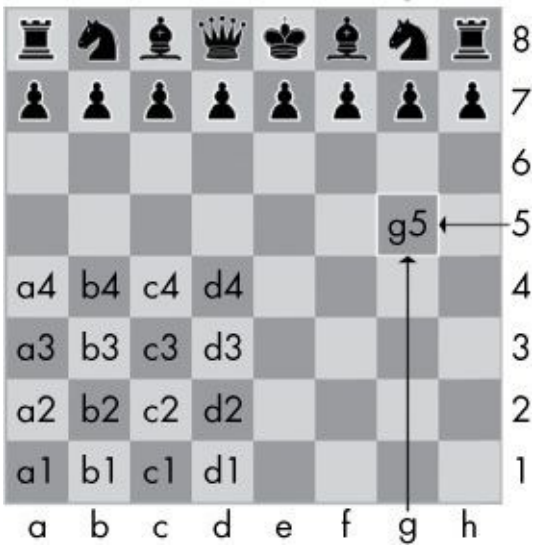


Figura 5-1. Las coordenadas de un tablero de ajedrez en notación algebraica de ajedrez

Las piezas de ajedrez se identifican por letras: *K* para rey, *Q* para reina, *R* para torre, *B* para alfil y *N* para caballo. Para describir una jugada se utiliza la letra de la pieza y las coordenadas de su destino. Un par de estas jugadas describe lo que ocurre en un solo turno (con las blancas primero); por ejemplo, la notación *2. Cf3 Cc6* indica que las blancas movieron un caballo a f3 y las negras un caballo a c6 en el segundo turno de la partida.

La notación algebraica es algo más que esto, pero la cuestión es que se puede utilizar para describir sin ambigüedades una partida de ajedrez sin necesidad de estar delante de un tablero. El adversario puede estar incluso al otro lado del mundo. De hecho, ni siquiera necesita un tablero físico si tiene buena memoria: Puedes simplemente leer las jugadas de ajedrez enviadas por correo y actualizar los tableros que tienes en tu imaginación.

Los ordenadores tienen buena memoria. Un programa en un ordenador moderno puede almacenar fácilmente miles de millones de cadenas como `'2. Cf3 Cc6'`. Así es como los ordenadores pueden jugar al ajedrez sin tener un tablero físico. Modelan los datos para representar un tablero de ajedrez, y se puede escribir código para trabajar con este modelo.

Aquí es donde entran en juego las listas y los diccionarios. Puedes usarlos para modelar cosas del mundo real, como tableros de ajedrez. Para el primer ejemplo, utilizaremos un juego un poco más simple que el ajedrez: el tres en raya.

Un tablero de tres en raya

Un tablero de tres en raya se parece a un gran símbolo de almohadilla (#) con nueve ranuras que pueden contener una *X*, una *O* o un espacio en blanco. Para representar el tablero con un diccionario, puedes asignar a cada ranura una clave cadena-valor, como se muestra en la Figura 5-2.

Puede utilizar valores de cadena para representar lo que hay en cada ranura del tablero: 'X', 'O', o ' ' (un carácter de espacio). lo tanto, necesitarás almacenar nueve cadenas. Para ello, puedes utilizar un diccionario de valores. El valor de cadena con la clave 'top-R' puede representar la esquina superior derecha, el valor de cadena con la clave 'low-L' puede
representa la esquina inferior izquierda, el valor de cadena con la clave 'mid-M' puede representar el centro, y así sucesivamente.

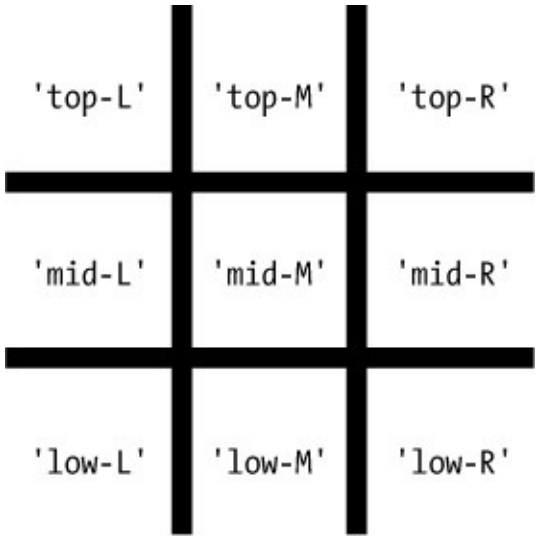


Figura 5-2. Las ranuras de un tablero tic-tactoe con sus teclas correspondientes

Este diccionario es una estructura de datos que representa un tablero de tres en raya. Guarda este tablero-como-un-diccionario en una variable llamada elTablero. Abre una nueva ventana del editor de archivos, e introduce el siguiente código fuente, guardándolo como *ticTacToe.py*:

```
theBoard= {'top-L': ' ', 'top-M': ' ', 'top-R': ' ',
           'mid-L': ' ', 'mid-M': ' ', 'mid-R': ' ',
           'low-L': ' ', 'low-M': ' ', 'low-R': ' '}
```

La estructura de datos almacenada en la variable theBoard representa el tablero tic-tactoe de la Figura 5-3.

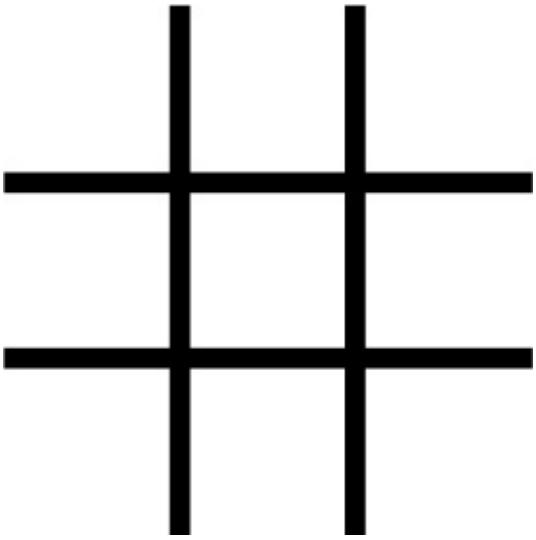


Figura 5-3. Un tablero de tres en raya vacío

Dado que el valor de cada clave en theBoard es una cadena de un solo espacio, este representa un tablero completamente despejado. Si el jugador X fuera primero y eligiera el espacio del medio, podrías representar ese tablero con este diccionario:

```
theBoard= {'top-L': ' ', 'top-M': ' ', 'top-R': ' ',
           'mid-L': ' ', 'mid-M': 'X', 'mid-R': ' ',
           'low-L': ' ', 'low-M': ' ', 'low-R': ' '}
```

La estructura de datos de theBoard representa ahora el tablero de tres en raya de la Figura 5-4.

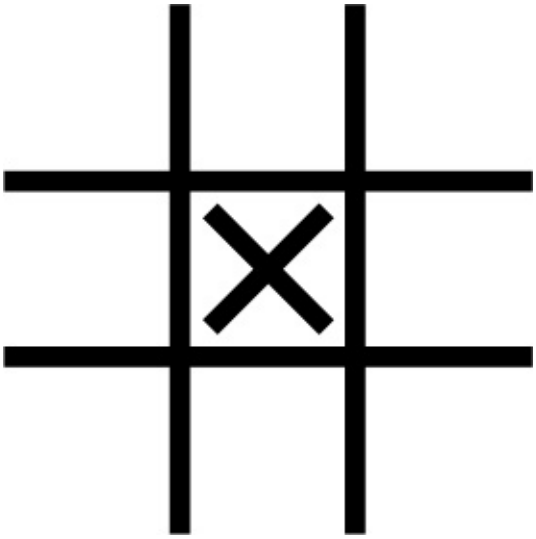


Figura 5-4. El primer movimiento

Un tablero en el que el jugador O ha ganado colocando Os en la parte superior podría tener el siguiente aspecto

esto:

```
theBoard= {'top-L': 'O', 'top-M': 'O', 'top-R': 'O',
           'mid-L': 'X', 'mid-M': 'X', 'mid-R': ' ',
           'low-L': ' ', 'low-M': ' ', 'low-R': 'X'}
```

La estructura de datos de theBoard representa ahora el tablero de tres en raya de la Figura 5-5.

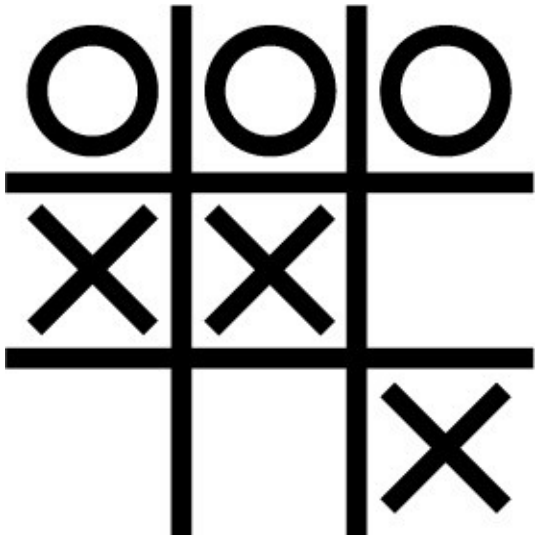


Figura 5-5. El jugador O gana.

Por supuesto, el jugador sólo ve lo que se imprime en la pantalla, no el contenido de las variables. Vamos a crear una función para imprimir el diccionario del tablero en la pantalla. Haz la siguiente adición a *ticTacToe.py* (el nuevo código está en negrita):

```
theBoard= {'top-L': ' ', 'top-M': ' ', 'top-R': ' ',
           'mid-L': ' ', 'mid-M': ' ', 'mid-R': ' ',
           'low-L': ' ', 'low-M': ' ', 'low-R': ' '}
def imprimirPizarra(tabla):
    print(tablero['top-L']+ '|'+ tablero['top-M']+ '|'+ tablero['top-R'])
    print('-+-+-')
    print(tablero['medio-L']+ '|'+ tablero['medio-M']+ '|'+ tablero['medio-
    R']) print('-+-+-')
    print(board['low-L']+ '|'+ board['low-M']+ '|'+ board['low-R']) printBoard(theBoard)
```

Cuando ejecutes este programa, printBoard() imprimirá un tablero de tic-tac en blanco.

```
  |
-+-+-
  |
-+-+-
```

```
||
```

La función printBoard() puede manejar cualquier estructura de datos tic-tac-toe que pases. Prueba a cambiar el código por el siguiente:

```
theBoard= {'top-L': 'O', 'top-M': 'O', 'top-R': 'O', 'mid-L': 'X', 'mid-M':
X', 'mid-R': ' ', 'low-L': ' ', 'low-M': ' ', 'low-R': 'X'}

def imprimirPizarra(tabla):
    print(tablero['top-L']+ '|'+ tablero['top-M']+ '|'+ tablero['top-R']) print('-
    +-+-')
    print(tablero['medio-L']+ '|'+ tablero['medio-M']+ '|'+ tablero['medio-R'])
    print('-+-+-')
    print(board['low-L']+ '|'+ board['low-M']+ '|'+ board['low-R']) printBoard(theBoard)
```

Ahora, cuando ejecutes este programa, la nueva placa se imprimirá en la pantalla.

```
O|O|O
-+-+-
X|X|
-+-+-
| |X
```

Como creó una estructura de datos para representar un tablero de tres en raya y escribió código en printBoard() para interpretar esa estructura de datos, ahora tiene un programa que "modela" el tablero de tres en raya. Podría haber organizado su estructura de datos de forma diferente (por ejemplo, usando claves como 'TOP-LEFT' en lugar de 'top- L'), pero mientras el código funcione con sus estructuras de datos, tendrá un programa que funciona correctamente.

Por ejemplo, la función printBoard() espera la estructura de datos del tres en raya sea un diccionario con claves para las nueve ranuras. Si al diccionario que pasas le faltara, por ejemplo, la tecla 'mid-L', tu programa dejaría de funcionar.

```
O|O|O
-+-+-
Traceback (llamada más reciente):
  File "ticTacToe.py", line 10, in <module>
    printBoard(theBoard)
  File "ticTacToe.py", line 6, in printBoard
    print(board['mid-L']+ '|'+ board['mid-M']+ '|'+ board['mid-R']) KeyError: 'mid-
L'
```

Ahora vamos a añadir código que permita a los jugadores introducir sus movimientos. Modifica el archivo
ticTacToe.py para que tenga este aspecto:

```
theBoard= {'top-L': ' ', 'top-M': ' ', 'top-R': ' ', 'mid-L': ' ', 'mid-M': '
    ', 'mid-R': ' ', 'low-L': ' ', 'low-M': ' ', 'low-R': ' '}

    def imprimirPizarra(tabla):
        print(tablero['top-L']+ '|'+ tablero['top-M']+ '|'+ tablero['top-R']) print('-
        +-+-')
        print(tablero['mid-L']+ '|'+ tablero['mid-M']+ '|'+ tablero['mid-R'])
        print('-+-+-')
        print(tablero['bajo-L']+ '|'+ tablero['bajo-M']+ '|'+ tablero['bajo-R'])
    gire= 'X'
    para i en rango(9):
❶       printPizarra(laPizarra)
        print('Gira para '+ gira+ '. ¿Mover en qué espacio?')
❷       mover= entrada()
➲       theBoard[move]= turn
❹       si gira== 'X':
            gira= 'O'
        si no:
            gire= 'X'
    printPizarra(laPizarra)
```

El nuevo código imprime el tablero al comienzo de cada nuevo turno ❶, obtiene el movimiento del jugador activo ❷, actualiza el tablero de juego en consecuencia ❸, y luego intercambia el jugador activo ❹ antes de pasar al siguiente turno.

Cuando ejecutes este programa, tendrá este aspecto:

```
  |  |
-+-+-
  |  |
-+-+-
  |  |
Giro para X. ¿Movimiento en qué espacio?
mediados de M
  |  |
-+-+-
  |X|
-+-+-
  |  |
Giro para O. ¿Movimiento en qué espacio?
low-L
  |  |
-+-+-
  |X|
-+-+-
|

--snip--
```

```
O|O|X
-+-+-
X|X|O
-+-+-
O| |X
Giro para X. ¿Movimiento en qué espacio?
baja-M
O|O|X
-+-+-
X|X|O
-+-+-
O|X|X
```

No es un juego de tres en raya completo -por ejemplo, no comprueba nunca si un jugador ha ganado-, pero es suficiente para ver cómo pueden utilizarse las estructuras de datos en los programas.

NOTA

Si tiene curiosidad, el código fuente de un programa completo de tres en raya se describe en los recursos disponibles en http://nostarch.com/automatestuff/.

Diccionarios y listas anidados

Modelar un tablero de tres en raya era bastante sencillo: El tablero sólo necesitaba un único valor de diccionario con nueve pares clave-valor. A medida que modeles cosas más complicadas, puede que necesites diccionarios y listas que contengan otros diccionarios y listas. Las listas son útiles para contener una serie ordenada de valores, y los diccionarios son útiles para asociar claves con valores. Por ejemplo, aquí tienes un programa que utiliza un diccionario que contiene otros diccionarios para ver quién lleva qué a un picnic. La función totalBrought() puede leer esta estructura de datos y calcular el número total de un artículo que traen todos los invitados.

```
allGuests= {'Alice': {'manzanas': 5, 'pretzels': 12},
                    'Bob': {'sandwiches de jamón': 3, 'manzanas': 2},
                    'Carol': {'tazas': 3, 'tartas de manzana': 1}}

    def totalCompra(invitados, artículo):
        númeroCompra= 0
❶       para k, v en invitados.items():
❷           numBrought= numBrought+ v.get(item, 0) return
        numBrought

    print('Número de cosas que se traen:')
    print(' - Manzanas         '+ str(totalSuministrado(todosLosInvitados,
    'manzanas')) print(' - Tazas       '+
    str(totalSuministrado(todosLosInvitados, 'tazas')) print(' - Pasteles
                                '+ str(totalComprado(todosLosInvitados,
    'pasteles'))
    print(' - Bocadillos de jamón '+ str(totalSuministrado(todosLosInvitados, 'bocadillos
    de jamón')) print(' - Tartas de manzana      +
    str(totalSuministrado(todosLosInvitados, 'tartas de manzana'))
```

Dentro de la función totalBrought(), el bucle for itera sobre los pares clave-valor en invitados ❶. Dentro del , la cadena del nombre del invitado se asigna a k, y el diccionario de artículos de picnic que trae se asigna a v. Si el parámetro de artículo existe como clave en este diccionario, su valor (la cantidad) se añade a numBrought ❷. Si no existe como clave, el método get() devuelve 0 que se añadirá a numBrought.

La salida de este programa tiene el siguiente aspecto:

```
Número de cosas que se traen:
- Manzanas 7
- Copas 3
- Pasteles 0
- Bocadillos de jamón 3
- Pasteles de manzana  1
```

Esto puede parecer algo tan simple de modelar que no necesitarías molestarte en escribir un programa para hacerlo. Pero date cuenta de que esta misma función totalBrought() podría manejar fácilmente un diccionario que contenga miles de invitados, cada uno trayendo *miles* de artículos de picnic diferentes. Entonces, ¡tener esta información en una estructura de datos junto con la función totalBrought() te ahorraría mucho tiempo!

Puedes modelar las cosas con estructuras de datos de la forma que quieras, siempre que el resto del código de tu programa pueda trabajar con el modelo de datos correctamente. Cuando empieces a programar, no te preocupes demasiado por la forma "correcta" de modelar los datos. A medida que adquieras más experiencia, puede que se te ocurran modelos más eficientes, pero lo importante es que el modelo de datos funcione para las necesidades de tu programa.

Resumen

En este capítulo aprendiste todo sobre los diccionarios. Las listas y los diccionarios son valores que pueden contener múltiples valores, incluyendo otras listas y diccionarios. Los diccionarios son útiles porque puedes asignar un elemento (la clave) a otro (el valor), a diferencia de las listas, que simplemente contienen una serie de valores en orden.
Para acceder a los valores de un diccionario se utilizan corchetes, igual que en las listas. En lugar de un índice entero, los diccionarios pueden tener claves de distintos tipos de datos: enteros, flotantes, cadenas o tuplas. Organizando los valores de un programa en estructuras de datos, puedes crear representaciones de objetos del mundo real. Has visto un ejemplo de esto con un tablero de tres en raya.

Preguntas prácticas

Q: 1. ¿Qué tiene el código de un diccionario vacío?

Q: 2. ¿Qué tiene un valor de diccionario con una clave `"foo"` y un valor `42`?

Q: 3. ¿Cuál es la principal diferencia entre un diccionario y una lista?

Q: 4. ¿Qué ocurre si intentas acceder a `spam["foo"]` si `spam` es `{'bar': 100}`?

Q: 5. Si se almacena un diccionario en `spam`, ¿cuál es la diferencia entre las expresiones `'cat' en spam` y `'cat' en spam.keys()`?

Q: 6. Si se almacena un diccionario en `spam`, ¿cuál es la diferencia entre las expresiones `'cat' en spam` y `'cat' en spam.values()`?

Q: 7. ¿Cuál es la abreviatura del siguiente código?

```
if 'color' not in spam: spam["color"]=
    'negro'
```

P: 8. ¿Qué módulo y función se pueden utilizar para "imprimir bonito" los valores del diccionario?

Proyectos prácticos

Para practicar, escribe programas que realicen las siguientes tareas.

Inventario de juegos de fantasía

Estás creando un videojuego de fantasía. La estructura de datos para modelar el inventario del jugador será un diccionario donde las claves son valores de cadena que describen el artículo en el inventario y el valor es un valor entero que detalla cuántos de ese artículo tiene el jugador. Por ejemplo, el valor del diccionario `{'cuerda': 1, 'antorcha': 6, 'moneda de oro': 42, 'daga': 1, 'flecha': 12}` significa que el jugador tiene 1 cuerda, 6 antorchas, 42 monedas de oro, etc.

Escriba una función llamada `displayInventory()` que tome cualquier "inventario" posible y lo muestre de la siguiente manera:

```
Inventario:
12 flechas
42 monedas de oro
1 cuerda
6 antorcha
1 daga

Número total de artículos: 62
```

Sugerencia: Puede utilizar un bucle `for` para recorrer todas las claves de un diccionario.

```
# inventory.py
cosas= {'cuerda': 1, antorcha': 6, 'moneda de oro': 42, 'daga': 1, 'flecha': 12}

def mostrarInventario(inventario):
    print("Inventario:")
    item_total= 0
    para k, v en inventario.items():
        # RELLENA ESTA PARTE
    print("Número total de artículos: "+ str(total_artículos))

displayInventory(cosas)
```

Función de lista a diccionario para el inventario de juegos de fantasía

Imagina que el botín de un dragón vencido se representa como una lista de cadenas como ésta:

```
dragonLoot= ['moneda de oro', 'daga', 'moneda de oro', 'moneda de oro', 'rubí']
```

Escribe una función llamada addToInventory(inventory, addedItems), donde el parámetro inventory es un diccionario que representa el inventario del jugador (como en el proyecto anterior) y el parámetro addedItems es una lista como dragonLoot. La función addToInventory() debería devolver un diccionario que represente el inventario del jugador.
inventario actualizado. Tenga en cuenta que la lista addedItems puede contener múltiplos del mismo artículo. Su código podría ser algo como esto

```
def addToInventory(inventario, addedItems): # tu
    código va aquí

inv= {'moneda de oro': 42, 'cuerda': 1}
dragonLoot = ['moneda de oro', 'daga', 'moneda de oro', 'moneda de
oro', 'rubí'] inv= addToInventory(inv, dragonLoot)
mostrarInventario(inv)
```

El programa anterior (con su función displayInventory() del proyecto anterior) mostraría lo siguiente:

```
Inventario:
45 monedas de oro
1 cuerda
1 rubí
1 daga

Número total de artículos: 48
```

Capítulo 6. Manipulación de cadenas Manipulación de cadenas

El texto es una de las formas más comunes de datos que manejarán sus programas. Ya sabes cómo concatenar dos valores de cadena con el operador**+** , pero puedes hacer mucho más que eso. Puede extraer cadenas parciales de valores de cadena, añadir o eliminar espaciado, convertir letras a minúsculas o mayúsculas, y comprobar que las cadenas tienen el formato correcto. Incluso puede escribir código Python para acceder al portapapeles y copiar y pegar texto.

En este capítulo aprenderás todo esto y mucho más. A continuación, trabajarás en dos proyectos de programación diferentes: un sencillo gestor de contraseñas y un programa para automatizar la aburrida tarea de dar formato a trozos de texto.

Trabajar con cuerdas

Veamos algunas de las formas en que Python te permite escribir, imprimir y acceder a cadenas en tu código.

Literales de cadena

Escribir valores de cadena en código Python es bastante sencillo: Empiezan y terminan con una comilla simple. Pero entonces, ¿cómo se puede utilizar una comilla dentro de una cadena?

Escribir **'Ese ES EL gato DE Alicia'** no funcionará, porque Python piensa que la cadena termina después de Alicia, y el resto (s gato') es código Python inválido. Afortunadamente, hay varias formas de escribir cadenas.

Citas dobles

Las cadenas pueden empezar y terminar con comillas dobles, igual que con comillas simples. Una de las ventajas de utilizar comillas dobles es que la cadena puede contener un carácter de comilla simple. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= "Ese ES el gato DE Alice".
```

Dado que la cadena comienza con una comilla doble, Python sabe que la comilla simple forma parte de la cadena y no marca el final de la misma. Sin embargo, si utilizar tanto comillas simples como dobles en la cadena, tendrás que utilizar caracteres de escape.

Escapar Personajes

Un *carácter de escape* le permite utilizar caracteres que de otro modo sería imposible introducir en una cadena. Un carácter de escape consiste en una barra invertida (\) seguida del carácter que desea añadir a la cadena. (A pesar de constar de dos caracteres, se suele denominar carácter de escape singular). Por ejemplo, el carácter de escape para una comilla simple es'. Puede utilizarlo dentro de una cadena que empiece y termine con comillas simples. Para ver cómo funcionan los caracteres de escape, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= 'Saluda a la madre DE Bob\'.
```

Python sabe que como la comilla simple en Bob\'s tiene una barra invertida, no es una comilla simple destinada a terminar el valor de la cadena. Los caracteres de escape**\'** y**\"** le permiten poner comillas simples y dobles dentro de sus cadenas, respectivamente.

La Tabla 6-1 enumera los caracteres de escape que puedes utilizar.

Tabla 6-1. Caracteres de escape

Carácter de escape	Imprime como
\'	Presupuesto único

\"	Cita doble
\t	Pestaña
\n	Nueva línea (salto de línea)
\\	Barra diagonal inversa

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> print("! ¿Cómo ? Estoy bien.")
¡Hola! ¿Cómo
? Bien.
```

Cuerdas en bruto

Puede colocar una r antes de la comilla inicial de una cadena para convertirla en una cadena sin procesar. Una *cadena raw* ignora completamente todos los caracteres de escape e imprime cualquier barra invertida que aparezca en la cadena. Por ejemplo, escriba lo siguiente en el intérprete de comandos interactivo:

```
>>> print(r'Ese ES EL gato DE Carol.')
Ese es el gato de Carol.
```

Como se trata de una cadena sin procesar, Python considera la barra invertida como parte de la cadena y no como el comienzo de un carácter de escape. Las cadenas sin procesar son útiles si está escribiendo valores de cadena que contienen muchas barras invertidas, como las cadenas usadas para expresiones regulares descritas en el siguiente capítulo.

Cadenas de varias líneas con comillas triples

Aunque puedes usar el carácter de escape \n para poner una nueva línea en una cadena, a menudo es más fácil usar cadenas multilínea. Una cadena multilínea en Python comienza y termina con tres comillas simples o tres comillas dobles. Todas las comillas, tabulaciones o nuevas líneas entre las "comillas triples" se consideran parte de la cadena.
Las reglas de indentación de Python para bloques no se aplican a las líneas dentro de una cadena multilínea.

Abre el editor de archivos y escribe lo siguiente:

```
print('''Querida Alice,

El gato de Eve ha sido detenido por secuestro, robo y extorsión.
```

```
Atentamente
, Bob')
```

Guarda este programa como *catnapping.py* y . La salida se verá así:

```
Querida Alice,

El gato de Eve ha sido detenido por secuestro, robo y extorsión de gatos. Atentamente,
Bob
```

Observe que el carácter de comilla simple en `Eve's` no necesita ser escapado. Escapar las comillas simples y dobles es opcional en las cadenas multilínea. La siguiente llamada a `print()` imprimiría un texto idéntico no utiliza una cadena multilínea:

```
print('Querida Alice, el gato de Eva ha sido arrestado por secuestro, robo y
extorsión. Atentamente, Bob')
```

Comentarios multilínea

Mientras que el carácter almohadilla (`#`) marca el inicio de un comentario para el resto de la línea, una cadena multilínea se utiliza a menudo para comentarios que abarcan varias líneas. Lo siguiente es código Python perfectamente válido:

```
"""Este es un programa Python de prueba.
Escrito por Al Sweigart al@inventwithpython.com

Este programa fue diseñado para Python 3, no para Python 2.
"""


def spam():
    """Este es un comentario multilínea para
    ayudar a explicar lo que hace la función
    spam()."""
    print('¡Hola!')
```

Indexación y segmentación de cadenas

Las cadenas utilizan índices y rebanadas del mismo modo las listas. Puedes considerar la cadena **"¡Hola mundo!"** como una lista y cada carácter de la cadena como un elemento con su índice correspondiente.

```
 '  H  e  l  l  o     w  o  r  l  d  ¡!  '
    0  1  2  3  4  5  6  7  8  9  10 11
```

El espacio y el signo de exclamación se incluyen en el recuento de caracteres, por lo que **"¡Hola mundo!"** tiene 12 caracteres, desde la H en el índice 0 hasta **!** en el índice 11.

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= '¡Hola mundo!'
>>> spam[0]
'H'
>>> spam[4]
'o'
>>> spam[-1]
'!'
>>> spam[0:5]
Hola.
>>> spam[:5]
Hola.
>>> spam[6:]
"¡Mundo!
```

Si especifica un índice, obtendrá el carácter en esa posición de la cadena. especifica un rango de un índice a otro, se incluye el índice inicial y no el final. Por eso, si spam es **"¡Hola mundo!"**, spam[0:5] es "Hola". La subcadena que obtengas de spam[0:5] incluirá todo lo que va de spam[0] a spam[4], dejando fuera el espacio del índice 5.

Tenga en cuenta que rebanar una cadena no modifica la cadena original. Puede capturar una rebanada de una variable en una variable separada. Intente escribir lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= '¡Hola mundo!'
>>> fizz= spam[0:5]
>>> efervescencia
Hola.
```

Al cortar y almacenar la subcadena resultante en otra variable, puedes tener a mano tanto la cadena completa como la subcadena para acceder a ellas de forma rápida y sencilla.

Los operadores in y not in con cadenas

Los operadores in y not in pueden utilizarse con cadenas de caracteres igual que con valores de listas. Una expresión con dos cadenas unidas mediante in o not in se evaluará como un booleano True o False. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> Hola" en "Hola Mundo
Verdadero
>>> Hola" en "Hola
Verdadero
>>> HELLO"" en "Hello World
Falso
>>> "" en 'spam'
Verdadero
>>> 'GATOS' no en 'PERROS y gatos'
Falso
```

Estas expresiones comprueban si la primera cadena (la cadena exacta, distinguiendo mayúsculas de minúsculas) puede encontrarse dentro de la segunda cadena.

Métodos de cadena útiles

Varios métodos de cadena analizan cadenas o crean valores de cadena transformados. Esta sección describe los métodos que utilizará con más frecuencia.

Métodos de cadena upper(), lower(), isupper() e islower()

Los métodos de cadena upper() y lower() devuelven una nueva cadena en la que todas las letras de la cadena original se han convertido a mayúsculas o minúsculas, respectivamente. Los caracteres no alfabéticos de la cadena permanecen inalterados. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam= '¡Hola mundo!'
>>> spam= spam.upper()
>>> spam
''¡HOLA MUNDO !
>>> spam= spam.lower()
>>> spam
"¡Hola mundo!
```

Tenga en cuenta que estos métodos no cambian la cadena en sí, sino que devuelven nuevos valores de cadena. Si quiere cambiar la cadena original, tiene que llamar a upper() o lower() sobre la cadena y luego asignar la nueva cadena a la variable donde estaba almacenada la original. Esta es la razón por la que debe utilizar spam= spam.upper() para cambiar la cadena en spam en lugar de simplemente spam.upper(). (Esto es igual que si una variable eggs contiene el valor 10. Escribir eggs+ 3 no cambia el valor de eggs, pero eggs= eggs+ 3 sí).

Los métodos upper() y lower() son útiles si necesita hacer una comparación que no distinga entre mayúsculas y minúsculas. Las cadenas 'great' y 'GREat' no son iguales entre sí. Pero en el siguiente pequeño programa, no importa si el usuario escribe Great, GREAT, o grEAT, porque la cadena se convierte primero a minúsculas.

```
print('¿Cómo ?') feeling=
input()
if sentimiento.lower()==
    'genial': print('Yo
    también me siento
    genial.')
si no:
    print('Espero que el resto de tu día sea bueno')
```

Cuando se ejecuta este programa, se muestra la pregunta, y la introducción de una variación de gran, como GREat, seguirá dando la salida I feel great too. Añadir código a su programa para manejar variaciones o errores en la entrada del usuario, como mayúsculas inconsistentes, hará que sus programas sean más fáciles de usar y menos propensos a fallar.

¿Qué ?
GREat
Yo también me siento muy bien.

Los métodos isupper() e islower() devolverán un valor booleano True si la cadena tiene al menos una letra y todas las letras son mayúsculas o minúsculas, respectivamente. En caso contrario, el método devuelve False. Introduzca lo siguiente en el intérprete de comandos interactivo y observe lo que devuelve cada llamada al método:

```
>>> spam= '¡Hola mundo!'
>>> spam.ISLOWER()
Falso
>>> spam.ISUPPER()
Falso
>>> HELLO'.ISUPPER()
Verdadero
>>> 'abc12345'.ISLOWER()
Verdadero
>>> '12345'.ISLOWER()
Falso
>>> '12345'.ISUPPER()
Falso
```

Dado que los métodos de cadena upper() y lower() devuelven cadenas, también puede llamar a métodos de cadena sobre *los* valores de cadena devueltos. Las expresiones que hacen esto parecerán una cadena de llamadas a métodos. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> 'Hola'.upper()
HOLA
>>> 'Hola'.upper().lower()
Hola.
>>> 'Hola'.superior().inferior().superior()
HOLA
>>> HELLO'.lower()
Hola.
>>> HELLO'.lower().ISLOWER()
Verdadero
```

Los métodos de cadena isX

Junto con islower() e isupper(), existen varios métodos de cadena cuyos nombres comienzan con la palabra *is*. Estos métodos devuelven un valor booleano que describe la naturaleza de la cadena. Estos son algunos métodos de cadena isX comunes:

- isalpha() devuelve True si la cadena está formada sólo por letras y no está en blanco.
- isalnum() devuelve True si la cadena está formada sólo por letras y números y no está en blanco.
- isdecimal() devuelve True si la cadena está formada sólo por caracteres numéricos y no está en blanco.
- isspace() devuelve True si la cadena está formada sólo por espacios, tabuladores y nuevas líneas y no está en blanco.
- istitle() devuelve True si la cadena está formada sólo por palabras que empiezan por una letra mayúscula seguida sólo de letras minúsculas.

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> 'hola'.ISALPHA()
Verdadero
>>> 'hola123'.ISALPHA()
Falso
>>> 'hola123'.ISALNUM()
Verdadero
>>> 'hola'.ISALNUM()
Verdadero
>>> 123'.ISDECIMAL()
Verdadero
>>> '       '.ISSPACE()
Verdadero
>>> 'ESTE ES EL caso del título'.ISTITLE()
Verdadero
>>> 'ESTE ES EL caso del título 123'.ISTITLE()
Verdadero
>>> 'ESTE no es el caso del título'.ISTITLE()
Falso
>>> 'ESTO tampoco es el caso del título'.ISTITLE()
Falso
```

Los métodos de cadena *isX* son útiles cuando se necesita validar la entrada del usuario. Por ejemplo, el siguiente pregunta repetidamente a los usuarios su edad y un

contraseña hasta que proporcionen una entrada válida. Abra una nueva ventana del editor de archivos e introduzca este programa, guardándolo como *validateInput.py*:

```
mientras sea verdad:
    print('Introduzca su
    edad:') edad= input()
    si edad.isdecimal():
        break
    print('Por favor, introduzca un número para su edad.')

mientras sea verdad:
    print('Seleccione una nueva contraseña (sólo letras y números):')
    password= input()
    si contraseña.isalnum():
        break
    print('Las contraseñas sólo pueden tener letras y números.")
```

En el primer bucle while, preguntamos al usuario su edad y almacenamos su entrada en age. Si la edad es un valor válido (decimal), salimos de este primer bucle while y pasamos al segundo, que pide una contraseña. En caso contrario, informamos al usuario de que debe introducir un número y le pedimos de nuevo que introduzca su edad en . En el segundo bucle while, pedimos una contraseña, almacenamos la entrada del usuario en contraseña y salimos del bucle si la entrada es alfanumérica. Si no es así, no estamos satisfechos, así le decimos al usuario que la contraseña debe ser alfanumérica y le pedimos de nuevo que la introduzca.

Cuando se ejecuta, la salida del programa tiene este aspecto:

```
Introduce tu edad:
cuarenta y dos
Introduzca un número para su . Introduzca
su edad:
42
Seleccione una nueva contraseña (sólo letras y números):
¡Secr3t!
Las contraseñas sólo pueden tener letras y números.
Seleccione una nueva contraseña (sólo letras y
números):
secr3t
```

Llamando isdecimal() e isalnum() sobre variablespodemos comprobar si los valores almacenados en esas variables son decimales o no, alfanuméricos o no. Aquí, estas pruebas nos ayudan a rechazar la entrada cuarenta y dos y aceptar 42, y a rechazar secr3t! y aceptar secr3t.

Los métodos de cadena startswith() y endswith()

Los métodos startswith() y endswith() devuelven True si el valor de la cadena a la que se llama comienza o termina (respectivamente) con la cadena pasada al método; en caso contrario, devuelven False. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> '¡Hola mundo!'.STARTSWITH('Hola')
Verdadero
>>> '¡Hola mundo!'.ENDSWITH('mundo!')
Verdadero
>>>    'abc123'.STARTSWITH('abcdef')
Falso
>>> 'abc123'.ENDSWITH('12')
Falso
>>> '¡Hola mundo!'.STARTSWITH('¡Hola mundo!')
Verdadero
>>> '¡Hola mundo!'.ENDSWITH('¡Hola mundo!')
Verdadero
```

Estos métodos son alternativas útiles al operador== equals si necesita comprobar sólo si la primera o la última parte de la cadena, en lugar de toda ella, es igual a otra cadena.

Los métodos de cadena join() y split()

El `join()` es útil cuando se tiene una lista de cadenas que necesitan unirse en un único valor de cadena. El método `join()` se ejecuta sobre una cadena, se le pasa una lista de cadenas y devuelve una cadena. La cadena devuelta es la concatenación de cada cadena de la lista pasada. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> ', '.join(['GATOS', 'RATAS', 'MURCIÉLAGOS'])
gatos, ratas, murciélagos
>>> ' '.join(['Mi', 'nombre', 'ES', 'Simón'])
Me llamo Simon
>>> 'ABC'.join(['Mi', 'nombre', 'ES', 'Simón'])
'MiNombreABCesABCSimon'
```

Observe que la cadena a la que llama `join()` se inserta entre cada cadena del argumento de la lista. Por ejemplo, cuando `join(['gatos', 'ratas', 'murciélagos'])` se llama a la cadena ', ', la cadena devuelta es 'gatos, ratas, murciélagos'.

Recuerde que `join()` se llama a un valor de cadena y se le pasa un valor de lista. (El método `split()` hace lo contrario: Se llama a un valor de cadena y devuelve una lista de cadenas. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> 'Me LLAMO Simón'.split()
['Mi', 'nombre', 'es', 'Simón']
```

Por defecto, la cadena `''Me llamo Simón"` se divide siempre que se encuentran caracteres de espacio en blanco como el espacio, el tabulador o la nueva línea. Estos caracteres de espacio en blanco no se incluyen en las cadenas de la lista devuelta. Puede pasar una cadena delimitadora al método `split()` para especificar una cadena diferente sobre la que dividir. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> 'MYABCNAMEABCISABCSIMON'.split('ABC')
['Mi', 'nombre', 'es', 'Simón']
>>> 'Me LLAMO Simón'.split('m')
['Mi na', 'e es Si', 'on']
```

Un uso común de `split()` es dividir una cadena multilínea a lo largo de los caracteres de nueva línea. Introduzca lo siguiente en el shell interactivo:

```
>>> spam= '''Querida Alice,
¿Cómo has estado? Estoy bien. HAY
un recipiente en la nevera
```

que lleva la etiqueta "Experimento con leche".

Por favor, no lo beba.
Atentamente,
Bob'''
>>> **spam.split('\n')**
['Querida Alice', '¿Cómo has estado? Estoy ', 'Hay un recipiente en la nevera', 'que lleva la etiqueta "Experimento con leche"', '', Por favor, no te lo bebas', 'Atentamente', 'Bob'].

Pasando a split() el argumento '\n' podemos dividir la cadena multilínea almacenada en spam a lo largo de las nuevas líneas y devolver una lista en la que cada elemento corresponde a una línea de la cadena.

Justificar texto con rjust(), ljust() y center()

Los métodos de cadena `rjust()` y `ljust()` devuelven una versión rellenada de la cadena a la que se llama, con espacios insertados para justificar el texto. El primer argumento de ambos métodos es una longitud entera para la cadena justificada. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> 'Hola'.RJUST(10)
'      Hola'
>>> 'Hola'.RJUST(20)
'                Hola'
>>> 'Hola Mundo'.RJUST(20)
'           Hola Mundo
>>> 'Hola'.LJUST(10)
Hola.
```

`Hola'.rjust(10)` dice que queremos justificar a la derecha `'Hola'` en una cadena de longitud total 10. `'Hola'` tiene cinco caracteres, por lo que se añadirán cinco espacios a su izquierda, lo que nos dará una cadena de 10 caracteres con `'Hola'` justificado a la derecha.

Un segundo argumento opcional para `rjust()` y `ljust()` especificará un carácter de distinto de un carácter de espacio. Introduzca lo siguiente en el shell interactivo:

```
>>> 'Hola'.RJUST(20, '*')
'***************Hello'
>>> 'Hola'.LJUST(20, '-')
Hola.----------------------'
```

El método de cadena `center()` funciona como `ljust()` y `rjust()` pero centra el texto en lugar de justificarlo a izquierda o derecha. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> 'Hola'.center(20)
'          Hola          '
>>> 'Hola'.center(20, '=')
'=======Hello========'
```

Estos métodos son especialmente útiles cuando necesitas imprimir datos tabulares que tengan el espaciado correcto. Abre una nueva ventana del editor de archivos e introduce el siguiente código, guardándolo como *picnicTable.py*:

```
def printPicnic(itemsDict, leftWidth, rightWidth): print('PICNIC
    ITEMS'.center(leftWidth+ rightWidth, '-')) for k, v in
    itemsDict.items():
        print(k.ljust(anchuraizquierda, '.')+ str(v).rjust(anchuraderecha))
```

```
picnicItems= {'sandwiches': 4, 'manzanas': 12, 'tazas': 4, 'galletas': 8000}
printPicnic(picnicItems, 12, 5)
printPicnic(picnicItems, 20, 6)
```

En este programa, definimos un método printPicnic() que tomará un diccionario de información y utilizará center(), ljust(), y rjust() para mostrar esa información en un formato similar a una tabla perfectamente alineada.

El diccionario que pasaremos a printPicnic() es picnicItems. En picnicItems, tenemos 4 sandwiches, 12 manzanas, 4 tazas y 8000 galletas. Queremos organizar esta información en dos columnas, con el nombre del a la izquierda y la cantidad a la derecha.

Para ello, decidimos qué anchura queremos que tengan las columnas izquierda y derecha. Junto con nuestro diccionario, pasaremos estos valores a printPicnic().

printPicnic() recibe un diccionario, un leftWidth para la columna izquierda de una tabla y un rightWidth para la columna derecha. Imprime un título, PICNIC ITEMS, centrado sobre la tabla. A continuación, recorre el , imprimiendo cada par clave-valor en una línea con la clave justificada a la izquierda y rellenada con puntos, y el valor justificado a la derecha y rellenado con espacios.

Después de definir printPicnic(), definimos el diccionario picnicItems y llamamos a printPicnic() dos veces, pasándole anchos diferentes para las columnas izquierda y derecha de la tabla.

Al ejecutar este programa, los elementos del picnic se muestran dos veces. La primera vez, la columna de la izquierda tiene 12 caracteres de ancho y la de la derecha, 5 caracteres. La segunda vez tienen 20 y 6 caracteres, respectivamente.

```
--ARTÍCULOS DE PICNIC--
sándwiches..         4
manzanas.....12
tazas.....         4
galletas .....8000
-------PICNIC ITEMS------
   bocadillos.....           4
manzanas.....          12
            ...
 tazas.....              4
 galletas.....        8000
```

El uso de rjust(), ljust() y center() le permite asegurarse de que las cadenas están perfectamente alineadas, incluso si no está seguro de cuántos caracteres sus cadenas.

Eliminación de espacios en blanco con strip(), rstrip() y lstrip()

A veces es posible que desee eliminar los caracteres de espacio en blanco (espacio, tabulador y nueva línea) del izquierdo, derecho o de ambos lados de una cadena. El método `strip()` string devolverá una nueva cadena sin ningún carácter de espacio en blanco al principio o al final. Los métodos `lstrip()` y `rstrip()` eliminarán los caracteres de espacio en blanco de los extremos izquierdo y derecho, respectivamente. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam='          Hola Mundo      '
>>> spam.strip()
"Hola Mundo"
>>> spam.LSTRIP()
'Hola Mundo '
>>> spam.RSTRIP()
'      Hola Mundo'
```

Opcionalmente, un argumento de cadena especificará qué caracteres de los extremos deben eliminarse. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> spam='SPAMSPAMBACONSPAMEGGSSPAMSPAM'
>>> spam.strip('ampS')
'BaconSpamEggs'
```

Si se pasa a `strip()` el argumento `'ampS'`, se le indicará que elimine las apariciones de `a`, `m`, `p` y `S` mayúscula de los extremos de la cadena almacenada en `spam`. El orden de los caracteres en la cadena pasada a `strip()` no importa: `strip('ampS')` hará lo mismo que `strip('mapS')` o `strip('Spam')`.

Copiar y pegar cadenas con el módulo pyperclip

El módulo pyperclip tiene funciones copy() y paste() que pueden enviar y recibir texto del portapapeles de su ordenador. El envío de la salida de su programa al portapapeles hará que sea fácil de pegar a un correo electrónico, procesador de textos, o algún otro software.

Pyperclip no viene con Python. Para instalarlo, siga las instrucciones de instalación de módulos de terceros del Apéndice A. Después de instalar el módulo pyperclip, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar pyperclip
>>> pyperclip.copy('¡Hola mundo!')
>>> pyperclip.pegar()
"¡Hola mundo!
```

Por supuesto, si algo fuera de su programa cambia el contenido del portapapeles, la función paste() lo devolverá. Por ejemplo, si copiara esta frase el portapapeles y luego llamara a paste(), quedaría así:

```
>>> pyperclip.pegar()
Por ejemplo, si copiara esta frase en el portapapeles y luego llamara a
paste(), quedaría así:'
```

EJECUCIÓN DE SCRIPTS PYTHON FUERA DE IDLE

Hasta ahora, has estado ejecutando tus scripts Python usando el shell interactivo y el editor de archivos en IDLE. Sin embargo, no querrás pasar por la molestia de abrir IDLE y el script de Python cada vez que quieras ejecutar un script. Afortunadamente, hay atajos que puedes configurar para facilitar la ejecución de scripts de Python. Los pasos son ligeramente diferentes para Windows, OS X y Linux, pero cada uno se describe en el Apéndice B. Dirígete al Apéndice B para aprender cómo ejecutar tus scripts Python convenientemente y ser capaz de pasarles argumentos de línea de comandos. (No podrá pasar argumentos de línea de comandos a sus programas usando IDLE).

Proyecto Password Locker

Probablemente tengas cuentas en muchos sitios web diferentes. Es un mal hábito utilizar la misma contraseña para cada una de ellas, porque si alguno de esos sitios sufre un fallo de seguridad, los piratas informáticos conocerán la contraseña de todas tus demás cuentas. Lo mejor utilizar un software de gestión de contraseñas en tu ordenador que utilice una contraseña maestra para desbloquear el gestor de contraseñas. A continuación, puedes copiar la contraseña de cualquier cuenta en el portapapeles y pegarla en el campo Contraseña del sitio web.

El programa gestor de contraseñas que crearás en este ejemplo no es seguro, pero ofrece una demostración básica de cómo funcionan estos programas.

LOS PROYECTOS DEL CAPÍTULO

Este es el primer "proyecto de capítulo" del libro. A partir de aquí, cada capítulo tendrá proyectos que demuestren los conceptos tratados en el capítulo. Los proyectos están escritos en un estilo que te lleva desde una ventana en blanco del editor de archivos hasta un programa completo y funcional. Al igual que con los ejemplos interactivos de shell, no se limite a leer las secciones de los proyectos, ¡siga el proceso en su ordenador!

Paso 1: Diseño del programa y estructuras de datos

Desea poder ejecutar este programa con un argumento de línea de comandos que sea el nombre de la , por ejemplo, *correo electrónico* o *blog*. La contraseña de esa se copiará en el portapapeles para que el usuario pueda pegarla en un campo Contraseña. De esta forma, el usuario puede tener contraseñas largas y complicadas sin tener que memorizarlas.

Abre una nueva ventana del editor de archivos y guarda el programa como *pw.py*. Necesitas iniciar el programa con una línea **#!** (*shebang*) (ver Apéndice B) y también debes escribir un comentario que describa brevemente el programa. Como quieres asociar el nombre de cada cuenta con su contraseña, puedes almacenarlas como cadenas en un diccionario.
El diccionario será la estructura de datos que organizará los datos de tus cuentas y contraseñas. Haz que tu programa tenga el siguiente aspecto:

```
#! python3
# pw.py - Un programa inseguro para guardar contraseñas.

CONTRASEÑAS={'email': 'F7minlBDDuvMJuxESSKHFhTxFtjVB6',
             'blog': 'VmALvQyKAxiVH5G8v01if1MLZF3sdt',
             'luggage': '12345'}
```

Paso 2: Manejar los argumentos de la línea de comandos

Los argumentos de la línea de comandos se almacenarán en la variable sys.argv. (Consulte el Apéndice B para obtener más información sobre cómo utilizar argumentos de línea de comandos en sus programas). El primer elemento de la lista sys.argv debe ser siempre una cadena que contenga el nombre de archivo del programa ('pw.py'), y el segundo elemento debe ser el primer argumento de la línea de comandos. Para este programa, este argumento es el nombre de la cuenta cuya contraseña desea. Dado que el argumento de línea de comandos es obligatorio, muestra un mensaje de uso al usuario si se olvida de añadirlo (es decir, si la lista sys.argv tiene menos de dos valores en ella). Haz que tu programa tenga el siguiente aspecto:

```
#! python3
# pw.py - Un programa inseguro para guardar contraseñas.

CONTRASEÑAS= {'email': 'F7minlBDDuvMJuxESSKHFhTxFtjVB6',
              'blog': 'VmALvQyKAxiVH5G8v01if1MLZF3sdt',
              'luggage': '12345'}

importar SISTEMA
si len(SYS.argv)< 2:
    print('USO: python pw.py [cuenta] - copiar contraseña de cuenta') SYS.exit()

cuenta= SYS.argv[1]          # primera línea de comando arg ES el nombre de la cuenta
```

Paso 3: Copiar la contraseña correcta

Ahora que el nombre de la cuenta está almacenado como una cadena en la variable accountnecesitas ver si existe en el diccionario PASSWORDS como una clave. Si es así, quieres copiar el valor de la al portapapeles usando pyperclip.copy(). (Ya que estás usando el módulo pyperclip, necesitas importarlo.) Nota que en realidad no *necesitas* la variable account; podrías simplemente usar sys.argv[1] en todos los lugares donde se usa account en este programa. Pero una variable llamada account es mucho más legible que algo críptico como sys.argv[1].

Haz que tu programa tenga el siguiente aspecto:

```
#! python3
# pw.py – Un programa inseguro para guardar contraseñas.
CONTRASEÑAS= {'email': 'F7minlBDDuvMJuxESSKHFhTxFtjVB6',
              blog': 'VmALvQyKAxiVH5G8v01if1MLZF3sdt', 'luggage':
              '12345'}

import sys, pyperclip
si len(sys.argv)< 2:
    print('Uso: py pw.py [cuenta] - copiar contraseña de cuenta')
    sys.exit()

cuenta= sys.argv[1]          # primera línea de comando arg es el nombre de la cuenta

if cuenta in CONTRASEÑAS:
    pyperclip.copy(CONTRASEÑAS[cuenta])
    print('Contraseña de '+ cuenta+ ' copiada al portapapeles.')
else:
    print('No hay ninguna cuenta llamada '+ account)
```

Este nuevo código busca el nombre de cuenta en el diccionario PASSWORDS. Si el nombre de la cuenta es una clave del diccionario, obtenemos el valor correspondiente a esa clave, lo copiamos en el portapapeles e imprimimos un mensaje diciendo que hemos copiado el valor. En caso contrario, mostramos un mensaje diciendo que no hay ninguna cuenta con ese .

Ese es el script completo. Usando las instrucciones del Apéndice B para lanzar programas de línea de comandos fácilmente, ahora tiene una forma rápida de copiar las contraseñas de sus cuentas al portapapeles. Tendrás que modificar el valor del diccionario PASSWORDS en el código fuente cada vez que quieras actualizar el programa con una nueva contraseña.

Por supuesto, probablemente no quieras guardar todas tus contraseñas en un solo lugar

donde cualquiera podría copiarlos fácilmente. Pero puedes modificar este programa y para copiar rápidamente texto normal en el portapapeles. Digamos que usted está enviando varios correos electrónicos que tienen muchos de los mismos párrafos en común. Podrías poner cada párrafo como un valor en el diccionario PASSWORDS (probablemente renombrar el diccionario en este punto), y entonces tendrías una manera de rápidamente seleccionar y copiar uno de los muchos fragmentos de texto estándar en el portapapeles.

En Windows, puede crear un archivo por lotes para ejecutar este programa con la ventana WIN-R Run. (Para obtener más información sobre los archivos por lotes, consulte el Apéndice B.) Escriba lo siguiente en el editor de archivos y guarde el archivo como *pw.bat* en la carpeta *C:\Windows*:

```
@py.exe C:\Python34\pw.py %*
@pause
```

Con este archivo por lotes creado, ejecutar el programa de seguridad de contraseñas en Windows es sólo cuestión de pulsar WIN-R y escribir pw <nombre de cuenta>.

Proyecto: Añadir viñetas al marcado wiki

Al editar un artículo de Wikipedia, puedes crear una lista con viñetas colocando cada elemento de la lista en su propia línea y poniendo una estrella delante. Pero supongamos que tienes una lista muy larga a la que quieres añadir viñetas. Podrías simplemente escribir esas estrellas al principio de cada línea, una a una. O puedes automatizar esta tarea con un breve script de Python.

El script *bulletPointAdder.py* obtendrá el texto del portapapeles, añadirá una estrella y un espacio al principio de cada línea y, a continuación, pegará este nuevo texto en el portapapeles. Por ejemplo, si copiara el siguiente texto (para el artículo de Wikipedia "Lista de listas de listas") en el portapapeles:

```
Listas de animales
Listas de acuarios
Listas de biólogos por abreviatura de autor
Listas de cultivares
```

y luego ejecutó el programa *bulletPointAdder.py*, el portapapeles contendría entonces lo siguiente:

```
* Listas de animales
* Listas de acuarios
* Listas de biólogos por abreviatura de autor
* Listas de cultivares
```

Este texto con prefijo de estrella está listo para pegarse en un artículo de Wikipedia como lista con viñetas.

Paso 1: Copiar y pegar desde el portapapeles

Quieres que el programa *bulletPointAdder.py* haga lo siguiente:

1. Pegar texto desde el portapapeles
2. Hazle algo
3. Copiar el nuevo texto en el portapapeles

El segundo paso es un poco complicado, pero los pasos 1 y 3 son bastante sencillos: Sólo involucran las funciones pyperclip.copy() y pyperclip.paste(). Por ahora, vamos a escribir la parte del programa que cubre los pasos 1 y 3. Escribe lo siguiente, guardando el programa como *bulletPointAdder.py*:

```
#! python3
# bulletPointAdder.py - Añade viñetas Wikipedia al inicio # de cada
línea de texto en el portapapeles.

importar pyperclip
texto= pyperclip.paste()

# TODO: Separar líneas y añadir estrellas.

pyperclip.copy(text)
```

El comentario TODO es un recordatorio de que debe completar esta parte del programa eventualmente. El siguiente paso es implementar esa parte del programa.

Paso 2: Separar las líneas de texto y añadir la estrella

La llamada a pyperclip.paste() devuelve todo el texto del portapapeles como una gran cadena. Si utilizamos el ejemplo "Lista de listas de listas", la cadena almacenada en texto tendría este aspecto:

```
Listas de animales Listas de animales de acuario Listas de biólogos por abreviatura
de autor Listas de cultivares
```

Los caracteres de nueva línea en esta cadena hacen que se muestre con varias líneas cuando se imprime o se pega desde el portapapeles. Hay muchas "líneas" en este valor de cadena. Desea añadir una estrella al principio de cada una de estas líneas.

Podrías escribir código que busque cada carácter de nueva línea en la cadena y luego añada la estrella justo después. Pero sería más fácil usar el método split() para devolver una lista de cadenas, una por cada línea de la cadena original, y luego añadir la estrella al principio de cada cadena de la lista.

Haz que tu programa tenga el siguiente aspecto:

```
#! python3
# bulletPointAdder.py - Añade viñetas Wikipedia al inicio # de cada
línea de texto en el portapapeles.

importar pyperclip
texto= pyperclip.paste()

# Separa LAS LÍNEAS y añade ESTRELLAS.
LÍNEAS= text.split('\n')
for i in range(len(LINES)):          # recorre todos los ÍNDICES de la lista "LÍNEAS"
    LÍNEAS[i]= '* '+ LÍNEAS[i] # añade una estrella a cada cadena de la lista
    "LÍNEAS

pyperclip.copy(texto)
```

Dividimos el texto a lo largo de sus nuevas líneas para obtener una lista en la que cada elemento es una línea del texto. Almacenamos la lista en líneas y la recorremos en bucle. Para cada línea, añadimos una estrella y un espacio al principio de la línea. Ahora, cada cadena de las líneas comienza con una estrella.

Paso 3: Unir las líneas modificadas

La lista de líneas contiene ahora las líneas modificadas que empiezan por estrellas. Pero pyperclip.copy() espera un único valor de cadena, no una lista de valores de cadena. Para hacer este único valor de cadena, pasa las líneas al método join() para obtener una única cadena unida a partir de las cadenas de la lista. Haz que tu programa se parezca a lo siguiente:

```
#! python3
# bulletPointAdder.py - Añade viñetas Wikipedia al inicio # de cada
línea de texto en el portapapeles.

importar pyperclip
texto= pyperclip.paste()

# Separa las líneas y añade estrellas.
líneas= text.split('\n')
for i in range(len(lines)):       # bucle a través de todos los índices para
    "líneas" lista líneas[i]= '* '+ líneas[i] # añadir estrella a cada
    cadena en "líneas" lista
texto= '\n'.join(LÍNEAS)
pyperclip.copy(texto)
```

Cuando se ejecuta este programa, sustituye el texto del portapapeles por texto que tiene estrellas al principio de cada línea. Ahora el programa está completo, y puedes intentar ejecutarlo con el texto copiado en el portapapeles.

Aunque no necesites automatizar esta tarea específica, es posible que quieras automatizar algún otro tipo de manipulación de texto, como eliminar los espacios al final de las líneas o convertir el texto a mayúsculas o minúsculas. Sean cuales sean tus necesidades, puedes utilizar el portapapeles como entrada y salida.

Resumen

El texto es una forma común de datos, y Python viene con muchos métodos de cadena útiles para procesar el texto almacenado en valores de cadena. En casi todos los programas que escribas en Python utilizarás métodos de indexación, troceado y cadenas.

Los programas que está escribiendo ahora no parecen demasiado sofisticados-no tienen interfaces gráficas de usuario con imágenes y texto colorido. Hasta ahora, estás mostrando texto con `print()` y dejando que el usuario introduzca texto con `input()`.
Sin embargo, el usuario puede introducir rápidamente grandes cantidades de texto a través del portapapeles.
Esta capacidad proporciona una vía útil para escribir programas que manipulen cantidades masivas de texto. Puede que estos programas basados en texto no tengan ventanas o gráficos llamativos, pero pueden mucho trabajo útil rápidamente.

Otra forma de manipular grandes cantidades de texto es leer y escribir archivos directamente desde el disco duro. Aprenderás cómo hacerlo con Python en el próximo capítulo.

Esto cubre todos los conceptos básicos de la programación en Python. Continuarás aprendiendo nuevos conceptos a lo largo del resto de este libro, pero ahora ya sabes lo suficiente para empezar a escribir algunos programas útiles que pueden automatizar tareas. Puede que pienses que no tienes suficientes conocimientos de Python para hacer cosas como descargar páginas web, actualizar hojas de cálculo o enviar mensajes de texto, ¡pero ahí es donde entran los módulos de Python! Estos módulos, escritos por otros programadores, proporcionan funciones que te facilitan hacer todas estas cosas. Así que vamos a aprender a escribir programas reales para hacer tareas automatizadas útiles.

Preguntas prácticas

P: 1. ¿Qué son los caracteres de escape?

P: 2. ¿Qué representan los caracteres de escape \n y \t?

P: 3. ¿Cómo se puede poner un carácter de barra invertida \ en una cadena?

P: 4. El valor de cadena "Howl's Moving Castle" es una cadena válida. ¿Por qué no es un problema que el carácter de comilla simple de la palabra Howl's no esté escapado?

P: 5. Si no quieres poner \n en tu cadena, ¿cómo puedes escribir una cadena con nuevas líneas en ella?

P: 6. ¿A cuánto se evalúan las siguientes expresiones?

- '¡Hola mundo!'[1]
- '¡Hola mundo!'[0:5]
- '¡Hola mundo!'[:5]
- '¡Hola mundo!'[3:]

P: 7. ¿Qué evalúan las siguientes expresiones?

- Hola'.superior()
- 'Hola'.superior().isuper()
- 'Hola'.upper().lower()

P: 8. ¿A cuánto se evalúan las siguientes expresiones?

- "Recuerda, recuerda, el cinco de
- noviembre.".split() '-'.join('Sólo puede haber uno.'.split())

P: 9. ¿Qué métodos de cadena puedes usar para justificar a la derecha, a la izquierda y centrar una cadena?

P: 10. ¿Cómo se pueden recortar los espacios en blanco al principio o al final de una cadena?

Proyecto práctico

Para practicar, escribe un programa que haga lo siguiente.

Impresora de mesa

Escriba una función llamada printTable() que tome una lista de listas de cadenas y la muestre en una tabla bien organizada con cada columna justificada a la derecha. Asuma que todas las listas internas contendrán el mismo número de cadenas. Por ejemplo, el valor podría tener este aspecto:

```
tableData= [['manzanas', 'naranjas', 'cerezas', 'plátano'], ['Alice',
                'Bob', 'Carol', 'David'],
                ['perros', 'gatos', 'alces', 'gansos']]
```

Su función printTable() imprimiría lo siguiente:

```
manzanas Alicia
               perros
naranjas    Bob   gatos
cerezas Carol alces
   plátano David ganso
```

Pista: Tu código tendrá que encontrar primero la cadena más larga de cada una de las listas interiores para que toda la columna pueda ser lo suficientemente ancha como para que quepan todas las cadenas. Puede almacenar el ancho máximo de cada columna como una lista de enteros. La función printTable() puede comenzar con colWidths= [0] * len(tableData), que creará una lista que contiene el mismo número de valores 0 que el número de listas internas en tableData. De esta manera, colWidths[0] puede almacenar el ancho de la cadena más larga en tableData[0], colWidths[1] puede almacenar el ancho de la cadena más larga en tableData[1], y así sucesivamente. A continuación, puede encontrar el valor más grande en la lista colWidths para averiguar qué ancho entero pasar al método de cadena rjust().

Parte II. Automatización de tareas

Capítulo 7. Comparación de patrones con expresiones regulares Comparación de patrones con expresiones regulares

Es posible que esté familiarizado con la búsqueda de texto pulsando CTRL-F y escribiendo las palabras que busca. *Las expresiones regulares* van un paso más allá: especificar *un patrón* de texto para la búsqueda. Puede que no conozca el número de teléfono exacto de una empresa, pero si vive en Estados Unidos o Canadá, sabe que tendrá tres dígitos, seguidos de un guión y cuatro dígitos más (y, opcionalmente, un código de área de tres dígitos al principio). Así es como usted, como ser humano, reconoce un número de teléfono cuando lo ve: 415-555-1234 es un número de teléfono, pero 4.155.551.234 no lo es.

Las expresiones regulares son útiles, pero no muchos no programadores las conocen, a pesar de que la mayoría de los editores de texto y procesadores de texto modernos, como Microsoft Word u OpenOffice, disponen de funciones de búsqueda y búsqueda-y-reemplazo que permiten realizar búsquedas basadas en expresiones regulares. Las expresiones regulares ahorran mucho tiempo, no sólo a los usuarios de software, sino también a los programadores de . De hecho, Cory Doctorow, escritor especializado en tecnología, afirma que incluso antes de enseñar programación, deberíamos enseñar expresiones regulares:

> *"Conocer [las expresiones regulares] puede suponer la diferencia entre resolver un problema en tres pasos o en tres mil. Cuando eres un empollón, te olvidas de que los problemas que resuelves con un par de pulsaciones pueden llevar a otras personas días de trabajo tedioso y propenso a errores".([1])*

En este capítulo, empezarás escribiendo un programa para encontrar patrones de texto *sin* usar expresiones regulares y luego verás cómo usar expresiones regulares para hacer el código mucho menos hinchado. Te mostraré el emparejamiento básico con expresiones regulares y luego pasaré a algunas funciones más potentes, como la sustitución de cadenas y la creación de tus propias clases de caracteres. Por último, al final del capítulo, escribirás un programa que puede extraer automáticamente números de teléfono y direcciones de correo electrónico de un bloque de texto.

Búsqueda de patrones de texto sin expresiones regulares

Supongamos que quiere encontrar un número de teléfono en una cadena. Conoces el patrón: tres números, un guión, tres números, un guión y cuatro números. He aquí un ejemplo: 415-555-4242.

Vamos a utilizar una función llamada isPhoneNumber() para comprobar si una cadena coincide con este patrón, devolviendo True o False. Abra una nueva ventana del editor de archivos e introduzca el siguiente código; a continuación, guarde el archivo como *isPhoneNumber.py*:

```
def esNúmeroDeTeléfono(texto):
❶     si len(texto) != 12:
          devolver False
      para i en rango(0, 3):
❷         if not text[i].isdecimal():
              return False
➲     if text[3] != '-':
          return False
      para i en rango(4, 7):
❹         if not text[i].isdecimal():
              return False
❺     if text[7] != '-':
          return False
      para i en rango(8, 12):
❻         if not text[i].isdecimal():
              return False
❼     devolver True

print('415-555-4242 es un número de
teléfono:') print(isPhoneNumber('415-
555-4242')) print('Moshi moshi es un
número de teléfono:')
print(isPhoneNumber('Moshi
moshi'))
```

Cuando se ejecuta este programa, la salida tiene este aspecto:

```
415-555-4242 es un número de teléfono:
Verdadero
Moshi moshi es un número de teléfono:
Falso
```

La función isPhoneNumber() tiene un código que realiza varias comprobaciones para ver si la cadena de texto es un número de teléfono válido. Si alguna de estas comprobaciones falla, la función devuelve False. En primer lugar, el código comprueba que la cadena sea exactamente 12

caracteres ❶. A continuacióncomprueba que el prefijo (es decir, los tres primeros caracteres del texto) esté formado sólo por caracteres numéricos ❷. El resto de la función comprueba que la cadena sigue el patrón de un número de teléfono: El número debe tener el primer guión después del prefijo telefónico❸, tres caracteres numéricos más❹, luego otro guión ❺, y finalmente cuatro números más ❻. Si la ejecución del programa consigue superar todas las comprobaciones, devuelve True ❼.

Si se llama a isPhoneNumber() con el argumento '415-555-4242' se obtendrá True. Si se llama a isPhoneNumber() con el argumento "Moshi moshi", se obtiene False; la primera prueba falla porque "Moshi moshi" no tiene 12 caracteres.

Tendrías que añadir aún más código para encontrar este patrón de texto en una cadena más grande. Sustituya las cuatro últimas llamadas a la función print() en *isPhoneNumber.py* por lo siguiente:

```
mensaje= 'Llámame mañana al 415-555-1011. 415-555-9999 es mi oficina.' for i in
    range(len(message)):
❶       chunk= mensaje[i:i+12]
❷       si isPhoneNumber(chunk):
            print('Número de teléfono encontrado: '+ chunk)
    print('Hecho')
```

Cuando se ejecute este programa, la salida tendrá el siguiente aspecto:

```
Número de teléfono encontrado: 415-
555-1011 Número de teléfono
encontrado: 415-555-9999 Hecho
```

En cada iteración del bucle for, se asigna un nuevo trozo de 12 caracteres del mensaje a la variable trozo ❶. Por ejemplo, en la primera iteración, i es 0, y a chunk se le asigna message[0:12] (es decir, la cadena 'Llámame a las 4'). En la siguiente iteración, i es 1, y a chunk se le asigna mensaje[1:13] (la cadena 'llámame al 41').

Se pasa chunk a isPhoneNumber() para ver si coincide con el patrón de número de teléfono ❷, y si es así, se imprime el chunk.

Continúa el bucle a través del mensaje, y finalmente los 12 caracteres del trozo serán un número de teléfono. El bucle recorre toda la cadena, probando cada trozo de 12 caracteres e imprimiendo cualquier trozo que encuentre que satisfaga isPhoneNumber(). Una vez que hemos terminado de recorrer el mensaje, imprimimos Done.

Aunque la cadena del mensaje es corta en este ejemplo, podría tener millones de caracteres y el programa seguiría ejecutándose en menos de un segundo. Algo similar

que encuentra números de teléfono utilizando expresiones regulares también se ejecutaría en menos de un segundo, pero las expresiones regulares hacen que sea más rápido escribir estos programas.

Búsqueda de patrones de texto con expresiones regulares

El programa anterior de búsqueda de números de teléfono funciona, pero utiliza mucho código para hacer algo limitado: La función isPhoneNumber() tiene 17 líneas pero sólo puede encontrar un patrón de números de teléfono. ¿Qué pasa con un número de teléfono formateado como 415.555.4242 o (415) 555-4242? ¿Y si el número de teléfono tuviera una extensión, como 415-555-4242 x99? La función isPhoneNumber() fallaría al validarlos. Podría añadir más código para estos patrones adicionales, pero hay una manera más fácil.

Las expresiones regulares, llamadas *regexes* para abreviar, son descripciones de un patrón de texto. Por ejemplo, una \d en una regex significa un carácter de dígito, es decir, cualquier número del 0 al 9. La regex \d\d-\d\d\d\d-\d\d\d\d se utiliza para describir un patrón de texto. Python utiliza la expresión regular \d-\d-\d-\d-\d-\d-\d para que coincida con el mismo texto que la función anterior isPhoneNumber(): una cadena de tres números, un guión, tres números más, otro guión y cuatro números más.
números. Cualquier otra cadena no coincidiría con el \d\d\d-\d\d\d-\d\d\d regex.

Pero las expresiones regulares pueden ser mucho más sofisticadas. Por ejemplo, añadir un 3 entre llaves ({3}) después de un patrón es como decir: "Coincide con este patrón tres veces". Así, la expresión regular ligeramente más corta \d{3}-\d{3}-\d{4} también coincide con el formato correcto del número de teléfono.

Creación de objetos Regex

Todas las funciones regex de Python se encuentran en el módulo re. Introduzca lo siguiente en el intérprete de comandos interactivo para importar este módulo:

```
>>> importar re
```

NOTA

La mayoría de los ejemplos que siguen en este capítulo requerirán el módulo re, así que recuerda importarlo al principio de cualquier script que escribas o cada vez que reinicies IDLE. De lo contrario, obtendrá un mensaje de error NameError: name 're' is not defined.

Pasar un valor de cadena que represente su expresión regular a re.compile() devuelve un objeto de patrón Regex (o simplemente, un objeto Regex).

Para crear un objeto Regex que coincida con el patrón del número de teléfono, introduzca lo siguiente en el intérprete de comandos interactivo. (Recuerde que \d significa "un carácter de dígito" y \d\d\d-\d\d\d\d\d\d es la expresión regular para el patrón correcto del número de teléfono).

```
>>> phoneNumRegex= re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
```

Ahora la variable phoneNumRegex contiene un objeto Regex.

PASAR CADENAS SIN PROCESAR A RE.COMPILE( )

Recuerde que los caracteres de escape en Python utilizan la barra invertida (\). El valor de la cadena '\n' representa un único carácter de nueva línea, no una invertida seguida de una *n* minúscula. Así que '\\n' es la cadena que representa una barra invertida seguida de una *n* minúscula. Sin embargo, poniendo una r antes de la primera comilla del valor de la cadena, puede marcar la cadena como una *cadena sin procesar*, que no escapa caracteres.

Dado que las expresiones regulares utilizan frecuentemente barras invertidas, es conveniente pasar cadenas sin procesar a la función re.compile() en lugar de escribir barras invertidas adicionales. Escribir r'\d\d-\d\d\d-\d\d\d\d' es mucho más fácil que escribir '\d\d\d-\d\d\d\d\d\d\d\d\d\d\d'.

Comparación de objetos Regex

El método search() de un objeto Regex busca en la cadena que se le pasa cualquier coincidencia con el regex. El método search() devolverá None si el patrón regex no se encuentra en la cadena. Si *se* encuentra el patrón, el método search() devuelve un objeto Match. Los objetos Match tienen un método group() que devolverá el texto coincidente de la cadena buscada. (Explicaré los grupos en breve.) Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> phoneNumRegex= re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
>>> mo= phoneNumRegex.search('Mi número ES 415-555-4242.')
>>> print('Número de teléfono encontrado: '+ mo.group())
Número de teléfono encontrado: 415-555-4242
```

El nombre de la variable mo es sólo un nombre genérico a utilizar para los objetos Match. Este ejemplo puede parecer complicado al principio, pero es mucho más corto que el anterior programa *isPhoneNumber.py* y hace lo mismo.

Aquí, pasamos nuestro patrón deseado a re.compile() y almacenamos el objeto Regex resultante en phoneNumRegex. Luego llamamos a search() en phoneNumRegex y le pasamos a search() la cadena que queremos buscar para encontrar una coincidencia. El resultado de la búsqueda se almacena en la variable mo. En este ejemplo, sabemos que nuestro patrón se encontrará en la cadena, por lo que sabemos que se devolverá un objeto Match. Sabiendo que mo contiene un objeto Match y no el valor nulo None, podemos llamar a group() sobre mo para devolver la coincidencia. Escribiendo mo.group() dentro de nuestra sentencia print se muestra la coincidencia completa, 415-555-4242.

Revisión de la concordancia de expresiones regulares

Aunque hay varios pasos para utilizar expresiones regulares en Python, cada paso es bastante sencillo.

1. Importa el módulo regex con `import re`.
2. Crea un objeto `Regex` con la función `re.compile()`. (Recuerde utilizar una cadena sin procesar).
3. Pase la cadena que buscar en el objeto `Regex` `search()` método. Devuelve un objeto `Match`.
4. Llame al método `group()` del objeto `Match` para que devuelva una cadena con texto coincidente.

NOTA

Aunque te animo a que introduzcas el código de ejemplo en el intérprete de comandos interactivo, también deberías hacer uso de los comprobadores de expresiones regulares basados en la web, que pueden mostrarte exactamente cómo coincide una expresión regular con un fragmento de texto que introduzcas. Recomiendo el comprobador de http://regexpal.com/.

Más comparaciones de patrones con expresiones regulares

Ahora que conoces los pasos básicos para crear y encontrar objetos de expresiones regulares con Python, estás listo para probar algunas de sus capacidades más potentes de concordancia de patrones.

Agrupación con paréntesis

Digamos que desea separar el código de área del resto del número de teléfono. Si añade paréntesis, se crearán *grupos* en la expresión regular: (\d\d\d)-(\d\d\d-).\d\d\d\d). A continuación, puede utilizar el método de objeto coincidente group() para obtener el texto coincidente de un solo grupo.

El primer conjunto de paréntesis en una cadena regex será el grupo 1. El segundo conjunto será el grupo 2. Pasando el número entero 1 o 2 al método group() del objeto coincidente, puede obtener diferentes partes del texto coincidente. Si pasa 0 o nada al método group() obtendrá el texto completo. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> phoneNumRegex = re.compile(r'(\d\d\d)-(\d\d\d-\d\d\d)')
>>> mo= phoneNumRegex.search('Mi número ES 415-555-4242.')
>>> mo.grupo(1)
'415'
>>> mo.grupo(2)
'555-4242'
>>> mo.group(0)
'415-555-4242'
>>> mo.grupo()
'415-555-4242'
```

Si desea recuperar todos los grupos a la , utilice el método groups() - tenga en cuenta la forma plural del nombre.

```
>>> mo.GRUPOS()
('415', '555-4242')
>>> areaCode, mainNumber= mo.GROUPS()
>>> print(areaCode)
415
>>> print(numeroprincipal)
555-4242
```

Dado que mo.groups() devuelve una tupla de múltiples valores, puede utilizar el truco de asignación múltiple para asignar cada valor a una variable independiente, como en la línea anterior areaCode, mainNumber= mo.groups().

Los paréntesis tienen un significado especial en las expresiones regulares, pero ¿qué puede hacer si necesita que un texto coincida con un paréntesis? Por ejemplo, puede que los números de teléfono que intenta buscar tengan el prefijo entre paréntesis. En este , debe escapar los caracteres (y con una barra invertida. Introduzca

siguiente en el intérprete de comandos interactivo:

```
>>> phoneNumRegex= re.compile(r'(\(\d\d\d\)) (\d\d\d-\d\d\d)')
>>> mo= phoneNumRegex.search('Mi número de teléfono ES (415) 555-4242.')
>>> mo.grupo(1)
'(415)'
>>> mo.grupo(2)
'555-4242'
```

Los caracteres de escape(y) en la cadena sin procesar pasada a re.compile() coincidirán con caracteres de paréntesis reales.

Emparejar varios grupos con la tubería

El carácter| se denomina *tubería*. Puede utilizarlo en cualquier lugar donde quiera que coincida con una de muchas expresiones. Por ejemplo, la expresión regular r'Batman|Tina Fey' coincidirá tanto con 'Batman' como con 'Tina Fey'.

Cuando tanto Batman como Tina Fey aparecen en la cadena buscada, la primera aparición del texto coincidente será devuelta como objeto Coincidencia. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> heroRegex= re.compile (r'Batman|Tina Fey')
>>> mo1= heroRegex.search('Batman y Tina Fey.')
>>> mo1.group()
'Batman

>>> mo2= heroRegex.search('Tina Fey y Batman.')
>>> mo2.group()
Tina Fey
```

NOTA

Puede encontrar todas las *ocurrencias coincidentes con el método findall() que se discute en El método findall().*

También puede utilizar la tubería para hacer coincidir uno de varios patrones como parte de su regex. Por ejemplo, supongamos que queremos que coincida con cualquiera de las cadenas 'Batman', 'Batmóvil', "Batcóptero" y "Batbat". Como todas estas cadenas empiezan por Bat, sería bueno poder especificar ese prefijo una sola vez. Esto se puede hacer con paréntesis. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> batRegex = re.compile(r'Bat(man|mobile|copter|bat)')
>>> mo= batRegex.search('El Batmóvil perdió una rueda')
>>> mo.grupo()
Batmóvil
>>> mo.grupo(1)
móvil
```

La llamada al método mo.group() devuelve el texto coincidente completo 'Batmobile', mientras que mo.group(1) devuelve sólo la parte del texto coincidente dentro del primer grupo de paréntesis, 'mobile'. Utilizando el carácter pipa y agrupando paréntesis, puede especificar varios patrones alternativos con los que desea que coincida la expresión regular.

Si necesita que coincida con un carácter de tubería real, escápelo con una barra invertida, como \|.

Emparejamiento opcional con el signo de interrogación

A veces hay un patrón que se desea que coincida sólo opcionalmente. Es decir, la expresión regular debe encontrar una coincidencia tanto si ese fragmento de texto está ahí como si no. El carácter ? marca el grupo que le precede como parte opcional del patrón. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> batRegex= re.compile(r'Bat(wo)?man')
>>> mo1= batRegex.search('Las AVENTURAS de Batman')
>>> mo1.group()
''Batman

>>> mo2= batRegex.search('Las AVENTURAS de Batwoman')
>>> mo2.group()
Batwoman
```

La parte (wo)? de la expresión regular significa que el patrón wo es un grupo opcional. La expresión regular coincidirá con texto que contenga cero o una instancia de *wo*. Por eso la expresión regular coincide tanto con ''Batwoman'' como con ''Batman''.

Utilizando el ejemplo del número de teléfono anterior, puede hacer que la expresión regular busque números de teléfono que tengan o no código de área. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> phoneRegex = re.compile(r'(\d\d\d-)?\d\d\d-\d\d\d')
>>> mo1= phoneRegex.search('Mi número ES 415-555-4242')
>>> mo1.group()
'415-555-4242'

>>> mo2= phoneRegex.search('Mi número ES 555-4242')
>>> mo2.group()
'555-4242'
```

Puedes pensar en la ? como si dijera: "Empareja el cero o uno del grupo que precede a este signo de interrogación".

Si necesita que coincida con un signo de real, escápelo con?

Emparejar cero o más con la estrella

El * (llamado *estrella* o *asterisco*) significa "coincidencia cero o más": el grupo que precede a la estrella puede aparecer cualquier número de veces en el texto. Puede estar completamente ausente o repetirse una y otra . Veamos de nuevo el ejemplo de Batman.

```
>>> batRegex= re.compile(r'Bat(wo)*man')
>>> mo1= batRegex.search('Las AVENTURAS de Batman')
>>> mo1.group()
'Batman'

>>> mo2= batRegex.search('Las AVENTURAS de Batwoman')
>>> mo2.group()
'Batwoman'

>>> mo3= batRegex.search('Las AVENTURAS de Batwowowowoman')
>>> mo3.group()
'Batwowowowoman'
```

Para "Batman", la parte (wo)* de la expresión regular coincide con cero casos de wo en la cadena; para "Batwoman", (wo)* coincide con un caso de wo; y para "Batwowowowowoman", (wo)* coincide con cuatro casos de wo.

Si necesita que coincida con un carácter de estrella real, anteponga a la estrella en la expresión regular una barra invertida, *.

Combinar uno o más con el Plus

Mientras que * significa "coincide con cero o más",+ (o *más*) significa "coincide con uno o más". A diferencia de la estrella, que no requiere que su grupo aparezca en la cadena coincidente, el grupo que precede a un signo más debe aparecer *al menos una vez*. No es opcional.
Introduzca lo siguiente en el intérprete de comandos interactivo y compárelo con las expresiones regulares en estrella de la sección anterior:

```
>>> batRegex= re.compile(r'Bat(wo)+man')
>>> mo1= batRegex.search('Las AVENTURAS de Batwoman')
>>> mo1.group()
Batwoman

>>> mo2= batRegex.search('Las AVENTURAS de Batwowowowoman')
>>> mo2.group()
Batwowowowoman

>>> mo3= batRegex.search('Las AVENTURAS de Batman')
>>> mo3== Ninguno
Verdadero
```

La expresión regular Bat(wo)+hombre no coincidirá con la cadena "Las aventuras de Batman"" porque el signo más requiere al menos una wo.

Si necesita que coincida con un carácter de signo más real, anteponga al signo más una barra invertida para escaparlo: \+.

Emparejar repeticiones específicas con paréntesis rizados

Si tiene un grupo que desea repetir un número determinado de veces, escriba a continuación del grupo un número entre llaves. Por ejemplo, la expresión regular (Ha){3} coincidirá con la cadena "HaHaHa", pero no con "HaHa", ya que esta última sólo tiene dos repeticiones del grupo (Ha).

En lugar de un número, puede especificar un intervalo escribiendo un mínimo, una coma y un máximo entre las llaves. Por ejemplo, el regex (Ha){3,5} coincidirá con 'HaHaHa', 'HaHaHaHa' y 'HaHaHaHaHa'.

También puede omitir el primer o segundo número de las llaves para dejar el mínimo o el máximo sin límites. Por ejemplo, (Ha){3,} coincidirá con tres o más instancias del grupo (Ha), mientras que (Ha){,5} coincidirá con cero a cinco instancias. Las llaves pueden ayudar a acortar las expresiones regulares. Estas dos expresiones regulares coinciden con patrones idénticos:

```
(Ha){3}
(Ha)(Ha)(Ha)
```

Y estas dos expresiones regulares también coinciden con patrones idénticos:

```
(Ha){3,5} ((Ha)(Ha)(Ha))|((Ha)(Ha)(Ha)(Ha))|((Ha)(Ha)(Ha)(Ha)))
```

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> haRegex = re.compile(r'(Ha){3}')
>>> mo1= haRegex.search('HaHaHa')
>>> mo1.group()
'HaHaHa'

>>> mo2= haRegex.search('Ha')
>>> mo2== Ninguno
Verdadero
```

Aquí, (Ha){3} coincide con 'HaHaHa' pero no con 'Ha'. Como no coincide con 'Ha', search() devuelve None.

Emparejamiento codicioso y no codicioso

Dado que (Ha){3,5} puede coincidir con tres, cuatro o cinco instancias de Ha en la cadena 'HaHaHaHaHa', puede preguntarse por qué la llamada del objeto Match a group() en el ejemplo anterior de llaves devuelve 'HaHaHaHaHa' en lugar de las posibilidades más cortas. Después de todo, 'HaHaHa' y 'HaHaHaHa' también son coincidencias válidas de la expresión regular (Ha){3,5}.

Las expresiones regulares de Python son *codiciosas* por defecto, lo que significa que en situaciones ambiguas coincidirán con la cadena más larga posible. La versión *no codiciosa* de las llaves, que coincide con la cadena más corta posible, tiene la llave de cierre seguida de un signo de interrogación.

Introduzca lo siguiente en el intérprete de comandos interactivo y observe la diferencia entre las formas codiciosas y no codiciosas de las llaves que buscan en la misma cadena:

```
>>> greedyHaRegex = re.compile(r'(Ha){3,5}')
>>> mo1= greedyHaRegex.search('HaHaHaHaHa')
>>> mo1.group()
'HaHaHaHaHa'

>>> nongreedyHaRegex = re.compile(r'(Ha){3,5}?')
>>> mo2= nongreedyHaRegex.search('HaHaHaHaHa')
>>> mo2.group()
'HaHaHa'
```

Tenga en cuenta que el signo de interrogación puede tener dos significados en las expresiones regulares: declarar una coincidencia no buscada o marcar un grupo opcional. Estos significados no tienen nada que ver.

El método findall()

Además del método search(), los objetos Regex también tienen un método **findall()**. Mientras que search() devolverá un objeto Match del *primer* texto coincidente en la cadena buscada, el método **findall()** devolverá las cadenas de *cada* coincidencia en la cadena buscada. Para ver cómo search() devuelve un objeto Match sólo en la primera instancia de texto coincidente, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> phoneNumRegex= re.compile(r'\d\d\d-\d\d\d-\d\d\d')
>>> mo= phoneNumRegex.search('Móvil: 415-555-9999 Trabajo: 212-555-0000')
>>> mo.grupo()
'415-555-9999'
```

Por otro lado, findall() no devolverá un objeto Match sino una lista de cadenas -siempre *que no haya grupos en la expresión regular*. Cada cadena de la lista es un fragmento del texto buscado que coincide con la expresión regular. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> phoneNumRegex= re.compile(r'\d\d\d-\d\d\d-\d\d\d') # no TIENE GRUPOS
>>> phoneNumRegex.findall('Móvil: 415-555-9999 Trabajo: 212-555-0000')
['415-555-9999',    '212-555-0000']
```

Si *hay* grupos en la expresión regular**findall()** devolverá una lista de tuplas. Cada tupla representa una coincidencia encontrada, y sus elementos son las cadenas coincidentes para cada grupo en la expresión regular. Para ver **findall()** en acción, introduzca lo siguiente en el intérprete de comandos interactivo (observe que la expresión regular que se está compilando ahora tiene grupos entre paréntesis):

```
>>> phoneNumRegex= re.compile(r'(\d\d\d)-(\d\d\d)-(\d\d\d)') # TIENE GRUPOS
>>> phoneNumRegex.findall('Móvil: 415-555-9999 Trabajo: 212-555-0000')
[('415', '555', '1122'), ('212', '555', '0000')]
```

Para resumir lo que devuelve el método findall(), recuerde lo siguiente:

1. Cuando se llama a una expresión regular sin grupos, como \d\d\d-\d\d\d-\d\d\d, el método findall() devuelve una lista de coincidencias de cadenas, como por ejemplo ['415-555-9999', '212-555-0000'].
2. Cuando se llama a una expresión regular que tiene grupos, como (\d\d\d)-(\d\d\d)- (\d\ d\d\d), el método findall() devuelve una lista de tuplas de cadenas (una cadena por cada grupo), como **[('415', '555', '1122'), ('212',**
   '555', '0000')].

Clases de personajes

En el ejemplo anterior, aprendió que \d puede representar cualquier dígito numérico. Es decir, \d es la abreviatura de la expresión regular (0|1|2|3|4|5|6|7|8|9). Hay muchas *clases de caracteres abreviados*, como se muestra en la Tabla 7-1.

Tabla 7-1. Códigos abreviados para clases de caracteres comunes

Clase de caracteres abreviados	Representa
\d	Cualquier dígito numérico de 0 a 9.
\D	Cualquier carácter que *no sea* un dígito numérico del 0 al 9.
\w	Cualquier letra, dígito numérico o guión bajo. (Piense que se trata de caracteres de "palabra" coincidentes).
\W	Cualquier carácter que *no sea* una letra, un dígito numérico o el carácter de subrayado.
\s	Cualquier carácter de espacio, tabulación o nueva línea. (Piense en esto como una coincidencia de caracteres de "espacio").
\S	Cualquier carácter que *no sea* un espacio, un tabulador o una nueva línea.

Las clases de caracteres son útiles para acortar las expresiones regulares. La clase de caracteres [0-5] sólo coincidirá con los números del 0 al 5; esto es mucho más corto que escribir (0|1|2|3|4|5).

Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> xmasRegex = re.compile(r'\d+\s\w+')
>>> xmasRegex.findall('12 tamborileros, 11 gaiteros, 10 señores, 9 damas, 8 doncellas, 7 cisnes, 6 ocas, 5 anillas, 4 pájaros, 3 gallinas, 2 palomas, 1 perdiz")
['12 tamborileros', '11 gaiteros', '10 señores', '9 señoras', '8 doncellas', '7 cisnes', '6 gansos", 5 anillas", "4 pájaros", "3 gallinas", "2 palomas", "1 perdiz"].
```

La expresión regular \d+\s\w+ coincidirá con texto que tenga uno o más dígitos numéricos (\d+), seguido de un carácter de espacio en blanco (\s), seguido de uno o más caracteres de letra/dígito/subrayado (\w+). El método `findall()` devuelve todas las cadenas coincidentes del patrón regex en una lista.

Crea tus propias clases de personajes

Hay ocasiones en las que se desea emparejar un conjunto de caracteres pero las clases abreviadas de caracteres (\d, \w, \s, etc.) son demasiado amplias. Puede definir su propia clase de caracteres utilizando corchetes. Por ejemplo, la clase de caracteres [aeiouAEIOU] coincidirá con cualquier vocal, tanto en minúsculas como en mayúsculas. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> vowelRegex= re.compile(r'[aeiouAEIOU]')
>>> vowelRegex.findall('RoboCop COME comida para bebés. COMIDA PARA BEBÉS.')
['o', 'o', 'o', 'e', 'a', 'a', 'o', 'o', 'A', 'O', 'O']
```

También puede incluir rangos de letras o números utilizando un guión. Por ejemplo, la clase de caracteres [a-zA-Z0-9] coincidirá con todas las minúsculas, mayúsculas y números.

Tenga en cuenta que dentro de los , los símbolos normales de expresión regular no se interpretan como tales. Esto significa que no es necesario escapar los caracteres ., *, ? o () con una barra invertida precedente. Por ejemplo, la clase de caracteres [0-5.] coincidirá con los dígitos 0 a 5 y un punto. No es necesario escribirlo como [0-5\.].

Colocando un carácter de intercalación (^) justo después del corchete de apertura de la clase de caracteres, puede crear una *clase de caracteres negativa*. Una clase de caracteres negativa coincidirá con todos los caracteres que *no* estén en la clase de caracteres. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> consonantRegex = re.compile(r'[^aeiouAEIOU]')
>>> CONSONANTREGEX.findall('RoboCop COME comida para bebés. COMIDA PARA BEBÉS.')
['R', b', 'c', 'p', ' ', 't', 's', ' ', 'b', 'b', 'y', ' ', 'f', 'd', '.', ' ',
'B', 'B', 'Y', ' , 'F', 'D', '.']
```

Ahora, en lugar de coincidir con todas las vocales, coincidimos con todos los caracteres que no son vocales.

Caracteres del signo de intercalación y del signo de dólar

También puede utilizar el símbolo de intercalación (^) al principio de una expresión regular para indicar que la coincidencia debe producirse *al principio* del texto buscado. Del mismo modo, puede poner un signo de dólar ($) al final de la expresión regular para indicar que la cadena debe *terminar* con este patrón. Y puede utilizar^ y$ juntos para indicar que toda la cadena debe coincidir con el regex, es decir, no basta con que se produzca una coincidencia en algún subconjunto de la cadena.

Por ejemplo, la cadena de expresión regular r'^Hola' coincide con cadenas que empiecen por 'Hola'. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> beginsWithHello= re.compile(r'^Hello')
>>> beginsWithHello.search("¡Hola mundo!')
<_sre.SRE_Match object; span=(0, 5), match='Hola'>
>>> beginsWithHello.search("Dijo hola.')== None
Verdadero
```

La cadena de expresión regular r'\d$' coincide con cadenas que terminan con un carácter numérico del 0 al 9. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> endsWithNumber = re.compile(r'\d$')
>>> ENDSWITHNUMBER.search('Tu número ES 42')
<_sre.SRE_Match object; span=(16, 17), match='2'>
>>> endsWithNumber.search("Su número ES cuarenta y dos.')== None
Verdadero
```

La cadena de expresión regular r'^\d+$' coincide con cadenas que comienzan y terminan con uno o más caracteres numéricos. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> wholeStringIsNum= re.compile(r'^\d+$')
>>> WHOLESTRINGISNUM.search('1234567890')
<_sre.SRE_Match object; span=(0, 10), match='1234567890'>
>>> WHOLESTRINGISNUM.search('12345xyz67890')== None
Verdadero
>>> WHOLESTRINGISNUM.search('12 34567890')== None
Verdadero
```

Las dos últimas llamadas a search() en el ejemplo de shell interactivo anterior demuestran cómo toda la cadena debe coincidir con la expresión regular si se utilizan^ y$.

Siempre confundo los significados de estos dos símbolos, así que utilizo la mnemotecnia "Las zanahorias cuestan dólares" para recordarme que el signo de la zanahoria va primero y el del dólar después.

El personaje comodín

El carácter `.` (o *punto*) en una expresión regular se denomina *comodín* y coincide con cualquier carácter excepto con una nueva línea. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> atRegex= re.compile(r'.at')
>>> atRegex.findall('El gato del sombrero se sentó en la alfombrilla plana.')
['cat', 'hat', 'sat', 'lat', 'mat']
```

Recuerde que el carácter punto sólo coincide con un carácter, por lo que la coincidencia del texto `plano` del ejemplo anterior sólo coincide con `lat`. Para que coincida con un punto real, escape el punto con una barra invertida: `\.`.

Todo a juego con Dot-Star

A veces querrá que coincida todo y cualquier cosa. Por ejemplo, supongamos que desea buscar la cadena "Nombre:", seguida de cualquier texto, seguida de "Apellidos:" y, a continuación, seguida cualquier cosa. Puede utilizar el punto-estrella (.*) para sustituir a ese "cualquier cosa". Recuerde que el carácter punto significa "cualquier carácter simple excepto la nueva línea", y el carácter estrella significa "cero o más del carácter precedente".

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> nombreRegex= re.compile(r'Nombre: (.*) : (.*)')
>>> mo= nombreRegex.search(': Al Apellido: Sweigart')
>>> mo.grupo(1)
Al
>>> mo.grupo(2)
Sweigart
```

El dot-star utiliza el modo *codicioso*: Siempre intentará buscar la mayor cantidad de texto posible. Para hacer coincidir cualquier texto de forma *no codiciosa*, utilice el punto, la estrella y el signo de interrogación (.*?). Al igual que con las llaves, el signo de interrogación le indica a Python que no busque coincidencias con avaricia.

Introduzca lo siguiente en el intérprete de comandos interactivo para ver la diferencia entre las versiones codiciosa y no codiciosa:

```
>>> nongreedyRegex = re.compile(r'<.*?>')
>>> mo= nongreedyRegex.search('<Servir al hombre> para cenar.>')
>>> mo.grupo()
'<Para servir al hombre '>

>>> greedyRegex= re.compile(r'<.*>')
>>> mo= greedyRegex.search('<Servir al hombre> para cenar.>')
>>> mo.grupo()
'<Servir al hombre> para cenar.>'
```

Ambos regexes se traducen aproximadamente como "Coincide con un paréntesis angular de apertura, seguido de cualquier cosa, seguido de un paréntesis angular de cierre". Pero la cadena '<Servir al hombre> para cenar.>' tiene dos posibles coincidencias para el paréntesis angular de cierre. En la versión no codiciosa de la expresión regular, Python busca la cadena más corta posible: '<Para servir al hombre> '. En la versión codiciosa, Python busca la cadena más larga posible
cadena: '<Servir al hombre> para cenar.>'.

Coincidencia de nuevas líneas con el carácter de punto

El punto-estrella coincidirá con todo excepto con una nueva línea. Pasando re.DOTALL como segundo argumento a re.compile(), puede hacer que el carácter punto coincida *con todos los* caracteres, incluido el carácter de nueva línea.

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> noNewlineRegex = re.compile('.*')
>>> noNewlineRegex.search('Servir a la confianza pública.\nProteger a los inocentes.
\Cumplir la ").group()
"Servir a la confianza pública".

>>> newlineRegex= re.compile('.*', re.DOTALL)
>>> newlineRegex.search('Servir a la confianza pública.\nProteger a los inocentes.
\Cumplir la ").group()
Servir a la confianza pública. Proteger a los inocentes. Respetar la ley.
```

El regex noNewlineRegex, al que no se pasó re.DOTALL a la llamada re.compile() que lo creó, coincidirá con todo sólo hasta el primer carácter de nueva línea, mientras que newlineRegex, al que *sí* se le pasó re.DOTALL a re.compile(), coincide con todo. Por eso la llamada a newlineRegex.search() coincide con toda la cadena, incluidos los caracteres de nueva línea.

Revisión de los símbolos Regex

En este capítulo se ha tratado mucha notación, así que aquí tienes un repaso rápido de lo que has aprendido:

- La ? coincide con cero o uno de los grupos anteriores.
- El * coincide con cero o más del grupo anterior.
- La dirección+ coincide con uno o varios de los grupos anteriores.
- El {n} coincide exactamente con *n* del grupo anterior.
- El {n,} coincide con *n* o más del grupo anterior.
- El {,m} coincide con 0 a *m* del grupo anterior.
- El {n,m} coincide con al menos *n* y como máximo *m* del grupo anterior.
- {n,m}? o *? o +? efectúa una coincidencia no libre del grupo precedente.
- ^spam significa que la cadena debe empezar por *spam*.
- spam$ significa que la cadena debe terminar con *spam*.
- . coincide con cualquier carácter, excepto con caracteres de nueva línea.
- \d, \w y \s coinciden con un dígito, una palabra o espacio, respectivamente.
- \D, \W y \S coinciden con cualquier cosa excepto dígito, palabra o espacio, respectivamente.
- [abc] coincide con cualquier carácter entre los corchetes (como *a*, *b* o *c*).
- [^abc] coincide con cualquier carácter que no esté entre los corchetes.

Coincidencia sensible a mayúsculas y minúsculas

Normalmente, las expresiones regulares coinciden con el texto que se especifica. Por ejemplo, las siguientes expresiones regulares coinciden con cadenas completamente diferentes:

```
>>> regex1= re.compile('RoboCop')
>>> regex2= re.compile('ROBOCOP')
>>> regex3= re.compile('robOcop')
>>> regex4= re.compile('RobocOp')
```

Pero a veces sólo le interesa que coincidan las letras sin preocuparse de si están en mayúsculas o en minúsculas. Para que la expresión regular no distinga entre mayúsculas y minúsculas, puede pasar re.IGNORECASE o re.I como segundo argumento a re.compile(). Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> robocop= re.compile(r'robocop', re.I)
>>> robocop.search('RoboCop ES parte hombre, parte máquina, todo policía.').group()
RoboCop

>>> robocop.search('ROBOCOP PROTEGE A los inocentes.').group()
ROBOCOP

>>> robocop.search('Al, ¿por QUÉ tu libro de programación habla tanto de robocop?').group()
robocop
```

Sustitución de cadenas con el método sub()

Las expresiones regulares no sólo pueden encontrar patrones texto, sino que también pueden sustituirlos por texto nuevo. Al método sub() de los objetos Regex se le pasan dos argumentos. El primer argumento es una cadena para reemplazar cualquier coincidencia. El segundo es la cadena de la expresión regular. El método sub() devuelve una cadena con las sustituciones aplicadas.

Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> namesRegex= re.compile(r'Agente')
>>> namesRegex.sub('CENSURADO', 'La agente Alice entregó los documentos secretos al agente Bob.')
'CENSORED entregó los documentos secretos a CENSORED'.
```

A veces puede que necesite utilizar el propio texto coincidente como parte de la sustitución. En el primer argumento de sub(), puede escribir \1, \2, \3, etc., para significar "Introduzca el texto del grupo 1, 2, 3, etc., en la ".

Por ejemplo, supongamos que desea censurar los nombres de los agentes secretos mostrando sólo las primeras letras de sus nombres. Para ello, podría utilizar el agente regex `(\w)\w*` y pasar `r'\1****'` como primer argumento a sub(). `\1` de esa cadena se sustituirá por el texto que coincida con el grupo 1, es decir, el grupo `(\w)` de la expresión regular.

```
>>> agentNamesRegex= re.compile(r'Agente (\w)\w*')
>>> agentNamesRegex.sub(r'\1****', 'La agente Alice le dijo a la agente Carol que la agente Eve sabía que el agente Bob era un agente doble.')
A**** le dijo a C**** que E**** sabía que B**** era un agente doble.'
```

Gestión de expresiones regulares complejas

Las expresiones regulares están bien si el patrón de texto con el que debe coincidir es sencillo. Pero para encontrar patrones de texto complicados puede ser necesario expresiones regulares largas y complicadas. Esto se puede mitigar indicando a la función re.compile() que ignore los espacios en blanco y los comentarios dentro de la cadena de expresión regular. Este "modo detallado" puede activarse pasando la variable re.VERBOSE como segundo argumento a re.compile().

Ahora en lugar de una expresión regular difícil de leer como esta:

```
phoneRegex = re.compile(r'((\d{3}| \(\d{3}\))?(\s| -|\.)?\d{3}(\s|-|\.)\d{4}
(\s*(ext|x|ext.)\s*\d{2,5})?)')
```

puede distribuir la expresión regular en varias líneas con comentarios como éste:

```
phoneRegex= re.compile(r'''( (\d{3}|\(\d{3}\))?
                                    # código de
    área
    (\s|-|\.)?                      # separator
    \d{3}                           # 3 primeros dígitos
    (\s|-|\.)                       # separador
    \d{4}                           # últimos 4 dígitos
    (\s*(ext|x|ext.)\s*\d{2,5})?    # extensión
    )''', re.VERBOSE)
```

Observe cómo el ejemplo anterior utiliza la sintaxis de comillas triples (''') para crear una cadena de varias líneas, de forma que pueda extender la definición de la expresión regular a lo largo de muchas líneas, haciéndola mucho más legible.

Las reglas de comentario dentro de la cadena de expresión regular son las mismas que las del código regular de Python: El símbolo # y todo lo que sigue hasta el final de la línea son ignorados. Además, los espacios adicionales dentro de la cadena multilínea de la expresión regular no se consideran parte del patrón de texto que se va a comparar. Esto le permite organizar la expresión regular para que sea más fácil de leer.

Combinación de re.IGNORECASE, re.DOTALL y re.VERBOSE

¿Qué pasa si quiere utilizar re.VERBOSE para escribir comentarios en su expresión regular pero también quiere utilizar re.IGNORECASE para ignorar las mayúsculas? Por desgracia, la función re.compile() sólo toma un único valor como segundo argumento. Puede evitar esta limitación combinando la función re.IGNORECASE, re.DOTALL, y re.VERBOSE utilizando el carácter pipa (|), que en este contexto se conoce como el operador *bitwise o*.

Por lo tanto, si desea una expresión regular que no distinga entre mayúsculas y minúsculas *y que* incluya nuevas líneas para que coincida con el carácter de punto, formule la llamada a re.compile() de la manera:

```
>>> someRegexValue= re.compile('foo', re.IGNORECASE| re.DOTALL)
```

Las tres opciones para el segundo argumento tendrán este aspecto:

```
>>> someRegexValue= re.compile('foo', re.IGNORECASE| re.DOTALL| re.VERBOSE)
```

Esta sintaxis es un poco anticuada y tiene su origen en las primeras versiones de Python. Los detalles de los operadores bit a bit están fuera del alcance de este libro, pero echa un vistazo a los recursos en *http://nostarch.com/automatestuff/* para obtener más información.
También puedes pasar otras opciones para el segundo argumento; son poco comunes, pero también puedes leer más sobre ellas en los recursos.

Proyecto: Extractor de números de teléfono y direcciones de correo electrónico

Supongamos que tienes la aburrida tarea de encontrar todos los números de teléfono y direcciones de correo electrónico en una larga página web o documento. Si te desplazas manualmente por la página, puedes acabar buscando durante mucho tiempo. Pero si tuvieras un programa que pudiera buscar números de teléfono y direcciones de correo electrónico en el texto del portapapeles, sólo tendrías que pulsar CTRL-A para seleccionar todo el texto, pulsar CTRL-C para copiarlo en el portapapeles y ejecutar el programa. El programa podría sustituir el texto del portapapeles por los números de teléfono y las direcciones de correo electrónico que encontrara.

A la hora de abordar un nuevo proyecto, puede resultar tentador lanzarse a escribir código. Pero la mayoría de las veces, lo mejor es dar un paso atrás y tener en cuenta el panorama general. Te recomiendo que primero elabores un plan de alto nivel sobre lo que debe hacer tu programa. No pienses todavía en el código en sí: de eso ya te ocuparás más adelante. De momento, cíñete a grandes rasgos.

Por ejemplo, el extractor de direcciones de teléfono y correo electrónico deberá hacer lo siguiente:

- Obtener el texto del portapapeles.
- Encuentra todos los números de teléfono y direcciones de correo electrónico en el texto.
- Pégalos en el portapapeles.

Ahora puedes empezar a pensar en cómo podría funcionar esto en código. El código tendrá que hacer lo siguiente:

- Utiliza el módulo pyperclip para copiar y pegar cadenas.
- Cree dos expresiones regulares, una para los números de teléfono y otra para las direcciones de correo electrónico.
- Busca todas las , no sólo la primera, de ambas expresiones regulares.
- Formatea limpiamente las cadenas coincidentes en una sola cadena para pegar.
- Mostrar algún tipo de mensaje si no se han encontrado coincidencias en el texto.

Esta lista es como una hoja de ruta para el proyecto. Mientras escribes el código, puedes centrarte en cada uno de estos pasos por separado. Cada paso es bastante manejable y está expresado en términos de cosas que ya sabes hacer en Python.

Paso 1: Crear una Regex para números de teléfono

En primer lugar, tienes que crear una expresión regular para buscar números de teléfono. Crea un nuevo archivo, introduce lo siguiente y guárdalo *como teléfonoycorreo.py*:

```
#! python3
# phoneAndEmail.py - Busca números de teléfono y direcciones de correo electrónico en el

portapapeles. import pyperclip, re

phoneRegex      =      re.compile(r'''(
      (\d{3}|\(\d{3}\))?                  # código de área
      (\s|-|\.)?                          # separator
      (\d{3})                             # 3 primeros dígitos
      (\s|-|\.)                           # separador
      (\d{4})                             # últimos 4 dígitos
      (\s*(ext|x|ext.)\s*(\d{2,5}))?      # Extensión
      )''',  re.VERBOSE)

# TODO: Crear regex de email.

# TODO: Buscar coincidencias en el texto del

portapapeles. # TODO: Copiar resultados al

portapapeles.
```

Los comentarios TODO son sólo un esqueleto del programa. Serán reemplazados a medida que escriba el código real.

El número de teléfono comienza con un código de área opcional, por lo que el grupo de códigos de área va seguido de un signo de interrogación. Dado que el código de área puede ser sólo de tres dígitos (es decir, \d{3}) *o* de tres dígitos entre paréntesis (es decir,(\d{3}\)), debe tener una tubería que une esas partes. Puede añadir el comentario regex # Código de área a esta parte de la cadena multilínea para ayudarle a recordar qué (\d{3}|\\\
(\d{3}\))? debe coincidir.

El carácter separador del número de teléfono puede ser un espacio (\s), un guión (-) o un punto (.), por lo que estas partes también deben ir unidas por tubos. Las siguientes partes de la expresión regular son sencillas: tres dígitos, seguidos de otro separador, seguidos de cuatro dígitos. La última parte es una extensión opcional formada por cualquier número de espacios seguidos de ext, x, o ext., seguidos de dos a cinco dígitos.

Paso 2: Crear una Regex para las direcciones de correo electrónico

También necesitará una expresión regular que pueda coincidir con direcciones de correo electrónico. Haz que tu programa se parezca a lo siguiente:

```
#! python3
    # phoneAndEmail.py - Busca números de teléfono y direcciones de correo electrónico en el
    portapapeles. import pyperclip, re

    phoneRegex      =      re.compile(r'''(
    --snip--

    # Create email regex.
    emailRegex= re.compile(r'''(
❶       [a-zA-Z0-9._%+-]+          # Nombre DE USUARIO
❷       @                          # Símbolo @
➲       [a-zA-Z0-9.-]+             # nombre de
        dominio (\.[a-zA-Z]{2,4}) # punto-algo
        )''', re.VERBOSE)

    # TODO: Buscar coincidencias en el texto del portapapeles.

    # TODO: Copiar resultados al portapapeles.
```

La parte del nombre de usuario de la dirección de correo electrónico ❶ es uno o más caracteres que pueden ser cualquiera de los siguientes: letras minúsculas y mayúsculas, números, un punto, un guión bajo, un signo de porcentaje, un signo más o un guión. Todos estos caracteres pueden incluirse en una clase de caracteres: [a-zA-Z0-9._%+-].

El dominio y el nombre de usuario están separados por el símbolo @❷. El nombre de dominio ➲ tiene una clase de caracteres algo menos permisiva, con sólo letras, números, puntos y guiones: [a-zA-Z0-9.-]. Y por último estará la parte "punto-com" (técnicamente conocida como *dominio de nivel superior*), que en realidad puede ser punto-cualquier cosa. Tiene entre dos y cuatro caracteres.

El formato de las direcciones de correo electrónico tiene muchas reglas extrañas. Esta expresión regular no coincidirá con todas las direcciones de correo electrónico válidas posibles, pero coincidirá con casi cualquier dirección de correo electrónico típica que te encuentres.

Paso 3: Buscar todas las coincidencias en el texto del portapapeles

Ahora que ha especificado las expresiones regulares para números de teléfono y direcciones de correo electrónico, puede dejar que el módulo re de Python haga el trabajo duro de encontrar todas las coincidencias en el portapapeles. La función pyperclip.paste() obtendrá un valor de cadena del texto en el portapapeles, y el método regex findall() devolverá una lista de tuplas.

Haz que tu programa tenga el siguiente aspecto:

```
#! python3
    # phoneAndEmail.py - Busca números de teléfono y direcciones de correo electrónico en el

    portapapeles. import pyperclip, re

    phoneRegex      =      re.compile(r'''(
    --snip--

    # Buscar COINCIDENCIAS en el texto del
    portapapeles. text=
    str(pyperclip.paste())
❶ COINCIDE CON= []
❷ for GROUPS in phoneRegex.findall(text):
        phoneNum= '-'.join([GRUPOS[1], GRUPOS[3], GRUPOS[5]]) if
        GRUPOS[8] != '':
            phoneNum+= ' x'+ GROUPS[8]
        MATCHES.append(phoneNum)
➲   for GROUPS in emailRegex.findall(text):
        MATCHES.append(GROUPS[0])

    # TODO: Copiar resultados al portapapeles.
```

Hay una tupla por cada coincidencia, y cada tupla contiene cadenas para cada grupo de la expresión regular. Recuerde que el grupo 0 coincide con toda la expresión regular, por lo que el grupo en el índice 0 de la tupla es el que le interesa.

Como puedes ver en❶, almacenará las coincidencias en una variable de lista llamada coincidencias. Comienza como una lista vacía, y un par de bucles for. Para las direcciones de correo electrónico, añades el grupo 0 de cada coincidencia❸. En el caso de los números de teléfono coincidentes, no quiere añadir sólo el grupo 0. Aunque el programa *detecta* números de teléfono en varios formatos, quiere que el número de teléfono añadido esté en un único formato estándar. La variable phoneNum contiene una cadena construida a partir de los grupos 1, 3, 5 y 8 del texto coincidente❷. (Estos grupos son el prefijo, los tres primeros dígitos, los cuatro últimos dígitos y la extensión).

Paso 4: Unir las coincidencias en una cadena para el portapapeles

Ahora que tiene las direcciones de correo electrónico y los números de teléfono como una lista de cadenas en `matches`, quiere ponerlos en el portapapeles. La función `pyperclip.copy()` toma sólo un valor de cadena, no una lista de cadenas, por lo que se llama al método `join()` en `matches`.

Para que sea más fácil ver que el programa funciona, imprimamos en el terminal las coincidencias encontradas. Y si no se encontraron números de teléfono o direcciones de correo electrónico, el programa debe decirle esto al usuario.

Haz que tu programa tenga el siguiente aspecto:

```
#! python3
# phoneAndEmail.py - Busca números de teléfono y direcciones de correo electrónico en el portapapeles.

--snip--
for groups in emailRegex.findall(text):
    matches.append(groups[0])

# Copiar RESULTADOS al
portapapeles. if len(MATCHES)> 0:
    pyperclip.copy('\n'.join(MATCHES))
    print('Copiado al portapapeles:')
    print('\n'.join(MATCHES))
si no:
    print('No se han encontrado NÚMEROS DE teléfono ni DIRECCIONES DE correo electrónico.')
```

Ejecución del programa

Por ejemplo, abra su navegador web en la página de contacto de No Starch Press en http://www.nostarch.com/contactus.htm, pulse CTRL-A para seleccionar todo el texto de la página y pulse CTRL-C para copiarlo en el portapapeles. Cuando ejecute este programa, el resultado será algo parecido a esto:

```
Copiado al
portapapeles: 800-
420-7240
415-863-9900
415-863-9950
info@nostarch.com media@nostarch.com
academic@nostarch.com help@nostarch.com
```

Ideas para programas similares

Identificar patrones de texto (y posiblemente sustituirlos con la función sub()
) tiene muchas aplicaciones potenciales.

- Busque URL de sitios web que empiecen por *http://* o https://.
- Limpie las fechas en diferentes formatos de fecha (como 14/3/2015, 14/03-2015 y 14/3/2015) sustituyéndolas por fechas en un único formato estándar.
- Elimine la información confidencial, como los números de la Seguridad Social o de tarjetas de crédito.
- Encuentra errores tipográficos comunes como múltiples espacios entre palabras, palabras repetidas accidentalmente o múltiples signos de exclamación al final de las frases. ¡¡Esos sí que molestan!!

Resumen

Aunque un ordenador puede buscar texto rápidamente, hay que indicarle con precisión lo que debe buscar. Las expresiones regulares permiten especificar patrones precisos de caracteres. De hecho, algunas aplicaciones de tratamiento de textos y hojas de cálculo ofrecen funciones de buscar y reemplazar que permiten buscar utilizando expresiones regulares.

El módulo re que viene con Python le permite compilar objetos Regex. Estos valores tienen varios métodos: search() para encontrar una única coincidencia, findall() para encontrar todas las instancias coincidentes, y sub() para hacer una sustitución de texto mediante buscar y reemplazar.

La sintaxis de las expresiones regulares es un poco más compleja de lo que se describe en este capítulo. Puedes encontrar más información en la documentación oficial de Python en http://docs.python.org/3/library/re.html. El sitio web del tutorial *http://www.regular-expressions.info/* también es un recurso útil.

Ahora que ya tienes experiencia manipulando y emparejando cadenas, es el momento de sumergirte en cómo leer y escribir en archivos del disco duro de tu ordenador.

Preguntas prácticas

P: 1. ¿Cuál es la función que crea objetos Regex?

P: 2. ¿Por qué se suelen utilizar cadenas sin procesar al crear objetos Regex?

P: 3. ¿Qué devuelve el método search()?

P: 4. ¿Cómo se obtienen las cadenas reales que coinciden con el patrón de un objeto Match?

P: 5. En el regex creado a partir de r'(\d\d\d)-(\d\d\d-\d\d\d)', ¿qué cubre el grupo 0? ¿el grupo 1? ¿Grupo 2?

P: 6. Los paréntesis y puntos tienen significados específicos en la sintaxis de expresiones regulares. ¿Cómo especificaría que desea que una expresión regular coincida con caracteres de paréntesis y puntos reales?

P: 7. El método findall() devuelve una lista de cadenas o una lista de tuplas de cadenas. ¿Qué hace que devuelva una u otra?

P: 8. ¿Qué significa el carácter| en las expresiones regulares?

P: 9. ¿Qué dos cosas significa el carácter ? en las expresiones regulares?

P: 10. ¿Cuál es la diferencia entre los caracteres+ y * en las expresiones regulares?

P: 11. ¿Cuál es la diferencia entre {3} y {3,5} en expresiones regulares?

Q: 12. ¿Qué significan las clases de caracteres abreviados \d, \w, y \s en expresiones regulares?

P: 13. ¿Qué significan las clases de caracteres abreviados \D, \W, y \S en expresiones regulares?

P: 14. ¿Cómo hacer que una expresión regular no distinga entre mayúsculas y minúsculas?

P: 15. ¿A qué corresponde normalmente el carácter .? ¿Con qué coincide si re.DOTALL se pasa como segundo argumento a re.compile()?

Q: 16. ¿Cuál es la diferencia entre .* y .*?

P: 17. ¿Cuál es la sintaxis de la clase de caracteres para que coincidan todos los números y letras minúsculas?

P: 18. Si numRegex= re.compile(r'\d+ '), ¿qué devolverá numRegex.sub("X", '12 tamborileros, 11 gaiteros, cinco anillos, 3 gallinas')?

P: 19. ¿Qué te permite hacer pasar re.VERBOSE como segundo argumento a re.compile()?

P: 20. ¿Cómo escribirías una regex que coincida con un número con comas cada tres dígitos? Debe

coincide con lo siguiente:

- '42'
- '1,234'
- '6,368,745'

pero no lo siguiente:

- 12.34.567" (que sólo tiene dos cifras entre comas)
- '1234' (que carece de comas)

P: 21. ¿Cómo escribirías una regex que coincida con el nombre completo de alguien cuyo apellido es Nakamoto? Puedes asumir que el primer nombre que le precede siempre será una palabra que comienza con mayúscula. La regex debe coincidir con lo siguiente

- Satoshi Nakamoto
- Alice Nakamoto
- RoboCop Nakamoto

pero no lo siguiente:

- satoshi Nakamoto"" (sin mayúsculas en el nombre)
- "Sr. Nakamoto" (cuando la palabra precedente tenga un carácter no alfabético)
- "Nakamoto' (que no tiene de pila)
- 'Satoshi nakamoto" (donde Nakamoto no va en mayúsculas)

P: 22. ¿Cómo escribiría una expresión regular que coincida con una oración en la que la primera palabra sea *Alice*, *Bob* o *Carol*; la segunda palabra sea *come*, *mascotas* o *lanza*; la tercera palabra sea *manzanas*, *gatos* o *pelotas de béisbol*; y la oración termine con un punto? Esta expresión regular no debe distinguir entre mayúsculas y minúsculas. Debe coincidir con lo siguiente:

- Alice come
- manzanas'. Bob
- tiene gatos.
- Carol lanza pelotas de
- béisbol'. Alice lanza
- manzanas. BOB COME
- GATOS'.

pero no lo siguiente:

- RoboCop come manzanas.
- ALICE LANZA BALONES.
- 'Carol se come 7 gatos'.

Proyectos prácticos

Para practicar, escribe programas que realicen las siguientes tareas.

Detección de contraseñas seguras

Escribe una función que utilice expresiones regulares para asegurarse de que la cadena de contraseña que se le pasa es segura. Una contraseña segura se define como aquella que al menos ocho caracteres, contiene tanto mayúsculas como minúsculas y tiene al menos un dígito. Es posible que tenga que comprobar la cadena con varios patrones de expresiones regulares para validar su solidez.

Versión regex de strip()

Escriba una función que tome una cadena y haga lo mismo que el método `strip()` string. Si no se pasan más argumentos que la cadena a eliminar, se eliminarán los espacios en blanco al principio y al final de la cadena. En caso contrario, se eliminarán de la cadena los caracteres especificados en el segundo argumento de la función.

---

[1] Cory Doctorow, "Here's what ICT should really teach kids: how to do regular expressions", *Guardian*, 4 de diciembre de 2012, *http://www.theguardian.com/technology/2012/dec/04/ict-teach-kids-regular- expressions/*.

Capítulo 8. Lectura y escritura de archivos Lectura y escritura de archivos

Las variables son una buena forma de almacenar datos mientras tu programa se está ejecutando, pero si quieres que tus datos persistan incluso después de que tu programa haya terminado, necesitas en un archivo. Puedes pensar en el contenido de un archivo como un único valor de cadena, potencialmente de gigabytes de tamaño. En este capítulo, aprenderás cómo usar Python para crear, leer y guardar archivos en el disco duro.

Archivos y rutas de archivos

Un archivo tiene dos propiedades clave: un *nombre de archivo* (normalmente escrito en una sola palabra) y una *ruta*. La ruta especifica la ubicación de un archivo en el ordenador. Por ejemplo, hay un archivo en mi ordenador portátil con Windows 7 con el nombre de archivo *proyectos.docx* en la ruta *C:\sers\asweigart\Documents*. La parte del nombre de archivo después del último punto se denomina *extensión* del archivo y le indica el tipo de archivo. *project.docx* es un documento de Word, y *Users*, *asweigart* y *Documents* se refieren a *carpetas* (también denominadas *directorios*). Las carpetas pueden contener archivos y otras carpetas. Por ejemplo, *proyecto.docx* está en la carpeta Documentos, que está dentro de la carpeta *asweigart*, que está dentro de la carpeta *Usuarios*. La Figura 8-1 muestra esta organización de carpetas.

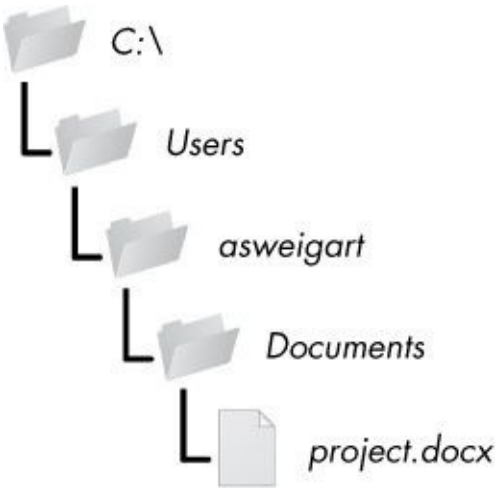


Figura 8-1. Un archivo en una jerarquía de carpetas

La parte *C:* de la ruta es la *carpeta raíz*, que contiene todas las demás carpetas. En Windows, la carpeta raíz se llama *C:* y también se denomina *unidad C:*. En OS X y Linux, la carpeta raíz es */*. En este libro, utilizaré la carpeta raíz de Windows, *C:*. Si estás entrando en los ejemplos de shell interactivo en OS X o Linux, introduce `/` en su lugar.

Los volúmenes adicionales, como una unidad de DVD o una memoria USB, aparecerán de forma diferente en los distintos sistemas operativos. En Windows, aparecen como nuevas unidades raíz con letras, como *D:* o *E:*. En OS X, aparecen como nuevas carpetas bajo la carpeta */Volumes*. En Linux, aparecen como nuevas carpetas bajo la carpeta */mnt* ("mount"). Ten en cuenta que los nombres de carpetas y archivos no distinguen entre mayúsculas y minúsculas en Windows y OS X, pero sí en Linux.

Barra invertida en Windows y barra diagonal en OS X y Linux

En Windows, las rutas se escriben utilizando barras invertidas (\) como separador entre los nombres de las carpetas. OS X y Linux, sin embargo, utilizan la barra oblicua (/) como separador de rutas. Si quieres que tus programas funcionen en todos los sistemas operativos, tendrás que escribir tus scripts Python para manejar ambos casos.

Afortunadamente, esto es sencillo de hacer con la función `os.path.join()`. Si le pasas los valores de cadena de los nombres individuales de archivos y carpetas de tu ruta, `os.path.join()` devolverá una cadena con una ruta de archivo utilizando los separadores de ruta correctos. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar os
>>> os.path.join('usr', 'bin', 'spam')
'usr\\bin\\spam'
```

Estoy ejecutando estos ejemplos de shell interactivo en Windows, por lo que `os.path.join('usr', 'bin', 'spam')` devolvió `'usr\\bin\\spam'`. (Observe que las barras invertidas se duplican porque cada barra invertida necesita ser escapada por otro carácter de barra invertida). Si hubiera llamado a esta función en OS X o Linux, la cadena habría sido `'usr/bin/spam'`.

La función os.path.join() es útil si necesita crear cadenas para los nombres de archivos. Estas cadenas se pasarán a varias de las funciones relacionadas con archivos introducidas en este capítulo. Por ejemplo, el siguiente ejemplo une nombres de una lista de nombres de archivos al final del nombre de una carpeta:

```
>>> misArchivos= ['cuentas.txt', 'detalles.csv', 'invitar.docx']
>>> para nombrearchivo en misArchivos:
        print(os.path.join('C:\\sers\asweigart', filename))
C:\sers\asweigart\accounts.txt C:\sers\asweigart\details.csv
C:\Usuarios\asweigart\invitación.docx
```

Directorio de trabajo actual

Todos los programas que se ejecutan en su ordenador tienen un *directorio de trabajo actual*, o *cwd*. Cualquier nombre de archivo o ruta que no comience con la carpeta raíz se asume que está bajo el directorio de trabajo actual. Puede obtener el directorio de trabajo actual como un valor de cadena con la función os.getcwd() y cambiarlo con os.chdir(). Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar os
>>> os.getcwd()
C:Python34
>>>      os.chdir("C:\Windows\System32")
>>> os.getcwd()
"C:WindowsSystem32"
```

Aquí, el directorio de trabajo actual se establece en *C:\Python34*, por lo que el nombre de archivo *proyecto.docx* se refiere a *C:\Python34\proyecto.docx*. Cuando cambiamos el directorio de trabajo actual a *C:\Windows*, *project.docx* se interpreta como *C:\Windows\project.docx*.

Python mostrará un error si intenta cambiar a un directorio que no existe.

```
>>> os.chdir('C:\ThisFolderDoesNotExist')
Traceback (llamada más reciente):
  File "<pyshell#18>", line 1, in <module>
    os.chdir('C:\\ThisFolderDoesNotExist')
FileNotFoundError: [WinError 2] El sistema no puede encontrar el archivo especificado:
'C:\\ThisFolderDoesNotExist'
```

NOTA

Aunque carpeta es el nombre más moderno para directorio, ten en cuenta que directorio de trabajo actual *(o simplemente* directorio de trabajo*) es el término estándar, no carpeta de trabajo actual.*

Trayectorias absolutas frente a trayectorias relativas

Hay dos formas de especificar una ruta de archivo.

- Una *ruta absoluta*, que siempre empieza por la carpeta raíz
- Una *ruta relativa*, que es relativa al directorio de trabajo actual del programa.

También existen las carpetas *punto* (`.`) y *punto* (`..`). No son carpetas reales, sino nombres especiales que pueden utilizarse en una ruta. Un solo punto ("punto") para el nombre de una carpeta es la abreviatura de "este directorio". Dos puntos ("punto-punto") significa "la carpeta padre".

La figura 8-2 es un ejemplo de algunas carpetas y ficheros. Cuando el directorio de trabajo actual se establece en *C:\bacon*, las rutas relativas para las otras carpetas y archivos se establecen como están en la figura.

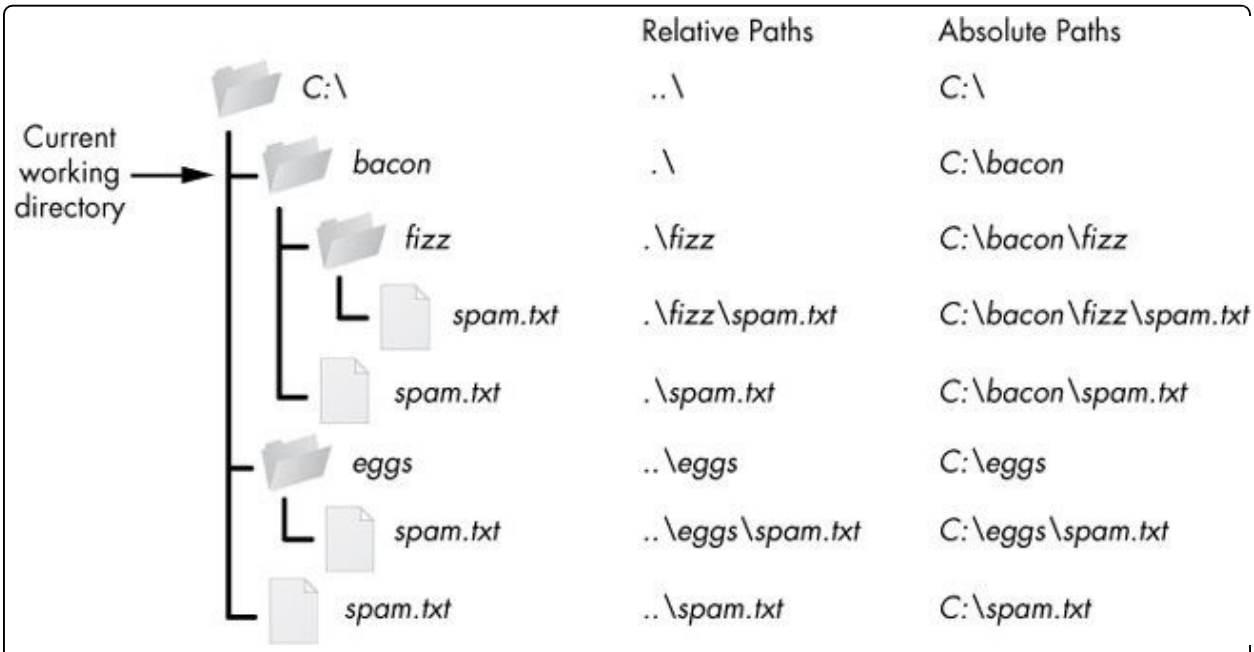


Figura 8-2. Las rutas relativas para carpetas y archivos en el directorio de trabajo C:\bacon

El `.\` al principio de una ruta relativa es opcional. Por ejemplo, *.\spam.txt* y *spam.txt* se refieren al mismo archivo.

Creación de nuevas carpetas con os.makedirs()

Sus programas pueden crear nuevas carpetas (directorios) con la función os.makedirs() función. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar os
>>> os.makedirs('C:\\delicious\walnut\waffles')
```

Esto creará no sólo la carpeta *C:\delicious* sino también una carpeta *walnut* dentro de *C:\delicious* y una carpeta *waffles* dentro de *C:\delicious\walnut*. Es decir, os.makedirs() creará las carpetas intermedias necesarias para asegurar que existe la ruta completa. La Figura 8-3 muestra esta jerarquía de carpetas.

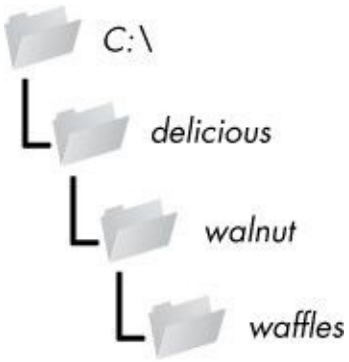


Figura 8-3. El resultado de os.makedirs('C:\\delicious')

El módulo os.path

El módulo os.path contiene muchas funciones útiles relacionadas con nombres y rutas de archivos. Por ejemplo, ya ha utilizado os.path.join() para construir rutas de forma que funcionen en cualquier sistema operativo. Como os.path es un módulo dentro del módulo os, puedes importarlo simplemente ejecutando import os. Siempre que tus programas necesiten trabajar con archivos, carpetas o rutas de archivos, puedes referirte a la breve función
ejemplos de esta sección. La documentación completa del módulo os.path se encuentra en el sitio web de Python en http://docs.python.org/3/library/os.path.html.

NOTA

La mayoría de los ejemplos que siguen en esta sección requerirán el módulo os, así que recuerda importarlo al principio de cualquier script que escribas y cada vez que reinicies IDLE. De lo contrario, obtendrás un mensaje de error NameError: name 'os' is not defined.

Manejo de rutas absolutas y relativas

El módulo os.path proporciona funciones para devolver la ruta absoluta de una ruta relativa y para comprobar si una ruta dada es una ruta absoluta.

- Llamar a os.path.abspath(*path*) devolverá una cadena con la ruta absoluta del argumento. Esta es una forma sencilla de convertir una ruta relativa en una absoluta.
- Llamar a os.path.isabs(*path*) devolverá True si el argumento es una ruta absoluta y False si es una ruta relativa.
- Llamar a os.path.relpath(*path, start*) devolverá una cadena de una ruta relativa desde la ruta de *inicio* hasta *path*. Si no se proporciona *start*, se utilizará el directorio de trabajo actual como ruta de inicio.

Pruebe estas funciones en el shell interactivo:

```
>>> os.path.abspath('.')
C:Python34
>>> os.path.abspath('.\\Scripts')
'C:Python34\\Scripts'
>>> os.path.isabs('.')
Falso
>>> os.path.isabs(os.path.abspath('.'))
Verdadero
```

Dado que *C:*\Python34 era el directorio de trabajo cuando se llamó a os.path.abspath(), la carpeta "de un solo punto" representa la ruta absoluta 'C:\Python34'.

> **NOTA**
>
> *Dado que tu sistema probablemente tenga archivos y carpetas diferentes a los míos, no podrás seguir exactamente todos los ejemplos de este capítulo. Aún así, intenta seguirlo usando carpetas que existan en tu ordenador.*

Introduzca las siguientes llamadas a os.path.relpath() en el intérprete de comandos interactivo:

```
>>> os.path.relpath("C:33Windows", 'C:33')
Windows
>>> os.path.relpath("C:\\Windows", 'C:\Spam\\Eggs')
"...Windows
>>> os.getcwd() 'C:\\\Python34'
```

Llamar a os.path.dirname(*path*) devolverá una cadena con todo lo que viene antes de la última barra en el argumento path. Llamando a os.path.basename(*path*)

devolverá una cadena con todo lo que viene después de la última barra en la ruta argumento. El nombre dir y el nombre base de una ruta se esbozan en la Figura 8-4.

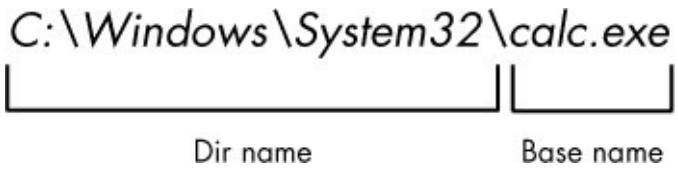


Figura 8-4. Nombre base El nombre base sigue a la última barra en una ruta y es el mismo que el nombre del fichero. El nombre dir es todo lo que hay antes de la última barra.

Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> ruta = 'C:\\Windows\System32\calc.exe'
>>> os.path.basename(ruta)
'calc.exe'
>>> os.path.dirname(ruta)
'C:WindowsSystem32'
```

Si necesita el nombre dir y el nombre base de una ruta juntos, sólo tiene que llamar a os.path.split() para obtener un valor de tupla con estas dos cadenas, así:

```
>>> calcFilePath = 'C:\\Windows\System32\calc.exe'
>>> os.path.split(calcFilePath)
('C:\Windows\System32', 'calc.exe')
```

Observe que podría crear la misma tupla llamando a os.path.dirname() y a os.path.basename() y colocando sus valores de retorno en una tupla.

```
>>> (os.path.dirname(calcFilePath), os.path.basename(calcFilePath))
('C:\Windows\System32', 'calc.exe')
```

Pero os.path.split() es un buen atajo si necesitas ambos valores.

Además, tenga en cuenta que os.path.split() *no* toma una ruta de archivo y devuelve una lista de cadenas de cada carpeta. Para ello, utilice el método split() string y divídalo en la cadena de os.sep. Recuerde que la variable os.sep se establece en la barra separadora de carpetas correcta para el ordenador que ejecuta el programa.

Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> calcFilePath.split(os.path.sep)
['C:', 'Windows', 'System32', 'calc.exe']
```

En los sistemas OS X y Linux, habrá una cadena en blanco al principio de la lista devuelta:

```
>>> '/usr/bin'.split(os.path.sep)
```

```
['', 'usr', 'bin']
```

El método de cadena `split()` funcionará para devolver una lista de cada parte de la ruta. Funcionará en cualquier sistema operativo si se le pasa `os.path.sep`.

Cómo encontrar el tamaño de los archivos y el contenido de las carpetas

Una vez que disponga de formas de manejar las rutas de los archivos, podrá empezar a recopilar información sobre archivos y carpetas específicos. El módulo os.path proporciona funciones para encontrar el tamaño de un archivo en bytes y los archivos y carpetas dentro de una carpeta determinada.

- Llamar a os.path.getsize(*path*) devolverá el tamaño en bytes del archivo en el argumento *path*.
- Llamar a os.listdir(*path*) devolverá una lista de cadenas de nombres de archivo para cada archivo en el argumento *path*. (Tenga en cuenta que esta función se encuentra en el módulo os, no en os.path).

Esto es lo que obtengo cuando pruebo estas funciones en el shell interactivo:

```
>>> os.path.getsize("C:\Windows\System32\calc.exe")
776192
>>> os.listdir("C:\\Windows\\System32")
['0409', '12520437.cpx', '12520850.cpx', '5U877.ax', 'aaclient.dll',
--snip--
xwtpdui.dll', 'xwtpw32.dll', 'zh-CN', 'zh-HK', 'zh-TW', 'zipfldr.dll'].
```

Como puede ver, el programa *calc.exe* de mi ordenador tiene un tamaño de 776.192 bytes, y tengo muchos archivos en *C:\Windows\system32*. Si quiero encontrar el tamaño total de todos los archivos en este directorio, puedo usar os.path.getsize() y os.listdir() juntos.

```
>>> tamaño total= 0
>>> for filename in os.listdir("C:\\Windows\System32"):
        totalSize= totalSize+ os.path.getsize(os.path.join("C:\Windows\System32", filename))

>>> print(tamañoTotal)
1117846456
```

A medida que recorro cada nombre de fichero en la carpeta *C:\Windows\System32*, la variable totalSize se incrementa con el tamaño de cada fichero. Observe que cuando llamo a os.path.getsize()utilizo os.path.join() para unir el nombre de la carpeta con el nombre del fichero actual. El entero que devuelve os.path.getsize() se suma al valor de totalSize. Después de recorrer todos los archivos, imprimo totalSize para ver el tamaño total de la carpeta *C:\Windows\System32*.

Comprobación de la validez de la ruta

Muchas funciones de Python se bloquean con un error si se les proporciona una ruta que no existe. El módulo os.path proporciona funciones para comprobar si una ruta dada existe y si es un archivo o una carpeta.

- Llamar a os.path.exists(*path*) devolverá True si el archivo o carpeta al que se hace referencia en el argumento existe y devolverá False si no existe.
- Llamar a os.path.isfile(*path*) devolverá True si el argumento path existe y es un archivo y devolverá False en caso contrario.
- Llamar a os.path.isdir(*path*) devolverá True si el argumento path existe y es una carpeta y devolverá False en caso contrario.

Esto es lo que obtengo cuando pruebo estas funciones en el shell interactivo:

```
>>> os.path.exists("C:\Windows")
Verdadero
>>> os.path.exists('C:\\some_made_up_folder')
Falso
>>> os.path.isdir("C:\\Windows\System32")
Verdadero
>>> os.path.isfile("C:\Windows\System32")
Falso
>>> os.path.isdir("C:\Windows\System32\\calc.exe")
Falso
>>> os.path.isfile("C:\Windows\System32\\calc.exe")
Verdadero
```

Puede determinar si hay un DVD o una unidad flash actualmente conectada al ordenador comprobándolo con la función os.path.exists(). Por ejemplo, si quisiera comprobar si hay una unidad flash con el volumen llamado *D:* en mi ordenador Windows, podría hacerlo con lo siguiente:

```
>>> os.path.exists('D:\\')
Falso
```

¡Uy! Parece que olvidé conectar mi memoria USB.

El proceso de lectura y escritura de archivos

Una vez que te sientas cómodo trabajando con carpetas y rutas relativas, podrás especificar la ubicación de los archivos a leer y escribir. Las funciones cubiertas en las próximas secciones se aplicarán a los archivos de texto plano. *Los archivos de texto sin formato* sólo contienen caracteres de texto básicos y no incluyen información sobre fuentes, tamaños o colores. Los archivos de texto con *extensión .txt* o los archivos de script Python con extensión *.py* son ejemplos de de texto plano. Pueden abrirse con el Bloc de Notas de Windows o con la aplicación TextEdit de OS . Tus programas pueden leer fácilmente el contenido de los archivos de texto plano y tratarlo como un valor de cadena ordinario.

Los archivos binarios son todos los demás tipos de archivo, como documentos de procesamiento de texto, PDF, imágenes, hojas de cálculo y programas ejecutables. Si abre archivo binario en el Bloc de notas o en TextEdit, parecerá un sinsentido revuelto, como en la Figura 8-5.



Figura 8-5. El programa Windows calc.exe *abierto en el Bloc de notas*

Dado que cada tipo diferente de archivo binario debe ser manejado a su manera, este libro no se adentrará directamente en la lectura y escritura de archivos binarios en bruto. Afortunadamente, hay muchos módulos que facilitan el trabajo con archivos binarios; más adelante en este capítulo exploraremos uno de ellos, el módulo shelve.

Hay tres pasos para leer o escribir archivos en Python.

1. Llama a la función open() para devolver un objeto File.
2. Llama al método read() o write() del objeto File.

3. Cierra el archivo llamando al método close() del objeto File.

Abrir archivos con la función open()

Para abrir un archivo con la función open(), se le pasa una cadena de ruta que indica el archivo que se desea abrir; puede ser una ruta absoluta o relativa. La función open() devuelve un objeto File.

Pruébalo creando un archivo de texto llamado *hola.txt* utilizando el Bloc de notas o TextEdit. Escriba **¡Hola mundo!** como contenido de este archivo de texto y guárdelo en la carpeta de inicio de su usuario. A continuación, si utilizas Windows, introduce lo siguiente en el intérprete de comandos interactivo:

```
>>> holaArchivo= open('C:\\su_carpeta_de_tu_casa\hello.txt')
```

Si utiliza OS Xintroduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> holaArchivo= open('Usuariosu_carpeta_home/hola.txt')
```

Asegúrate de sustituir *tu_carpeta_home* por el nombre de usuario de tu ordenador. Por ejemplo, mi nombre de usuario es *asweigart*, así que introduciría 'C:\\sers\\\asweigart\\ hola.txt' en Windows.

Ambos comandos abrirán el archivo en modo "lectura de texto plano", o *modo lectura* para abreviar. Cuando un archivo se abre en modo lectura, Python sólo te permite leer datos del archivo; no puedes escribir o modificarlo de ninguna manera. El modo lectura es el modo por defecto para los archivos que abres en Python. Pero si no quieres confiar en los valores por defecto de Python, puedes especificar explícitamente el modo pasando la cadena
como segundo argumento de open(). Así, open('Usuariosasweigart/ hola.txt', 'r') y open('Usuariosasweigart/hola.txt') hacen lo mismo.

La llamada a open() devuelve un objeto File. Un objeto File representa un archivo en su ordenador; es simplemente otro tipo de valor en Python, muy parecido a las listas y diccionarios con los que ya está familiarizado. En el ejemplo anterior, almacenaste el objeto File en la variable helloFile. Ahora, cuando quieras leer o escribir en el archivo, puedes hacerlo llamando a los métodos del objeto Archivo en holaArchivo.

Lectura del contenido de los archivos

Ahora que tiene un objeto File, puede empezar a leer de él. Si desea leer todo el contenido de un archivo como un valor de cadena, utilice el método read() del objeto File. Continuemos con el objeto File *hello.txt* que almacenó en helloFile. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> holaContenido= holaArchivo.read()
>>> holaContenido
"¡Hola mundo!
```

Si considera el contenido de un archivo como un único valor de cadena de gran tamaño, la función read()
devuelve la cadena almacenada en el archivo.

Alternativamente, puede utilizar el método readlines() para obtener una *lista* de valores de cadena del archivo, una cadena por cada línea de texto. Por ejemplo, cree un archivo llamado *sonnet29.txt* en el mismo directorio que *hello.txt* y escriba el siguiente texto en él:

```
Cuando, en desgracia con la fortuna y los ojos de los
hombres, yo solo lloro mi estado de
paria,
Y alboroto al sordo cielo con mis gritos inútiles, Y me
miro a mí mismo y maldigo mi destino,
```

Asegúrese de separar las cuatro líneas con saltos de línea. A continuación, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> sonetoArchivo= open('soneto29.txt')
>>> sonnetFile.READLINES()
[Cuando, en desgracia con la fortuna y los ojos de los hombres, lloro
solo mi estado de paria, y molesto al cielo sordo con mis gritos inútiles, y me
miro a mí mismo y maldigo mi destino,']
```

Tenga en cuenta que cada uno de los valores de cadena termina con un carácter de nueva línea, \n, excepto la última línea del archivo. A menudo es más fácil trabajar con una lista de cadenas que con un único valor de cadena grande.

Escribir en archivos

Python te permite escribir contenido en un archivo de forma similar a como la función `print()` "escribe" cadenas en la pantalla. Sin embargo, no puedes escribir en un archivo que hayas abierto en modo lectura. En su lugar, necesitas abrirlo en modo "escribir texto plano" o "añadir texto plano", o *modo escribir* y *modo añadir* para abreviar.

El modo de escritura sobrescribirá el archivo existente y empezará desde cero, igual que cuando sobrescribe el valor de una variable con un nuevo valor. Pase `'w'` como segundo argumento a `open()` para abrir el fichero en modo escritura. El modo Append, por otro lado, añadirá texto al final del fichero existente. Puede pensar en esto como añadir a una lista en una variable, en lugar de sobrescribir la variable por completo. Pase `'a'` como segundo argumento a `open()` para abrir el fichero en modo append.

Si el nombre de fichero pasado a `open()` no existe, tanto el modo de escritura como el de adición crearán un nuevo fichero en blanco. Después de leer o escribir un archivo, llame al método `close()` antes de volver a abrir el archivo.

Unamos estos conceptos. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> baconFile= open('bacon.txt', 'w')
>>>   baconFile.write('¡Hola    mundo!\n')
13
>>> baconFile.close()
>>> baconFile= open('bacon.txt', 'a')
>>> baconFile.write('El bacon no es una verdura')
25
>>> baconFile.close()
>>> baconFile= open('bacon.txt')
>>> contenido= baconFile.read()
>>> baconFile.close()
>>> print(contenido)
¡Hola, mundo!
El beicon no es una verdura.
```

Primero, abrimos *bacon.txt* en modo escritura. Como todavía no existe *bacon.txt*, Python crea uno. Llamando `a write()` en el archivo abierto y pasando `a write()` el argumento cadena `'¡Hola mundo! /n'` escribe la cadena en el fichero y devuelve el número de caracteres escritos, incluyendo la nueva línea. Luego cerramos el archivo.

Para añadir texto al contenido existente del fichero en lugar de sustituir la cadena que acabamos de escribir, abrimos el fichero en modo append. Escribimos `'El beicon no es una verdura'` en el fichero y lo cerramos. Por último, para imprimir el contenido del fichero en el

abrimos el fichero en su modo de lectura por defecto, llamamos a read(), almacenamos el resultado en Objeto de archivo en el contenido, cerrar el archivo e imprimir el contenido.

Tenga en cuenta que el método write() no añade automáticamente un carácter de nueva línea al final de la cadena como hace la función print(). Tendrá que añadir este carácter usted mismo.

Guardar variables con el módulo shelve

Puedes guardar variables en tus programas Python en archivos binarios de estantería usando el módulo shelve. De esta forma, tu programa puede restaurar datos a variables desde el disco duro. El módulo shelve te permitirá añadir funciones Guardar y Abrir a tu . Por ejemplo, si ejecutas un programa e introduces algunos ajustes de configuración, puedes guardar esos ajustes en un archivo shelve y luego hacer que el programa los cargue la próxima vez que se ejecute.

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> estantería de importación
>>> shelfFile= shelve.open('mydata')
>>> GATOS= ['Zophie', 'Pooka', 'Simon']
>>> shelfFile['CATS']= CATS
>>> shelfFile.close()
```

Para leer y escribir datos utilizando el módulo shelve, primero hay que importar shelve. Llame a shelve.open() y pásele un nombre de archivo, y luego almacene el valor de estantería devuelto en una variable. Puede realizar cambios en el valor de la estantería como si fuera un diccionario.
Cuando haya terminado, llame a close() en el valor del estante. Aquí, nuestro valor de estantería se almacena en shelfFile. Creamos una lista gatos y escribimos shelfFile['gatos']= gatos para almacenar la lista en shelfFile como un valor asociado a la clave 'gatos' (como en un diccionario). Luego llamamos a close() en shelfFile.

Tras ejecutar el código anterior en Windows, verás tres nuevos archivos en el directorio de trabajo actual: *mydata.bak*, *mydata.dat* y *mydata.dir*. En OS X, sólo se creará un único archivo *mydata.db*.

Estos archivos binarios contienen los datos almacenados en la estantería. El formato de estos archivos binarios no es importante; sólo necesitas saber qué hace el módulo shelve, no cómo lo hace. El módulo le libera de preocuparse de cómo almacenar los datos de su programa en un fichero.

Sus programas pueden utilizar el módulo shelve para reabrir y recuperar posteriormente los datos de estos archivos de estantería. Los valores de la estantería no tienen que abrirse en modo de lectura o escritura: pueden hacer ambas cosas una vez abiertos. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> shelfFile= shelve.open('mydata')
>>> tipo(estanteriaArchivo)
<clase 'shelve.DbfilenameShelf'>
>>> shelfFile['GATOS']
```

```
['Zophie', 'Pooka', "Simon"]
>>> shelfFile.close()
```

Aquí, abrimos los archivos de estantería para comprobar que nuestros datos se almacenaron correctamente. Al introducir `shelfFile['cats']` nos devuelve la misma lista que almacenamos anteriormente, por lo que sabemos que la lista está correctamente almacenada, y llamamos a `close()`.

Al igual que los diccionarios, los valores de las estanterías tienen métodos `keys()` y `values()` que devuelven valores en forma de lista de las claves y valores de la estantería. Dado que estos métodos devuelven valores en forma de lista en lugar de verdaderas listas, debe pasarlos a la función `list()` para obtenerlos en forma de lista. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> shelfFile= shelve.open('mydata')
>>> list(estanteriaArchivo.CLAVES())
['gatos']
>>> list(estanteriaArchivo.VALORES())
[['Zophie', 'Pooka', "Simon"]]
>>> shelfFile.close()
```

Plaintext es útil para crear archivos que leerás en un editor de texto como Notepad o TextEdit, pero si quieres guardar datos de tus programas Python, utiliza el módulo `shelve`.

Guardar variables con la función pprint.pformat()

Recuerde de Impresión Bonita que la función pprint.pprint() "imprimirá bonitamente" el contenido de una lista o diccionario en la pantalla, mientras que la función pprint.pformat() devolverá este mismo texto como una cadena en lugar de . Esta cadena no sólo está formateada para ser fácil de leer, sino que también es código Python sintácticamente correcto. Digamos que tienes un diccionario almacenado en una variable
y quieres guardar esta variable y su contenido para un uso futuro. Usando pprint.pformat() obtendrás una cadena que puedes escribir en un archivo *.py*. Este archivo será tu propio módulo que podrás importar siempre que quieras utilizar la variable almacenada en él.

Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar pprint
>>> GATOS= [{'name': 'Zophie', 'desc': 'chubby'}, {'name': 'Pooka', 'desc':
'fluffy'}]
>>> pprint.pformat(GATOS)
"[{'desc': 'chubby', 'name': 'Zophie'}, {'desc': 'fluffy', 'name': 'Pooka'}]"
>>> fileObj= open('MISGATOS.py', 'w')
>>> fileObj.write('GATOS= '+ pprint.pformat(GATOS)+ '\n')
83
>>> fileObj.close()
```

Aquí, importamos pprint para permitirnos usar pprint.pformat(). Tenemos una lista de diccionarios, almacenada en una variable cats. Para mantener la lista en cats disponible incluso después de cerrar el shell, usamos pprint.pformat() para devolverla como una cadena. Una vez que tenemos los datos en gatos como una cadena, es fácil escribir la cadena en un archivo, que llamaremos *misGatos.py*.

Los módulos que importa una sentencia import son en sí mismos sólo scripts de Python. Cuando la cadena de caracteres de pprint.pformat() se guarda en un archivo *.py*, el archivo es un módulo que puede importarse como cualquier otro.

Y como los scripts Python son en sí mismos sólo archivos de texto con la extensión *.py*, tus programas Python pueden incluso generar otros programas Python. Luego puedes importar estos archivos en scripts.

```
>>> importar MISGATOS
>>> MISGATOS.GATOS
[{'name': 'Zophie', 'desc': 'chubby'}, {'name': Pooka', 'desc': 'fluffy'}]
```

```
>>> misGatos.gatos[0]
{"name": 'Zophie', 'desc': 'gordito'}
>>> misGatos.gatos[0]['nombre']
Zophie
```

La ventaja de crear un archivo *.py* (en lugar de guardar variables con el módulo shelve) es que, al tratarse de un archivo de texto, su contenido puede ser leído y modificado por cualquiera que disponga de un simple editor de texto. Para la mayoría de las aplicaciones, sin embargo, guardar datos utilizando el módulo shelve es la forma preferida de guardar variables en un archivo. Sólo los tipos de datos básicos como enteros, flotantes, cadenas, listas y diccionarios pueden escribirse en un fichero como texto simple. Los objetos de fichero, por ejemplo, no pueden codificarse como texto.

Proyecto: Generación de archivos aleatorios de cuestionarios

Supongamos que eres un profesor de geografía con 35 alumnos en clase y quieres hacer un examen sorpresa sobre las capitales de los estados de EE.UU.. Pero tu clase hay algunos alumnos malos y no puedes confiar en que no hagan trampas. Te gustaría aleatorizar el orden de las preguntas para que cada cuestionario sea único, de modo que nadie pueda copiar las respuestas de los demás. Por supuesto, hacer esto a mano sería un asunto largo y aburrido. Afortunadamente, sabes algo de Python.

Esto es lo que hace el programa:

- Crea 35 cuestionarios diferentes.
- Crea 50 preguntas de opción múltiple para cada cuestionario, en orden aleatorio.
- Proporciona la respuesta correcta y tres respuestas erróneas aleatorias para cada pregunta, en orden aleatorio.
- Escribe los cuestionarios en 35 archivos de texto.
- Escribe las claves de respuesta en 35 archivos de texto.

Esto significa que el código tendrá que hacer lo siguiente:

- Almacena los estados y sus capitales en un diccionario.
- Llame a `open()`, `write()`, y `close()` para los archivos de texto del cuestionario y de la clave de respuestas.
- Utilice `random.shuffle()` para aleatorizar el orden de las preguntas y las opciones de opción múltiple.

Paso 1: Almacenar los datos del cuestionario en un diccionario

El primer paso es crear un script esqueleto y llenarlo con los datos de tu cuestionario. Crea un archivo llamado *randomQuizGenerator.py*, y haz que se parezca a lo siguiente:

```
#! python3
    # randomQuizGenerator.py - Crea cuestionarios con preguntas y respuestas en # orden
    aleatorio, junto con la clave de respuestas.

❶ importar aleatorio

    # Los datos del cuestionario. Las claves son los estados y los valores sus capitales.
❷ capitales= {"Alabama": 'Montgomery', 'Alaska': 'Juneau', 'Arizona': 'Phoenix',
    'Arkansas': 'Little Rock', 'California': Sacramento', 'Colorado': 'Denver',
    'Connecticut': 'Hartford', 'Delaware': 'Dover', 'Florida': 'Tallahassee', "Georgia":
    Atlanta', Hawaii': Honolulu', 'Idaho': 'Boise', 'Illinois': 'Springfield',
    Indiana': 'Indianapolis', 'Iowa': 'Des Moines", 'Kansas': 'Topeka', 'Kentucky':
    Frankfort', 'Louisiana': 'Baton Rouge', 'Maine': Augusta', "Maryland": Annapolis',
    'Massachusetts': 'Boston', "Michigan": 'Lansing', "Minnesota": 'Saint Paul',
    'Mississippi': Jackson', 'Missouri': Jefferson City', "Montana": 'Helena',
    'Nebraska': 'Lincoln', 'Nevada': 'Carson City', 'New Hampshire': 'Concord', 'New
    Jersey': Trenton', 'Nuevo México": 'Santa Fe', 'Nueva York': 'Albany', 'Carolina del
    Norte': 'Raleigh', 'Dakota del Norte': "Bismarck", 'Ohio': "Columbus", "Oklahoma":
    "Oklahoma City',
    'Oregon': "Salem", 'Pennsylvania': Harrisburg', 'Rhode Island': Providence', 'South
    Carolina': "Columbia", 'South Dakota': 'Pierre', 'Tennessee': 'Nashville', 'Texas':
    'Austin', 'Utah': 'Salt Lake City', "Vermont": "Montpelier", 'Virginia':
    Richmond", "Washington": Olympia", "West Virginia': Charleston', "Wisconsin":
    "Madison", "Wyoming": 'Cheyenne'}.

        # Generate 35 quiz files.
➲  para quizNum en range(35):
        # TODO: Crear los archivos del quiz y de la clave

        de respuestas. # TODO: Escribir la cabecera para

        el quiz.

        # TODO: Barajar el orden de los estados.

        # TODO: Recorrer los 50 estados, haciendo una pregunta para cada uno.
```

Como este programa ordenará las preguntas y respuestas aleatoriamente, necesitarás importar el módulo random ❶ para hacer uso de sus funciones. La variable capitales ❷ contiene un diccionario con los estados de EE.UU. como claves y sus capitales como valores. Y ya que quieres crear 35 cuestionarios, el código que realmente genera los ficheros de cuestionarios y claves de respuestas (marcado con comentarios TODO por ahora) irá

dentro de un bucle for que se repite 35 veces ❸. (Este número puede cambiarse para generar cualquier número de archivos de cuestionarios).

Paso 2: Crear el archivo del cuestionario y cambiar el orden de las preguntas

Ahora es el momento de empezar a rellenar esos TODOs.

El código del bucle se repetirá 35 veces, una por cada cuestionario, por lo que sólo tendrás que preocuparte de un cuestionario a la vez dentro del bucle. Primero creará el archivo del cuestionario. Necesita tener un nombre de archivo único y también debe tener algún tipo de encabezado estándar, con lugares para que el estudiante rellene un nombre, fecha y período de clase. A continuación, tendrá que obtener una lista de estados en orden aleatorio, que se puede utilizar más tarde para crear las preguntas y respuestas para el .

Añade las siguientes líneas de código a *randomQuizGenerator.py*:

```
#! python3
    # randomQuizGenerator.py - Crea cuestionarios con preguntas y respuestas en # orden
    aleatorio, junto con la clave de respuestas.

    --snip--

    # Generar 35 archivos de
    cuestionarios. for quizNum in
    range(35):
        # Crear los ARCHIVOS del cuestionario y de la clave de respuestas.
❶       quizFile= open('CAPITALSQUIZ%S.txt' % (quizNum+ 1), 'w')
❷       answerKeyFile= open('CAPITALSQUIZ_ANSWERS%S.txt' % (quizNum+ 1), 'w')

        # Escribe la cabecera del cuestionario.
➲       quizFile.write('Nombre:Fecha:Periodo:Periodo')
        quizFile.write((' ' * 20)+ 'Cuestionario CAPITALES de Estado (Formulario %S)' % (quizNum+ 1))
        quizFile.write('\n\n')

        # Baraja el orden de los ESTADOS.
        STATES= list(CAPITALS.KEYS())
❹       random.shuffle(ESTADOS)

        # TODO: Recorrer los 50 estados, haciendo una pregunta para cada uno.
```

Los nombres de archivo para los cuestionarios serán *capitalsquiz<N>.txt*, donde *<N>* es un número único para el cuestionario que proviene de quizNum, el contador del bucle for. La clave de respuesta para *capitalsquiz<N>.txt* se almacenará en un archivo de texto llamado *capitalsquiz_answers<N>.txt*. Cada vez que pase por el bucle, el marcador de posición %s en 'capitalsquiz%s.txt' y 'capitalsquiz_answers%s.txt' será sustituido por (quizNum+ 1), por lo que el primer cuestionario y clave de respuestas creados serán

capitalsquiz1.txt y *capitalsquiz_answers1.txt*. Estos archivos se crearán con llamadas a la función open() en ❶ y❷, con `'w'` como segundo argumento para abrirlos en modo escritura.

Las sentencias write() de ❸ crean una cabecera de cuestionario para que el alumno la rellene. Por último, se crea una lista aleatoria de estados de EE.UU. con la ayuda de la función random.shuffle(❹, que reordena aleatoriamente los valores de cualquier lista que se pase.

Paso 3: Crear las opciones de respuesta

Ahora necesitas generar las opciones de respuesta para cada pregunta, que serán de opción múltiple de la A a la D. Necesitarás crear otro bucle for-éste para generar el contenido para cada una de las 50 preguntas del cuestionario. Luego habrá un tercer bucle for anidado dentro para generar las opciones de opción múltiple para cada pregunta. Haz que tu código sea como el siguiente:

```
#! python3
    # randomQuizGenerator.py - Crea cuestionarios con preguntas y respuestas en # orden
    aleatorio, junto con la clave de respuestas.

    --snip--

        # Recorre los 50 ESTADOS, haciendo una pregunta para cada uno. for
        questionNum in range(50):

            # Obtener RESPUESTAS correctas e incorrectas.
❶           correctAnswer= CAPITALES[ESTADOS[questionNum]]
❷           RESPUESTASEQUIVOCADAS = list(CAPITALS.VALUES())
➲           del RESPUESTASINCORRECTAS[RESPUESTASINCORRECTAS.index(RESPUESTACORRECTA)].
❹           RESPUESTASERRÓNEAS= random.sample(RESPUESTASERRÓNEAS, 3)
❺           ANSWEROPTIONS= WRONGANSWERS+ [RESPUESTACORRECTA]
❻           random.shuffle(OPCIONESRESPUESTA)

            # TODO: Escribir las OPCIONES DE pregunta y respuesta en el archivo

            del cuestionario. # TODO: Escribir la clave de respuestas en un

            archivo.
```

La respuesta correcta es fácil de obtener: se almacena como un valor en el diccionario de capitales ❶. Este bucle recorrerá los estados de la lista de estados barajados, desde states[0] hasta states[49], buscará cada estado en capitals y almacenará la capital correspondiente a ese estado en correctAnswer.

La lista de posibles respuestas erróneas es más complicada. Puedes obtenerla duplicando *todos* los valores del diccionario de mayúsculas❷, borrando la respuesta correcta❸, y seleccionando tres valores aleatorios de esta lista❹. La función random.sample() facilita esta selección. Su primer argumento es la lista de la que quieres seleccionar; el segundo argumento es el número de valores que quieres seleccionar. La lista completa de opciones de respuesta es la combinación de estas tres respuestas erróneas con el argumento

respuestas correctas ❺. Por último, las respuestas deben ser aleatorias ❻ para que la respuesta correcta no sea siempre la opción D.

Paso 4: Escribir el contenido de los archivos del cuestionario y de la clave de respuestas

Todo lo que queda es escribir la pregunta en el archivo del cuestionario y la respuesta en el archivo de la clave de respuestas. Haz que tu código sea como el siguiente:

```
#! python3
    # randomQuizGenerator.py - Crea cuestionarios con preguntas y respuestas en # orden
    aleatorio, junto con la clave de respuestas.

    --snip--
        # Recorre los 50 estados, haciendo una pregunta para cada uno.
        for questionNum in range(50):
            --snip--

            # Escribe la pregunta y las OPCIONES DE respuesta en el archivo del
            cuestionario. quizFile.write('%s. ¿Cuál ES la capital de %s?\n' % (questionNum+ 1,
                STATES[questionNum]))
❶           para i en rango(4):
❷               quizFile.write(' %s. %s\n' % ('ABCD'[i], ANSWEROPTIONS[i]))
            quizFile.write('\n')

            # Escribe la clave de respuestas en un archivo.
➲           answerKeyFile.write('%s. %s\n' % (questionNum+ 1, 'ABCD'[
                ANSWEROPTIONS.index(CORRECTANSWER)]))
        quizFile.close()
        answerKeyFile.close()
```

Un bucle for que recorre los enteros 0 a 3 escribirá las opciones de respuesta en la lista answerOptions ❶. La expresión 'ABCD'[i] en ❷ trata la cadena 'ABCD' como una matriz y se evaluará a 'A','B', 'C', y luego 'D' en cada iteración respectiva a través del .

En la línea finalla expresión answerOptions.index(correctAnswer) encontrará el índice entero de la respuesta correcta en las opciones de respuesta ordenadas aleatoriamente, y 'ABCD'[answerOptions.index(correctAnswer)] se evaluará a la letra de la respuesta correcta que se escribirá en el archivo de clave de respuesta.

Después de ejecutar el programa, este es el aspecto que tendrá su archivo *capitalsquiz1.txt*, aunque, por supuesto, sus preguntas y opciones de respuesta pueden ser diferentes de las que se muestran aquí, dependiendo del resultado de sus llamadas a random.shuffle():

```
Nombre:
```

Fecha:

Período:

Cuestionario sobre las capitales de los Estados (Formulario 1)

1. ¿Cuál es la capital de Virginia Occidental?
   A. Hartford
   B. Santa Fe
   C. Harrisburg
   D. Charleston

2. ¿Cuál es la capital de Colorado?
   A. Raleigh
   B. Harrisburg
   C. Denver
   D. Lincoln

--snip--

El archivo de texto correspondiente *capitalsquiz_answers1.txt* tendrá el siguiente aspecto:

1. D
2. C
3. A
4. C

--snip--

Proyecto: Tablero Multiclip

Supongamos que tienes la aburrida tarea de rellenar muchos formularios en una página web o un programa informático con varios campos de texto. El portapapeles te ahorra tener que escribir el mismo texto una y otra vez. Pero en el portapapeles sólo puede haber una cosa a la vez. Si tienes varios trozos de texto diferentes que tienes que copiar y pegar, tienes que ir resaltando y copiando las mismas cosas una y otra vez.

Puedes escribir un programa en Python para llevar la cuenta de múltiples trozos de texto. Este "multiclipboard" se llamará *mcb.pyw* (ya que "mcb" es más corto de escribir que "multiclipboard"). La extensión *.pyw* significa Python no mostrará una ventana de Terminal cuando ejecute este programa. (Ver Apéndice B para más detalles).

El programa guardará cada fragmento de texto del portapapeles con una palabra clave. Por ejemplo, al ejecutar py mcb.pyw save spam, el contenido actual del portapapeles se guardará con la palabra clave *spam*. Más tarde, este texto puede cargarse de nuevo en el portapapeles ejecutando py mcb.pyw spam. Y si el usuario olvida qué palabras clave tiene, puede ejecutar py mcb.pyw `list` para copiar una lista de todas las palabras clave en el portapapeles.

Esto es lo que hace el programa:

- Se comprueba el argumento de la línea de comandos para la palabra clave.
- Si el argumento es guardar, el contenido del portapapeles se guarda en la palabra clave.
- Si el argumento es `lista`, todas las palabras clave se copian en el portapapeles.
- De lo contrario, el texto de la palabra clave se copia en el teclado. Esto

significa que el código tendrá que hacer lo siguiente:

- Lee los argumentos de la línea de comandos desde sys.argv.
- Leer y escribir en el portapapeles.
- Guardar y cargar en un archivo de estantería.

Si utiliza Windows, puede ejecutar fácilmente este script desde la ventana Ejecutar... creando un archivo por lotes llamado *mcb.bat* con el siguiente contenido:

```
@pyw.exe C:\Python34\mcb.pyw %*
```

Paso 1: Comentarios y configuración de la estantería

Empecemos haciendo un esqueleto de script con algunos comentarios y una configuración básica. Haz que tu código sea como el siguiente:

```
#! python3
    # mcb.pyw - Guarda y carga fragmentos de texto en el portapapeles.
❶ # Uso: py.exe mcb.pyw guardar <palabra clave> - Guarda el portapapeles en la
    palabra clave. #         py.exe mcb.pyw <palabraclave> - Carga la palabra clave
    en el portapapeles.
    #             py.exe mcb.pyw list - Carga todas las palabras clave en el portapapeles.

❷ import shelve, pyperclip, sys

➲  mcbEstante    =    shelve.open("mcb")

    # TODO: Guardar el contenido del portapapeles.

    # TODO: Listar palabras clave y cargar contenido.

    mcbShelf.close()
```

Es una práctica común poner información general de uso en comentarios en la parte superior del archivo❶. Si alguna vez olvidas cómo ejecutar tu script, siempre puedes mirar estos comentarios como recordatorio. A continuación, importa tus módulos ❷. Copiar y pegar requerirá el módulo pyperclip, y leer los argumentos de la línea de comandos requerirá el módulo sys. El módulo shelve también será útil: Cada vez que el usuario quiera guardar un nuevo trozo de texto del portapapeleslo guardará en un archivo de estantería. Luego, cuando el usuario quiera volver a pegar el texto en su portapapeles, abrirás el archivo de la estantería y lo cargarás de nuevo en tu programa. El archivo
se denominará con el prefijo *mcb*❸.

Paso 2: Guardar el contenido del portapapeles con una palabra clave

El programa hace cosas diferentes dependiendo de si el usuario quiere guardar texto en una palabra clave, cargar texto en el portapapeles o listar todas las palabras clave existentes. Vamos a tratar el primer caso. Haz que tu código sea como el siguiente

```
#! python3
    # mcb.pyw - Guarda y carga fragmentos de texto en el portapapeles.
    --snip--

    # Guardar el contenido del portapapeles.
❶ si len(sys.argv)== 3 y sys.argv[1].lower()== 'guardar':
❷           mcbShelf[sys.argv[2]]= pyperclip.paste() elif
    len(sys.argv)== 2:
➲     # TODO: Listar palabras clave y cargar contenido.

    mcbShelf.close()
```

Si el primer argumento de la línea de órdenes (que siempre estará en el índice 1 de la lista sys.argv) es 'save'❶, el segundo argumento de la línea de órdenes es la palabra clave del contenido actual del portapapeles. La palabra clave se utilizará como clave para mcbShelf, y el valor será el texto actualmente en el portapapeles❷.

Si sólo hay un argumento en la línea de comandos, asumirá que es 'list' o una palabra clave para cargar contenido en el portapapeles. Implementarás ese código más adelante.

Por ahora, solo pon un comentario TODO ❸.

Paso 3: Enumerar palabras clave y cargar el contenido de una palabra clave

Por último, implementemos los dos casos restantes: El usuario quiere cargar el texto del portapapeles a partir de una palabra clave, o quiere una lista de todas las palabras clave disponibles. Haz que tu código sea como el siguiente:

```
#! python3
    # mcb.pyw - Guarda y carga fragmentos de texto en el portapapeles.
    --snip--

    # Guardar el contenido del portapapeles.
    if len(sys.argv)== 3 and sys.argv[1].lower()== 'save':
            mcbShelf[sys.argv[2]]= pyperclip.paste()
    elif len(sys.argv)== 2:
     # Listar PALABRAS CLAVE y cargar contenido.
        ❶       if SYS.argv[1].lower()== 'list':
❷           pyperclip.copy(str(list(mcbShelf.KEYS()))) elif
      SYS.argv[1] in mcbShelf:
➲           pyperclip.copy(mcbShelf[SYS.argv[1]])

    mcbShelf.close()
```

Si sólo hay un argumento en la línea de comandos, comprobemos primero si es "list".

❶. Si es así, se copiará en el portapapelesuna representación en forma de cadena de la lista de claves de estantería. El usuario puede pegar esta lista en un editor de texto abierto para .

De lo contrario, puede asumir que el argumento de la línea de comandos es una palabra clave. Si esta palabra clave existe en la estantería mcbShelf como clave, puede cargar el valor en el portapapeles ❸.

Y ya está. Iniciar este programa tiene diferentes pasos dependiendo del sistema operativo que utilice su ordenador. Consulte el Apéndice B para obtener más información sobre su sistema operativo.

Recuerda el programa de bloqueo de contraseñas que creaste en el Capítulo 6 y que almacenaba las contraseñas en un diccionario. Actualizar las contraseñas requería cambiar el código fuente del . Esto no es lo ideal porque los usuarios promedio no se sienten cómodos cambiando el código fuente para actualizar su software. Además, cada vez que se modifica el código fuente de un , se corre el riesgo de introducir accidentalmente nuevos errores. Si almacenas los datos de un programa en un lugar distinto del código, puedes hacer que tus programas sean más fáciles de usar y más resistentes a los errores.

Resumen

Los archivos se organizan en carpetas (también llamadas directorios), y una ruta describe la ubicación de un archivo. Cada programa que se ejecuta en su ordenador tiene un directorio de trabajo actual, lo que le permite especificar rutas de archivos relativas a la ubicación actual en lugar de escribir siempre la ruta completa (o absoluta). El módulo `os.path` tiene muchas funciones para manipular rutas de archivos.

Sus programas también pueden interactuar directamente con el contenido de los archivos de texto. La función `open()` puede abrir estos ficheros para leer su contenido como una gran cadena (con el método `read()`) o como una lista de cadenas (con el método `readlines()`). La función `open()` puede abrir archivos en modo escribir o añadir para crear nuevos archivos de texto o añadir a archivos de texto existentes, respectivamente.

En capítulos anteriores, utilizaste el portapapeles como una forma de introducir grandes cantidades de texto en un programa, en lugar de teclearlo todo. Ahora puedes hacer que tus programas lean archivos directamente del disco duro, lo que supone una gran mejora, ya que los archivos son mucho menos volátiles que el portapapeles.

En el próximo capítulo, aprenderás a manejar los propios archivos, copiándolos, borrándolos, renombrándolos, moviéndolos, etc.

Preguntas prácticas

P: 1. ¿A qué es relativa una ruta relativa?

P: 2. ¿Con qué empieza una ruta absoluta?

P: 3. ¿Qué hacen las funciones os.getcwd() y os.chdir()?

P: 4. ¿Qué son las carpetas . y ..?

P: 5. En *C:\bacon\eggs\spam.txt*, ¿que parte es el nombre dir, y que parte es el nombre base?

P: 6. ¿Cuáles son los tres argumentos "mode" que se pueden pasar a la función open()?

P: 7. ¿Qué ocurre si se abre un archivo existente en modo de escritura?

P: 8. ¿Cuál es la diferencia entre los métodos read() y readlines()?

P: 9. ¿A qué estructura de datos se parece un valor de estantería?

Proyectos prácticos

Para practicar, diseña y escribe los siguientes programas.

Ampliar el Multiclipboard

Amplíe el programa de tablero multiclip de este capítulo para que tenga una función de borrado <palabra clave> que borrará una palabra clave de la estantería. A continuación, añada un argumento de línea de comandos delete que borrará *todas las* palabras clave.

Mad Libs

Cree un programa Mad Libs que lea archivos de texto y permita al usuario añadir su propio texto en cualquier lugar del archivo de texto donde aparezca la palabra *ADJETIVO*, *NOMBRE*, *ADVERBO* o *VERBO*. Por ejemplo, un archivo de texto puede tener este aspecto:

> El ADJETIVO panda caminó hacia el NOMBRE y luego hacia el VERBO. Un sustantivo cercano no se vio afectado por estos acontecimientos.

El programa encontraría estas incidencias y pediría al usuario que las sustituyera.

> Introduzca un adjetivo:
> **tonto**
> Introduzca un sustantivo:
> **lámpara de araña**
> Introduzca un verbo:
> **gritó**
> Introduzca un sustantivo:
> **camioneta**

Se crearía entonces el siguiente archivo de texto:

> El panda tonto caminó hacia la araña y luego gritó. Una camioneta cercana no se vio afectada por estos acontecimientos.

Los resultados se imprimirán en la pantalla y se guardarán en un nuevo archivo de texto.

Búsqueda Regex

Escriba un programa que abra todos los archivos *.txt* de una carpeta y busque cualquier línea que coincida con una expresión regular proporcionada por el usuario. Los resultados deben imprimirse en la pantalla.

Capítulo 9. Organización de archivos

En el capítulo anterior, aprendiste cómo crear y escribir en nuevos archivos en Python. Tus programas también pueden organizar archivos preexistentes en el disco duro. Tal vez hayas tenido la experiencia de revisar una carpeta llena de docenas, cientos o incluso miles de archivos y copiarlos, renombrarlos, moverlos o comprimirlos a mano. O piensa en tareas como éstas:

- Hacer copias de todos los archivos PDF (y *sólo* de los archivos PDF) en cada subcarpeta de una carpeta
- Eliminación de los ceros a la izquierda en los nombres de todos los archivos de una carpeta con cientos de archivos llamados *spam001.txt*, *spam002.txt*, *spam003.txt*, etc.
- Comprimir el contenido de varias carpetas en un archivo ZIP (que podría ser un simple sistema de copia de seguridad).

Todas estas cosas aburridas están pidiendo ser automatizadas en Python. Programando tu ordenador para que haga estas tareas, puedes transformarlo en un archivero rápido que nunca comete errores.

Cuando empieces a trabajar con archivos, puede resultarte útil poder ver rápidamente cuál es la extensión (*.txt*, *.pdf*, *.jpg*, etc.) de un archivo. En OS X y Linux, lo más probable es que el explorador de archivos muestre las extensiones automáticamente. En Windows, las extensiones de archivo pueden estar ocultas por defecto. Para mostrar las extensiones, vaya a **Inicio► Panel de control► Apariencia y personalización► Opciones de carpeta**. En la pestaña Ver, en Configuración avanzada, desmarque la casilla **Ocultar extensiones para tipos de archivo conocidos**.

El módulo shutil

El módulo `shutil` (o shell utilities) tiene funciones que te permiten copiar, mover, renombrar y borrar ficheros en tus programas Python. Para usar las de `shutil`, primero necesitarás usar `import shutil`.

Copiar archivos y carpetas

El módulo shutil ofrece funciones para copiar archivos, así como carpetas enteras.

Al llamar a shutil.copy(*source, destination*) se copiará el archivo de la ruta *source* en la carpeta de la ruta *destination*. (Tanto el origen como *el destino* son cadenas.) Si *el destino* es un nombre de archivo, se utilizará como nuevo nombre del archivo copiado. Esta función devuelve una cadena con la ruta del fichero copiado.

Introduzca lo siguiente en el intérprete de comandos interactivo para ver cómo funciona shutil.copy():

```
>>> importar shutil, os
    >>> os.chdir('C:\\')
>>> ❶shutil.copy('C:\\\spam.txt', 'C:\\delicious')
    'C:-delicious-spam.txt'
>>> ❷shutil.copy('huevos.txt', 'C:huevos2.txt')
    'C:-delicioso-huevos2.txt'
```

La primera llamada a shutil.copy() copia el fichero en *C:\spam.txt* a la carpeta *C:\delicious*. El valor de retorno es la ruta del archivo recién copiado. Tenga en cuenta que, dado que se especificó una carpeta como destino❶, el nombre de archivo *spam.txt* original se utiliza para el nombre de archivo del nuevo archivo copiado. La segunda llamada a shutil.copy() ❷ también copia el fichero en *C:\eggs.txt* a la carpeta *C:\delicious* pero da al fichero copiado el nombre *eggs2.txt*.

Mientras que shutil.copy() copiará un único archivo, shutil.copytree() copiará una carpeta entera y todas las carpetas y archivos que contenga. Al llamar a shutil.copytree(*origen, destino*) se copiará la carpeta de la ruta de *origen*, junto con todos sus archivos y subcarpetas, en la carpeta de la ruta de *destino*. Los parámetros de *origen* y *destino* son cadenas. La función devuelve una cadena con la ruta de la carpeta copiada.

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar shutil, os
>>> os.chdir('C:\\')
>>> shutil.copytree('C:-bacon', 'C:-bacon_backup')
'C:\\bacon_backup'
```

La llamada a shutil.copytree() crea una nueva carpeta llamada *bacon_backup* con el mismo contenido que la carpeta *bacon* original. Ahora ha una copia de seguridad de su preciado, preciado bacon.

Mover y renombrar archivos y carpetas

Al llamar a shutil.move(*source, destination*) se moverá el archivo o carpeta de la ruta de *origen* a la ruta de *destino* y se devolverá una cadena con la ruta absoluta de la nueva ubicación.

Si *el destino* apunta a una carpeta, el archivo *de origen* se mueve al *destino* y mantiene su nombre de archivo actual. Por ejemplo, introduzca lo siguiente en el shell interactivo:

```
>>> importar shutil
>>> shutil.move('C:\\bacon.txt', 'C:\\EGGS')
'C:\eggs\\bacon.txt'
```

Asumiendo que ya existe una carpeta llamada *eggs* en el directorio *C:*, esto
La llamada shutil.move() dice: "Mueve *C:\bacon.txt* a la carpeta *C:\eggs*".

Si ya hubiera un archivo *bacon.txt* en *C:\eggs*, se habría sobrescrito. Dado que es fácil sobrescribir archivos accidentalmente de esta manera, debe tener cierto cuidado al utilizar move().

La ruta de *destino* también puede especificar un nombre de archivo. En el siguiente ejemplo, el archivo de origen se mueve *y* se le cambia el nombre.

```
>>> shutil.move('C:\\bacon.txt', 'C:\\EGGS\NEW_BACON.txt')
'C:³\eggs\\new_bacon.txt'
```

Esta línea dice: "Mueve *C:\bacon.txt* a la carpeta *C:\eggs*, y ya que estás, renombra ese archivo *bacon.txt* a *new_bacon.txt*".

Los dos ejemplos anteriores funcionaban bajo la suposición de que había una carpeta *eggs* en el directorio *C:*. Pero si no hay carpeta *eggs*, entonces move() renombrará *bacon.txt* a un fichero llamado *eggs*.

```
>>> shutil.move('C:\\bacon.txt', 'C:\\EGGS')
'C:\\\ggs'
```

Aquí, move() no puede encontrar una carpeta llamada *eggs* en el *directorio C:* y asume que *el destino* debe estar especificando un nombre de archivo, no una carpeta. Así que el archivo de texto *bacon.txt* es renombrado a *eggs* (un archivo de texto sin la extensión *.txt*) - ¡probablemente no lo que usted quería! Esto puede ser un error difícil de detectar en , ya que la llamada a move() puede hacer algo muy distinto lo que usted esperaba. Esta es otra razón para tener cuidado al usar move().

Por último, las carpetas que componen el destino ya deben existir, o de lo contrario Python lanzará una excepción. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> shutil.move('spam.txt', 'c:\DOES_NOT_EXIST\\EGGS\\HAM')
Traceback (llamada más reciente):
  File "C:Python34\lib\shutil.py", line 521, in move os.rename(src,
    real_dst)
FileNotFoundError: [WinError 3] El sistema no puede encontrar la ruta especificada:
'spam.txt' -> 'c:\does_not_exist\\eggs\\ham'

Durante la gestión de la excepción anterior, se produjo otra excepción: Traceback

(most recent call last):
  File "<pyshell#29>", line 1, in <module>
    shutil.move('spam.txt', 'c:\does_not_exist\\eggs\\ham')
  File "C:Python34\lib\shutil.py", line 533, in move copy2(src,
    real_dst)
  File "C:\Python34\lib\shutil.py", line 244, in copy2 copyfile(src,
    dst, follow_symlinks=follow_symlinks)
  File "C:Python34\lib\shutil.py", line 108, in copyfile with
    open(dst, 'wb') as fdst:
FileNotFoundError: [Errno 2] No such file or directory: 'c:\\does_not_exist\\\ eggs\\ham'
```

Python busca *huevos* y *jamón* dentro del directorio *does_not_exist*. No encuentra el directorio inexistente, por lo que no puede mover *spam.txt* a la ruta especificada.

Eliminación permanente de archivos y carpetas

Puede eliminar un único archivo o una única carpeta vacía con las funciones del módulo os, mientras que para eliminar una carpeta y todo su contenido, utilice el módulo shutil.

- Llamando a os.unlink(*path*) se borrará el fichero en *path*.
- Al llamar a os.rmdir(*ruta*) se borrará la carpeta de la *ruta*. Esta carpeta debe estar vacía de archivos o carpetas.
- Al llamar a shutil.rmtree(*ruta*) se eliminará la carpeta de la *ruta*, y todos los archivos y carpetas que contenga también se eliminarán.

Tenga cuidado al utilizar estas funciones en sus programas. A menudo es una buena idea ejecutar primero su programa con estas llamadas comentadas y con llamadas print() añadidas para mostrar los archivos que serían borrados. Aquí está un programa Python que estaba destinado a eliminar archivos que tienen la extensión de archivo *.txt* pero tiene un error tipográfico (resaltado en negrita) que hace que elimine archivos *.rxt* en su lugar:

```
importar os
para filename en os.listdir():
    if filename.endswith('.rxt'): os.unlink(filename)
```

Si tenía algún archivo importante que terminara en *.rxt*, se habría borrado de forma accidental y permanente. En lugar de eso, primero deberías haber ejecutado el programa así:

```
importar os
para filename en os.listdir():
    if filename.endswith('.rxt'):
        #os.unlink(nombrearchivo)
        print(nombrearchivo)
```

Ahora la llamada a os.unlink() está comentada, por lo que Python la ignora. En su lugar, imprimirá el nombre del fichero que se habría borrado. Ejecutar esta versión del programa primero le mostrará que accidentalmente le ha dicho al programa que borre archivos *.rxt* en lugar de archivos *.txt*.

Cuando esté seguro de que el programa funciona como es debido, elimine la línea print(nombrearchivo) y descomente la línea os.unlink(nombrearchivo). A continuación, ejecute el programa de nuevo para eliminar realmente los archivos.

Borrados seguros con el módulo send2trash

Dado que la función incorporada shutil.rmtree() de Python borra archivos y carpetas de forma irreversible, puede ser peligroso utilizarla. Una forma mucho mejor de borrar archivos y carpetas es con el módulo de terceros send2trash. instalar este módulo ejecutando pip install send2trash desde una ventana de Terminal. (Consulte el Apéndice A para una explicación más detallada de cómo instalar módulos de terceros). Usar send2trash es mucho más seguro que las funciones normales de borrado de Python, porque enviará carpetas y archivos a la papelera de tu ordenador o a la papelera de reciclaje en lugar de permanentemente. Si un error en tu programa borra algo con send2trash que no tenías intención de borrar, puedes restaurarlo más tarde desde la papelera de reciclaje.

Después de haber instalado send2trash, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar send2trash
>>> baconFile= open('bacon.txt', 'a') # CREA el archivo
>>> baconFile.write('El bacon no es una verdura')
25
>>> baconFile.close()
>>> SEND2TRASH.SEND2TRASH('bacon.txt')
```

En general, siempre debes utilizar la función send2trash.send2trash() para eliminar archivos y carpetas. Pero aunque enviar archivos a la papelera de reciclaje te permite recuperarlos más tarde, no liberará espacio en disco como lo hace permanentemente. Si quieres que tu programa libere espacio en disco, utiliza las funciones os y shutil para borrar archivos y carpetas. Tenga en cuenta que la función send2trash() sólo puede enviar ficheros a la papelera de reciclaje; no puede sacar ficheros de ella.

Recorrer un árbol de directorios

Supongamos que quieres renombrar todos los archivos de una carpeta y también todos los archivos de todas las subcarpetas de esa carpeta. Es decir, quieres recorrer el árbol de directorios, tocando cada archivo a medida que avanzas. Escribir un programa para hacer esto podría ser complicado; afortunadamente, Python proporciona una función para manejar este proceso por ti.

Veamos la carpeta *C:\delicious* con su contenido, que se muestra en la Figura 9-1.

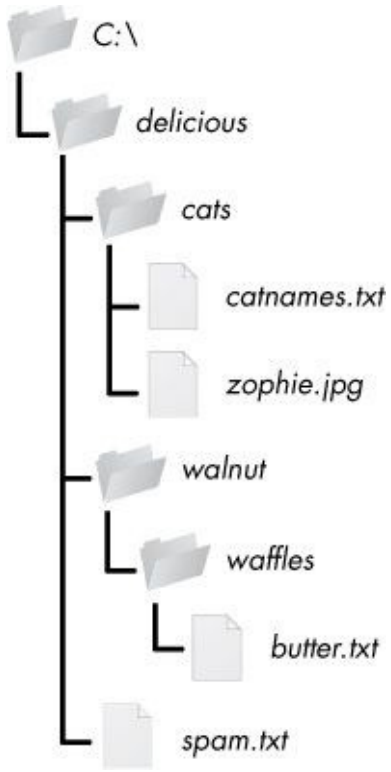


Figura 9-1. Una carpeta de ejemplo que contiene tres carpetas y cuatro archivos

A continuación se muestra un programa de ejemplo que utiliza la función os.walk() en el árbol de directorios de la Figura 9-1:

```
importar os

for folderName, subfolders, filenames in os.walk('C:\\delicious'): print('La
    carpeta actual es '+ folderName)

    para subcarpeta en subcarpetas:
        print('SUBCARPETAS DE '+ folderName+ ': '+ subfolder) for
    filename in filenames:
        print('FILE INSIDE '+ folderName+ ': '+ filename)
```

```
    print('')
```

A la función os.walk() se le pasa un único valor de cadena: la ruta de una carpeta. Puede utilizar os.walk() en una sentencia de bucle for para recorrer un árbol de directorios, de forma a como puede utilizar la función range() para recorrer un rango de números. A diferencia de range(), la función os.walk() devolverá tres valores en cada iteración a través del bucle:

1. Una cadena con el nombre de la carpeta actual
2. Una lista de cadenas de las carpetas de la carpeta actual
3. Una lista de cadenas de los archivos de la carpeta actual

(Por carpeta actual, me refiero a la carpeta de la iteración actual del bucle for. El directorio de trabajo actual del programa *no* es cambiado por os.walk().) Al igual que puedes elegir el nombre de la variable i en el código for i in range(10):, también puedes elegir los nombres de las variables para los tres valores listados anteriormente. Yo suelo utilizar los nombres foldername, subfolders y filenames.

Cuando ejecutes este programa, mostrará lo siguiente:

```
La carpeta actual es C:\delicious
SUBCARPETAS DE C:\delicious: gatos
SUBCARPETAS DE C:\delicious: nuez
ARCHIVO DENTRO DE C:\delicious:
spam.txt

La carpeta actual es C:\delicious\cats
ARCHIVO DENTRO DE C:\delicious\cats:
catnames.txt ARCHIVO DENTRO DE
C:\delicious\cats: zophie.jpg

La carpeta actual es C:\delicious\walnut SUBFOLDER OF
C:\delicious\walnut: waffles

La carpeta actual es C:\delicious\walnut\waffles FILE
INSIDE C:\delicious\walnut\waffles: butter.txt.
```

Dado que os.walk() devuelve listas de cadenas para las variables subcarpeta y nombre de fichero, puede utilizar estas listas en sus propios bucles for. Sustituya las llamadas a la función print() por su propio código personalizado. (O si no necesita una o ambas, elimine los bucles for).

Comprimir archivos con el módulo zipfile

Es posible que conozca los archivos ZIP (con la extensión de archivo *.zip*), que pueden contener el contenido comprimido de muchos otros archivos. Al comprimir un archivo se reduce su tamaño, lo que resulta útil al transferirlo por Internet. Y , ya que un archivo ZIP también puede contener varios archivos y subcarpetas, es una forma práctica de empaquetar varios archivos en uno solo. Este único fichero, llamado *archivo comprimido*, puede adjuntarse a un correo electrónico.

Tus programas Python pueden crear y abrir (o *extraer)* archivos ZIP usando las funciones del módulo `zipfile`. Digamos que tienes un archivo ZIP llamado *ejemplo.zip* que tiene el contenido mostrado en la Figura 9-2.

Puede descargar este archivo ZIP de *http://nostarch.com/automatestuff/* o simplemente seguirlo utilizando un archivo ZIP que ya tenga en su ordenador.

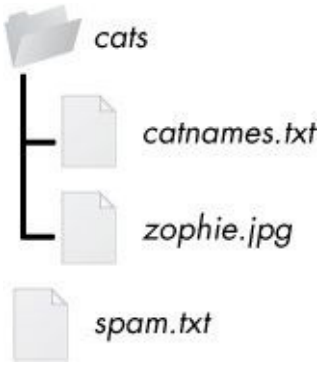


Figura 9-2. Contenido de example.zip

Lectura de archivos ZIP

Para leer el contenido de un archivo ZIP, primero debe crear un objeto ZipFile (observe las letras mayúsculas *Z* y *F*). Los objetos ZipFile son conceptualmente similares a los objetos File que vio que devolvía la función open() en el capítulo anterior: Son valores a través de los cuales el programa interactúa con el archivo. Para crear un objeto ZipFile, llame a la función zipfile.ZipFile(), pasándole una cadena del tipo *.zip*. Tenga en cuenta que zipfile es el nombre del módulo de Python, y ZipFile() es el nombre de la función.

Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar zipfile, os
    >>> os.chdir('C:\\')          # mover a la carpeta con ejemplo.zip
    >>> ejemploZip = zipfile.ZipFile('ejemplo.zip')
    >>> ejemploZip.namelist()
    ['spam.txt', 'cats/', 'cats/catnames.txt', 'cats/zophie.jpg']
    >>> spamInfo= exampleZip.getinfo('spam.txt')
    >>> spamInfo.file_size
    13908
    >>> spamInfo.compress_size
    3828
>>> ❶"¡El archivo comprimido es %sx más pequeño!" % (round(spamInfo.file_size / spamInfo
    .comprimir_tamaño, 2))
    El archivo comprimido es 3,63 veces más pequeño".
    >>> exampleZip.close()
```

Un objeto ZipFile tiene un método namelist() que devuelve una lista de cadenas para todos los archivos y carpetas contenidos en el archivo ZIP. Estas cadenas pueden pasarse al método getinfo() ZipFile para devolver un objeto ZipInfo sobre ese archivo en particular. Los objetos ZipInfo tienen sus propios atributos, como file_size y compress_size en bytes, que contienen enteros del tamaño original del archivo y
tamaño del archivo comprimido, respectivamente. Mientras que un objeto ZipFile representa un comprimido completo, un objeto ZipInfo contiene información útil sobre *un único archivo* comprimido.

El comando ❶ calcula con qué eficiencia se comprime *ejemplo.zip* dividiendo el tamaño del archivo original por el tamaño del archivo comprimido e imprime esta información utilizando una cadena formateada con %s.

Extracción de archivos ZIP

El método extractall() para objetos ZipFile extrae todos los archivos y carpetas de un archivo ZIP al directorio de trabajo actual.

```
>>> importar zipfile, os
    >>> os.chdir('C:³³')          # mover a la carpeta con ejemplo.zip
    >>>  ejemploZip  =  zipfile.ZipFile('ejemplo.zip')
>>> ❶ejemploZip.extractall()
    >>> exampleZip.close()
```

Después de ejecutar este código, el contenido de *ejemplo.zip* se extraerá a *C:³*. Opcionalmente, puede pasar un nombre de carpeta a extractall() para que extraiga los archivos en una carpeta distinta del directorio de trabajo actual. Si la carpeta al método extractall() no existe, será creada. Por ejemplo, si sustituye la llamada a ❶ por ejemploZip.extractall('C:'), el código extraerá los archivos de *ejemplo.zip* en una carpeta *C:\delicioso* recién creada.

El método extract() para objetos ZipFile extraerá un único archivo del archivo ZIP. Continúe con el ejemplo de shell interactivo:

```
>>> exampleZip.extract('spam.txt')
'C:\\spam.txt'
>>>  exampleZip.extract('spam.txt',  'C:\SOME\NEW\\FOLDERS')
'C:\some\new\folders\\spam.txt'
>>> exampleZip.close()
```

La cadena que pase a extract() debe coincidir con una de las cadenas de la lista devuelta por namelist(). Opcionalmente, puede pasar un segundo argumento a extract() para extraer el archivo en una carpeta distinta del directorio de trabajo actual. Si este segundo argumento es una carpeta que aún no existe, creará la carpeta. El valor que devuelve extract() es la ruta absoluta a la que se extrajo el archivo.

Crear y añadir a archivos ZIP

Para crear sus propios archivos ZIP comprimidos, debe abrir el objeto `ZipFile` en *modo de escritura* pasando `'w'` como segundo argumento. (Esto es similar a abrir un archivo de texto en modo escritura pasando `'w'` a la función `open()`). Cuando pasas una ruta al método `write()` de un `ZipFile`, Python comprimirá el archivo en esa ruta y añadirlo al archivo ZIP. El primer argumento del método `write()` es una cadena con el nombre del fichero a añadir. El segundo argumento es el parámetro de *tipo de compresión*, que indica al ordenador qué algoritmo debe utilizar para comprimir los archivos; siempre puede establecer este valor en `zipfile.ZIP_DEFLATED`. (Esto especifica el algoritmo de compresión *deflate*, que funciona bien en todos los tipos de datos). Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar archivo zip
>>> newZip= zipfile.ZipFile('new.zip', 'w')
>>>   newZip.write('spam.txt',    COMPRESS_TYPE=ZIPFILE.ZIP_DEFLATED)
>>> newZip.close()
```

Este código creará un nuevo archivo ZIP llamado *new.zip* que tendrá el contenido comprimido de *spam.txt*.

Tenga en cuenta que, al igual que al escribir en archivos, el modo de escritura borrará todo el contenido existente de un archivo ZIP. Si desea simplemente añadir archivos a un archivo ZIP existente, pase `'a'` como segundo argumento a `zipfile.ZipFile()` para abrir el archivo ZIP en *modo append*.

Proyecto: Renombrar archivos con fechas de estilo americano a fechas de estilo europeo

Supongamos que tu jefe te envía por correo electrónico miles de archivos con fechas de tipo americano (MM- DD-AAAA) en sus nombres y necesita que los renombres a fechas de tipo europeo (DD-MM-AAAA). Esta aburrida tarea te llevaría todo el día hacerla a mano. Escribamos un programa que lo haga.

Esto es lo que hace el programa:

- Busca fechas de estilo americano en todos los nombres de archivo del directorio de trabajo actual.
- Cuando encuentra uno, cambia el nombre del archivo con el mes y el día intercambiados para hacerlo al estilo europeo.

Esto significa que el código tendrá que hacer lo siguiente:

- Cree una expresión regular que pueda identificar el patrón de texto de las fechas de estilo americano.
- Llama a `os.listdir()` para encontrar todos los archivos en el directorio de trabajo.
- Haga un bucle sobre cada nombre de archivo, utilizando la expresión regular para comprobar si tiene una fecha.
- Si tiene una fecha, renombre el archivo con `shutil.move()`.

Para este proyecto, abre una nueva ventana del editor de archivos y guarda tu código como *renombrarFechas.py*.

Paso 1: Crear una Regex para fechas de estilo americano

La primera parte del programa necesitará importar los módulos necesarios y crear una regex que pueda identificar fechas MM-DD-YYYY. Los comentarios de TODO te recordarán lo que queda por escribir en este programa. Escribirlos como TODO hace que sean fáciles de encontrar usando la función de búsqueda CTRL-F de IDLE. Haga que su código se vea como el siguiente:

```
#! python3
    # renameDates.py - Cambia los nombres de archivos con formato de fecha americano MM-
    DD-AAAA # a europeo DD-MM-AAAA.

❶ import shutil, os, re

    # Crear una regex que coincida con los archivos con el formato de fecha americano.
❷ datePattern= re.compile(r"""^(.*?) # todo el texto antes de la fecha
        ((0|1)?\d)-                     # uno o dos dígitos para el mes
        ((0|1|2|3)?\d)-                 # uno o dos dígitos para el día
        ((19|20)\d\d)                   # cuatro dígitos para el año
        (.*?)$                          # todo el texto después de la fecha
➲       """,    re.VERBOSE)

    # TODO: Recorrer los ficheros del directorio de trabajo. #

    TODO: Saltar ficheros sin fecha.

    # TODO: Obtener las diferentes partes del nombre

    de fichero. # TODO: Formar el nombre de archivo al

    estilo europeo.

    # TODO: Obtener las rutas completas y absolutas de los archivos.

    # TODO: Renombrar los archivos.
```

De este , sabes que la función shutil.move() se puede utilizar para renombrar archivos: Sus argumentos son el nombre del archivo a renombrar y el nuevo nombre del archivo. Dado que esta función existe en el módulo shutil, debe importar dicho módulo ❶.

Pero antes de renombrar los archivos, tienes que identificar qué archivos quieres renombrar. Los nombres de archivo con fechas como *spam4-4-1984.txt* y *01-03-2014eggs.zip* deben renombrarse, mientras que los nombres de archivo sin fechas como *littlebrother.epub* pueden ignorarse.

Puede utilizar una expresión regular para identificar este patrón. Después de importar el módulo `re` en la parte superior, llame a `re.compile()` para crear un objeto Regex❷.

Pasando `re.VERBOSE` como segundo argumento ❸ permitirá espacios en blanco y comentarios en la cadena regex para hacerla más legible.

La cadena de expresión regular comienza con `^(.*?)` para coincidir con cualquier texto al principio del nombre de archivo que pueda ir antes de la fecha. El grupo `((0|1)?\d)` coincide con el mes. El primer dígito puede `ser 0` ó `1`, por lo que la expresión regular coincide con `12` para diciembre, pero también con `02` para febrero. Este dígito también es opcional, de modo que el mes `puede ser 04` o `4` para abril. El grupo para el día es `((0|1|2|3)?\d)` y sigue una lógica similar; `3`, `03` y `31` son todos números válidos para días. (Sí, este regex aceptará algunas fechas no válidas como `4-31-2014`, `2-29-2013` y `0- 15-2014`. Las fechas tienen un montón de casos especiales espinosos que pueden ser fáciles de pasar por alto. Pero para simplificar, la regex de este programa funciona suficientemente bien). Aunque 1885 es un año válido, puedes buscar años en los siglos XX o XXI. Esto evitará que su programa haga coincidir accidentalmente nombres de archivo sin fecha con un formato similar a la fecha, como *10-10-1000.txt*.

La parte `(.*?)$` de la expresión regular coincidirá con cualquier texto que aparezca después de la fecha.

Paso 2: Identificar las partes de fecha a partir de los nombres de archivo

A continuación, el programa tendrá que recorrer la lista de cadenas de nombres de archivo devuelta por `os.listdir()` y compararlas con la expresión regular. Los archivos que no contengan una fecha se omitirán. Para los nombres de archivo que tienen una fecha, el texto coincidente se almacenará en varias variables. Rellene los tres primeros TODOs en su programa con el siguiente código:

```
#! python3
    # renameDates.py - Cambia los nombres de archivos con formato de fecha americano MM-
    DD-AAAA # a europeo DD-MM-AAAA.

    --snip--

    # Recorre los ARCHIVOS del directorio de trabajo. for
    amerFilename in os.listdir('.'):
        mo  =  datePattern.search(amerFilename)

        # Saltar ARCHIVOS sin fecha.
❶       si mo== None:
❷           continuar

➲       # Obtener las diferentes PARTES del nombre
        de archivo. beforePart= mo.group(1)
        monthPart=      mo.group(2)
        dayPart    =    mo.group(4)
        yearPart   =    mo.group(6)
        afterPart  = mo.group(8)

    --snip--
```

Si el objeto Match devuelto por el método search() es None❶, entonces el nombre de fichero en amerFilename no coincide con la expresión regular. La sentencia continue ❷ saltará el resto del bucle y pasará al siguiente nombre de fichero.

De lo contrario, las distintas cadenas coincidentes en los grupos de expresiones regulares se almacenan en variables denominadas beforePart, monthPart, dayPart, yearPart y afterPart ❸. Las cadenas de estas variables se utilizarán para formar el nombre de archivo de estilo europeo en el siguiente paso.

Para mantener los números de grupo rectos, intente leer la expresión regular desde el principio y cuente hacia arriba cada vez que encuentre un paréntesis de apertura. Sin pensar en el código, escriba un esquema de la expresión regular. Esto puede ayudarle a

visualizar los grupos. Por ejemplo:

```
datePattern= re.compile(r"""^(1) # todo el texto anterior a la fecha
    (2 (3) )-                    # uno o dos dígitos para el mes
    (4 (5) )-                    # uno o dos dígitos para el día
    (6 (7) )                     # cuatro dígitos para el año
    (8)$                         # todo el texto
    después de la fecha """, re.VERBOSE)
```

Aquí, los números **del 1** al **8** representan los grupos en la expresión regular que escribiste. Hacer un esquema de la expresión regular, con sólo los paréntesis y los números de grupo, puede darle una comprensión más clara de su regex antes de seguir adelante con el resto del programa.

Paso 3: Formar el nuevo nombre de archivo y renombrar los archivos

Como paso final, concatena las cadenas en las variables hechas en el paso anterior con la fecha al estilo europeo: La fecha viene antes del mes. Rellene los tres TODOs restantes en su programa con el siguiente código:

```
#! python3
    # renameDates.py - Cambia los nombres de archivos con formato de fecha americano MM-
    DD-AAAA # a europeo DD-MM-AAAA.

    --snip--

        # Forma el nombre de archivo al estilo europeo.
❶       euroFilename= beforePart+ dayPart+ '-'+ monthPart+ '-'+ yearPart+  afterPart

        # Obtén las rutas completas y absolutas de
        los archivos. absWorkingDir= os.path.abspath('.')
        amerFilename= os.path.join(absWorkingDir, amerFilename)
        euroFilename= os.path.join(absWorkingDir, euroFilename)

        # Renombra los archivos.
❷       print('Renombrar "%s" a "%s"...' % (amerFilename, euroFilename))
➲       #shutil.move(amerFilename, euroFilename)        #descomentar después de probar
```

Almacena la cadena concatenada en una variable llamada euroFilename❶. A continuación, pasa el nombre de archivo original en amerFilename y la nueva variable euroFilename a la función shutil.move() para renombrar el archivo❸.

Este programa tiene la llamada shutil.move() comentada y en su lugar imprime los nombres de los archivos que serán renombrados❷. Ejecutar el programa de esta manera primero puede comprobar que los archivos se renombran correctamente. Luego puedes descomentar la llamada a shutil.move() y ejecutar el programa de nuevo para renombrar los archivos.

Ideas para programas similares

Hay muchas otras razones por las que puede querer renombrar un gran número de archivos.

- Para añadir un prefijo al principio del nombre del archivo, como añadir *spam_* al renombrar *huevos.txt* a *spam_huevos.txt*
- Para cambiar los nombres de archivo con fechas europeas por fechas americanas
- Para eliminar los ceros de archivos *como spam0042.txt*

Proyecto: Copia de seguridad de una carpeta en un archivo ZIP

Digamos que estás trabajando en un proyecto cuyos archivos guardas en una carpeta llamada *C:\lsPythonBook*. Te preocupa perder tu trabajo, así que te gustaría crear "instantáneas" de toda la carpeta en un archivo ZIP. Le gustaría mantener diferentes versiones, por lo que desea que el nombre del archivo ZIP se incremente cada vez que se hace; por ejemplo, *AlsPythonBook_1.zip*, *AlsPythonBook_2.zip*, *AlsPythonBook_3.zip*, y así sucesivamente. Podría hacerlo a mano, pero es bastante molesto, y podría equivocarse accidentalmente en la numeración de los nombres de los archivos ZIP. Sería mucho más sencillo ejecutar un programa que haga esta aburrida tarea por ti.

Para este proyecto, abre una nueva ventana del editor de archivos y guárdalo como *backupToZip.py*.

Paso 1: Averiguar el nombre del archivo ZIP

El código de este programa se colocará en una función llamada backupToZip(). Esto facilitará copiar y pegar la función en otros programas Python que necesiten esta funcionalidad. Al final del programa, la función será llamada para realizar la copia de seguridad. Haz que tu programa se vea así:

```
#! python3
    # backupToZip.py - Copia una carpeta entera y su contenido en # un
    archivo ZIP cuyo nombre se incrementa.

❶ importar zipfile, os

    def backupToZip(carpeta):
        # Copia de seguridad de todo el contenido de "carpeta" en un

        archivo ZIP. carpeta= os.path.abspath(carpeta) # asegúrese de

        carpeta es absoluta

        # Averiguar el nombre de archivo que este código debe usar
        basado en # los archivos que ya existen.
❷       número= 1
➲       mientras sea verdad:
            zipFilename= os.path.basename(folder)+ '_'+ str(number)+ '.zip' if not
            os.path.exists(zipFilename):
                romper
            número= número+ 1

❹       # TODO: Crear el archivo ZIP.

        # TODO: Recorrer todo el árbol de carpetas y comprimir los archivos de cada carpeta.
        print('Hecho.')

    backupToZip('C:\\delicious')
```

Haz primero lo básico: Añade la línea shebang (#!), describe lo que hace el programa e importa los módulos zipfile y os❶.

Defina una función backupToZip() que tome sólo un parámetro, folder. Este parámetro es una cadena de ruta a la carpeta cuyo contenido debe ser . La función determinará qué nombre de archivo usar para el archivo ZIP que creará; luego la función creará el archivo, recorrerá la carpeta, y agregará cada una de las subcarpetas y archivos al archivo ZIP. Escribe comentarios TODO para estos pasos en el código fuente para acordarte de hacerlos más tarde ❹.

La primera parte, nombrar el archivo ZIP, utiliza el nombre base de la ruta absoluta de la carpeta. Si la carpeta de la que se hace la copia de seguridad es *C:\delicious*, el nombre del archivo ZIP debería ser *delicious_N.zip*, donde *N*= 1 es la primera vez que se ejecuta el programa, *N*= 2 es la segunda vez, y así sucesivamente.

Puedes determinar cuál debe ser *N* comprobando si *delicious_1.zip* ya existe, luego comprobando si *delicious_2.zip* ya existe, y así sucesivamente. Usa una variable llamada número para *N* ❷, e increméntala dentro del bucle que llama a os.path.exists() para comprobar si el fichero existe ❸. El primer nombre de archivo inexistente encontrado hará que el bucle se rompa, ya que habrá encontrado el nombre de archivo del nuevo zip.

Paso 2: Crear el nuevo archivo ZIP

A continuación vamos a crear el archivo ZIP. Haz que tu programa tenga el siguiente aspecto:

```
#! python3
    # backupToZip.py - Copia una carpeta entera y su contenido en # un
    archivo ZIP cuyo nombre se incrementa.

    --snip--
        mientras sea verdad:
            zipFilename= os.path.basename(folder)+ '_'+ str(number)+ '.zip' if not
            os.path.exists(zipFilename):
                romper
            número= número+ 1

        # Crear el archivo ZIP.
        print('Creando %s...' % (zipFilename))
❶       backupZip= zipfile.ZipFile(zipFilename, 'w')

        # TODO: Recorrer todo el árbol de carpetas y comprimir los archivos de cada carpeta.
        print('Hecho.')

    backupToZip('C:\\delicious')
```

Ahora que el nombre del nuevo archivo ZIP está almacenado en la variable `zipFilename`, puedes llamar a `zipfile.ZipFile()` para crear realmente el archivo ZIP❶.

Asegúrese de pasar `'w'` como segundo argumento para que el archivo ZIP se abra en modo de escritura.

Paso 3: Recorrer el árbol de directorios y añadir al archivo ZIP

Ahora necesitas usar la función os.walk() para hacer el trabajo de listar cada archivo en la carpeta y sus subcarpetas. Haz que tu programa se parezca a lo siguiente:

```
#! python3
    # backupToZip.py - Copia una carpeta entera y su contenido en # un
    archivo ZIP cuyo nombre se incrementa.

    --snip--

        # Recorre todo el árbol de carpetas y COMPRIME LOS ARCHIVOS de cada carpeta.
❶       para nombrecarpeta, SUBCARPETAS, NOMBRESDEARCHIVO en
            OS.walk(carpeta): print('Añadiendo ARCHIVOS en %S...' %
            (nombrecarpeta))
            # Añade la carpeta actual al archivo ZIP.
❷           backupZip.write(nombrecarpeta)
            # Añade todos los ARCHIVOS de ESTA carpeta al archivo ZIP.
➲           para nombre de archivo en NOMBRES de archivo:
                newBase / OS.path.BASENAME(carpeta)+ '_'
                if filename.STARTSWITH(NEWBASE) and filename.ENDSWITH('.zip')
                    continue    # no hacer copia de seguridad de LOS
                    ARCHIVOS ZIP de copia de seguridad
                backupZip.write(OS.path.join(foldername, filename))
        backupZip.close()
        print('Hecho.')


    backupToZip('C:\\delicious')
```

Puede utilizar os.walk() en un bucle for ❶, y en cada iteración devolverá el nombre de la carpeta actual de la iteración, las subcarpetas de esa carpeta y los nombres de archivo de esa carpeta.

En el bucle for, la carpeta se añade al archivo ZIP ❷. El for anidado puede pasar por cada nombre de archivo de la lista de nombres de archivo ❸. Cada uno de ellos se añade al archivo ZIP, excepto los ZIP de copia de seguridad realizados previamente.

Cuando ejecute este programa, producirá una salida parecida a ésta:

```
Creando delicious_1.zip...
Añadiendo archivos en
C:\delicious...
Añadir archivos en C:\delicious\cats...
Añadir archivos en
C:\delicious\waffles... Añadir archivos
en C:\delicious\walnut...
```

```
Añadiendo archivos en
C:\delicious\walnut\waffles... Hecho.
```

La segunda vez que lo ejecutes, pondrá todos los archivos de *C:\delicious* en un archivo ZIP llamado *delicious_2.zip*, y así sucesivamente.

Ideas para programas similares

Puede recorrer un árbol de directorios y añadir ficheros a archivos ZIP comprimidos en varios otros programas. Por ejemplo, puede escribir programas que hagan lo siguiente:

- Recorrer un árbol de directorios y archivar sólo los ficheros con determinadas extensiones, como por ejemplo
  .txt o *.py*, y nada más
- Recorrer un árbol de directorios y archivar todos los ficheros excepto los *.txt* y *.py*
- Encontrar la carpeta de un árbol de directorios que contiene el mayor número de archivos o la carpeta que ocupa más espacio en disco.

Resumen

Aunque sea un usuario experimentado, probablemente maneje los archivos manualmente con el ratón y el teclado. Los exploradores de archivos modernos facilitan el trabajo con unos pocos archivos. Pero a veces necesitarás realizar una tarea que te llevaría horas utilizando el explorador de archivos de tu ordenador.

Los módulos `os` y `shutil` ofrecen funciones para copiar, mover, renombrar y borrar ficheros. Cuando borres ficheros, puede que quieras usar el módulo `send2trash` para mover los ficheros a la papelera de reciclaje o a la papelera en lugar de borrarlos permanentemente. Y cuando escribas programas que manejen ficheros, es una buena idea comentar el código que hace la copia/movimiento/renombrado/borrado y añadir una llamada a `print()` en su lugar para que puedas ejecutar el programa y verificar exactamente lo que hará.

A menudo necesitará realizar estas operaciones no sólo en los archivos de una carpeta, sino también en cada carpeta de esa carpeta, en cada carpeta de esas carpetas, y así sucesivamente. La función `os.walk()` se encarga de este recorrido por las carpetas para que usted pueda concentrarse en lo que su programa necesita hacer con los archivos que .

El módulo `zipfile` te ofrece una forma de comprimir y extraer ficheros en archivos *.zip* a través de Python. Combinado con las funciones de manejo de archivos de `os` y `shutil`, `zipfile` facilita el empaquetado de varios archivos desde cualquier lugar de tu disco duro. Estos archivos *.zip* son mucho más fáciles de subir a sitios web o enviar como adjuntos de correo electrónico que muchos archivos separados.

Los capítulos anteriores de este libro le han proporcionado código fuente para que lo copie. Pero cuando escribas tus propios programas, probablemente no te saldrán perfectos a la primera. El siguiente capítulo se centra en algunos módulos de Python que te ayudarán a analizar y depurar tus programas para que puedas conseguir rápidamente que funcionen correctamente.

Preguntas prácticas

P: 1. ¿Cuál es la diferencia entre shutil.copy() y shutil.copytree()?

P: 2. ¿Qué función se utiliza para renombrar archivos?

P: 3. ¿Cuál es la diferencia entre las funciones delete en los módulos send2trash y shutil?

P: 4. Los objetos ZipFile tienen un método close() igual que el método close() de los objetos File. ¿Qué ZipFile es equivalente al método open() de los objetos File?

Proyectos prácticos

Para practicar, escribe programas que realicen las siguientes tareas.

Copia selectiva

Escribe un programa que recorra un árbol de carpetas y busque archivos con una determinada extensión (como *.pdf* o *.jpg*). Copia estos archivos de la ubicación en la que se encuentren a una nueva carpeta.

Eliminar archivos innecesarios

No es raro que algunos archivos o carpetas innecesarios pero enormes ocupen la mayor parte del espacio del disco duro. Si estás intentando liberar espacio en tu ordenador, sacarás el máximo partido eliminando los archivos no deseados más voluminosos. Pero primero tienes que encontrarlos.

Escribe un programa que recorra un árbol de carpetas y busque archivos o carpetas excepcionalmente grandes, es decir, que tengan un tamaño superior a 100 MB. (Recuerda que para obtener el tamaño de un archivo, puedes utilizar `os.path.getsize()` del módulo `os`). Imprime estos archivos con su ruta absoluta en la pantalla.

Rellenar los huecos

Escriba un programa que busque todos los archivos con un prefijo determinado, como *spam001.txt*, *spam002.txt*, etc., en una misma carpeta y localice los huecos en la numeración (por ejemplo, si hay un *spam001.txt* y un *spam003.txt* pero no *un spam002.txt*). Haz que el programa cambie el nombre de todos los archivos posteriores para cerrar esta brecha.

Como reto añadido, escriba otro programa que pueda insertar huecos en ficheros numerados para poder añadir un nuevo fichero.

Capítulo 10. Depuración Depuración

Ahora que sabes lo suficiente como para escribir programas más complicados, puede que empieces a encontrar fallos no tan simples en ellos. Este capítulo cubre algunas herramientas y técnicas para encontrar la causa raíz de los fallos en tu programa para ayudarte a más rápido y con menos esfuerzo.

Parafraseando un viejo chiste entre programadores, "Escribir código supone el 90 por ciento de la programación. Depurar código supone el otro 90 por ciento".

Tu ordenador sólo hará lo que tú le digas que haga; no leerá tu mente y hará lo que tú *pretendías* que hiciera. Incluso los programadores profesionales crean errores todo el tiempo, así que no te desanimes si tu programa tiene algún problema.

Afortunadamente, existen algunas herramientas y técnicas para identificar qué está haciendo exactamente tu código y dónde está fallando. En primer lugar, examinaremos el registro y las aserciones, dos características que pueden ayudarte a detectar errores de forma temprana. En general, cuanto antes se detecten los errores, más fácil será solucionarlos.

En segundo lugar, verás cómo utilizar el depurador. El depurador es una característica de IDLE que ejecuta un programa una instrucción a la vez, dándole la oportunidad de inspeccionar los valores de las variables mientras su código se ejecuta, y hacer un seguimiento de cómo los valores cambian en el transcurso de su programa. Esto es mucho más lento que ejecutar el programa a toda velocidad, pero es útil para ver los valores reales en un programa mientras se ejecuta, en lugar de deducir lo que los valores podrían ser desde el código fuente.

Planteamiento de excepciones

Python lanza una excepción cada vez que intenta ejecutar código inválido. En el Capítulo 3, leíste sobre cómo manejar las excepciones de Python con sentencias try y except para que tu programa pueda recuperarse de las excepciones que tú anticipadas. Pero también puedes lanzar tus propias excepciones en tu código. Lanzar una excepción exception es una forma de decir: "Deja de ejecutar el código en esta función y mueve la ejecución del programa a la sentencia except".

Las excepciones se plantean con una sentencia raise. En código, una raise consiste en lo siguiente:

- La palabra clave aumentar
- Una llamada a la función Exception()
- Una cadena con un mensaje de error útil pasado a la función Exception() Por

ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> raise Exception('ESTE ES el mensaje de error.')
Traceback (llamada más reciente):
  File "<pyshell#191>", line 1, in <module>
    raise Exception('Este es el mensaje de error.')
Exception: Este es el mensaje de error.
```

Si no hay sentencias try y except que cubran la sentencia raise que lanzó la excepción, el programa simplemente se bloquea y muestra el mensaje de error de la excepción.

A menudo es el código que llama a la función, no la propia función, el que sabe cómo manejar una expectativa. Por lo tanto, es común ver una sentencia raise dentro de una función y las sentencias try y except en el código que llama a la función. Por ejemplo, abra una nueva ventana del editor de archivos, introduzca el siguiente código y guarde el programa como *boxPrint.py*:

```
def boxPrint(símbolo, anchura, altura): if
        len(símbolo) != 1:
❶           raise Exception('Symbol must be a single character
        string.') if width<= 2:
❷           raise Exception('La anchura debe ser mayor que 2.') if
        height<= 2:
➲           raise Exception('La altura debe ser mayor que 2.')
        print(símbolo  anchura)
        para i en rango(altura - 2):
```

```
            print(símbolo+ (' ' (anchura - 2))+
        símbolo) print(símbolo    anchura)

    para sym, w, h en (('', 4, 4), ('0', 20, 5), ('x', 1, 3), ('ZZ', 3, 3)):
        Inténtalo:
                boxPrint(sym, w, h)
❹       excepto Excepción como err:
❺           print('Se ha producido una excepción: '+ str(err))
```

Aquí hemos definido una función boxPrint() que toma un carácter, una anchura y altura, y utiliza el carácter para hacer una pequeña imagen de una caja con esa anchura y altura. Esta forma de caja se imprime en la pantalla.

Digamos que queremos que el carácter sea un único carácter, y que la anchura y la altura sean mayores que 2. Añadimos sentencias if para lanzar excepciones si no se cumplen estos requisitos. Más tarde, cuando llamemos a boxPrint() con varios argumentos, nuestro try/except manejará los argumentos inválidos.

Este programa utiliza la forma except Exception as err de la sentencia except❹.

Si un objeto Exception es devuelto por boxPrint() ❶❷❸, esta sentencia except lo almacenará en una variable llamada err. El objeto Exception puede entonces convertirse en una cadena pasándolo a str() para producir un mensaje de error ❺ amigable para el usuario. Cuando ejecutes este *boxPrint.py*, la salida tendrá este aspecto:

```
****


****
0000000000000000000 00
0                   0
0                   0
000000000000000000
Se ha producido una excepción: La anchura debe ser mayor que 2.
Se ha producido una excepción: El símbolo debe ser una cadena de un solo carácter.
```

Usando las sentencias try y except, puedes manejar los errores con más gracia en lugar de dejar que todo el programa se bloquee.

Obtener el Traceback como cadena

Cuando Python encuentra un error, produce un tesoro de información de error llamado *traza*. El rastreo incluye el mensaje de error, el número de la línea que causó el error y la secuencia de llamadas a la función que condujo al error. Esta secuencia de llamadas se denomina *pila de llamadas*.

Abre una nueva ventana del editor de archivos en IDLE, introduce el siguiente programa y guárdalo como *errorEjemplo.py*:

```
def spam():
    bacon()
def bacon():
    raise Exception('Este es el mensaje de error.')

spam()
```

Cuando ejecutes *errorExample.py*, la salida tendrá este aspecto:

```
Traceback (llamada más reciente):
  File "errorExample.py", line 7, in <module> spam()
  File "errorExample.py", line 2, in spam
    bacon()
  File "errorExample.py", line 5, in bacon
    raise Exception('Este es el mensaje de error.')
Exception: Este es el mensaje de error.
```

A partir del rastreo, se puede ver que el error se produjo en la línea 5, en la función bacon(). Esta llamada particular a bacon() vino de la línea 2, en la función spam(), que a su vez fue llamada en la línea 7. En programas donde las funciones pueden ser llamadas desde múltiples lugares, la pila de llamadas puede ayudarte a determinar qué llamada condujo al error.

La traza es mostrada por Python cada vez que una excepción no es tratada. Pero también puedes obtenerla como una cadena llamando a traceback.format_exc(). Esta función es útil si quieres la información de la traza de una pero también quieres una sentencia except para manejar la excepción con gracia. Necesitará importar el módulo traceback de Python antes de llamar a esta función.

Por ejemplo, en lugar de bloquear el programa justo cuando se produce una excepción, puedes escribir la información de rastreo en un archivo de registro y mantener el programa en ejecución. Puede consultar el archivo de registro más tarde, esté listo para depurar su programa.

programa. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar rastreo
>>> Inténtalo:
        raise Exception('ESTE ES el mensaje de error.')
excepto
:       errorFile = open('errorInfo.txt', 'w')
        errorFile.write(traceback.format_exc())     errorFile.close()
        print('La información de rastreo se escribió en errorInfo.txt.')

116
La información de rastreo se ha escrito en errorInfo.txt.
```

El 116 es el valor de retorno del método `write()`, ya que se escribieron 116 caracteres en el archivo. El texto de rastreo se escribió en *errorInfo.txt*.

```
Traceback (llamada más reciente):
  File "<pyshell#28>", line 2, in <module> Excepción:
Este es el mensaje de error.
```

Afirmaciones

Una *aserción* es una comprobación de sanidad para asegurarse de que su código no está haciendo algo obviamente incorrecto. Estas comprobaciones se realizan mediante sentencias assert. Si la comprobación falla, se lanza una excepción AssertionError. En código, una sentencia assert consiste en lo siguiente:

- La palabra clave assert
- Una condición (es decir, una expresión que se evalúa como True o False)
- Una coma
- Una cadena para mostrar cuando la condición es False

Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> podBayDoorStatus= 'abierto'
>>> assert podBayDoorStatus== 'open', 'Las puertas del módulo deben estar "abiertas".'
>>> podBayDoorStatus= 'Lo siento, Dave. Me temo que no puedo hacerlo.
>>> assert podBayDoorStatus== 'open', 'Las puertas del módulo deben estar "abiertas".'
Traceback (llamada más reciente):
  File "<pyshell#10>", line 1, in <module>
    assert podBayDoorStatus== 'open', 'Las puertas del módulo deben estar "abiertas".'
AssertionError: Las puertas del módulo deben estar "abiertas".
```

Aquí hemos establecido podBayDoorStatus a 'open', por lo que a partir de ahora, esperamos que el valor de esta variable sea 'open'. En un programa que utilice esta variable, puede que hayamos escrito mucho código asumiendo que el valor es 'open', código que depende de que sea 'open' para funcionar como esperamos. Así que añadimos una aserción para asegurarnos de que estamos en lo cierto al asumir que podBayDoorStatus está 'abierto'.
Aquí, incluimos el mensaje 'Las puertas de la bahía del pod necesitan estar "abiertas".' así será fácil ver qué está mal si la aserción falla.

Después, digamos que cometemos el error obvio de asignar a podBayDoorStatus otro valor, pero no nos damos cuenta entre tantas líneas de código. La aserción capta este error y nos dice claramente lo que está mal.

En lenguaje llano, una sentencia assert dice: "Afirmo que esta condición es cierta, y si no, hay un error en alguna parte del programa". A diferencia de las excepciones, tu código *no* debería manejar las sentencias assert con try y except; si una assert falla, tu programa *debería* bloquearse. Fallando rápido de esta manera, acortas el tiempo entre la causa original del y cuando te das cuenta del fallo por primera vez.

Esto reducirá la cantidad de código que tendrá que comprobar antes de encontrar el código que está causando el fallo.

Las aserciones son para errores de programador, no de usuario. Para errores de los que se puede recuperar (como que no se encuentre un archivo o que el usuario introduzca datos no válidos), lanza una excepción en lugar de detectarlo con una sentencia assert.

Uso de una aserción en una simulación de semáforo

Supongamos que estás construyendo un programa de simulación de semáforos. La estructura de datos que representa los semáforos en una intersección es un diccionario con las claves 'ns' y 'ew', para los semáforos orientados de norte a sur y de este a oeste, respectivamente. Los valores de estas claves serán una de las cadenas "verde", "amarillo" o "rojo". El código sería algo así

```
market_2nd= {'ns': 'verde', ew': 'red'}
mission_16th= {'ns': 'rojo', ew': 'green'}
```

Estas dos variables serán para las intersecciones de Market Street y 2nd Street, y Mission Street y 16th Street. Para iniciar el proyecto, se desea escribir una función switchLights(), que tomará un diccionario de intersección como argumento y conmutará las luces.

Al principio, se podría pensar que switchLights() simplemente debería cambiar cada luz al siguiente color de la secuencia: Cualquier valor 'verde' debería cambiar a 'amarillo', los valores 'amarillo' deberían cambiar a 'rojo', y los valores 'rojo' deberían cambiar a 'verde'. El código para implementar esta idea podría verse así:

```
def cambiarLuces(semáforo):
    para tecla en semáforo.teclas():
        si semáforo[tecla]== 'verde':
            semáforo[tecla]= 'amarillo'
        elif semáforo[tecla]==
            'amarillo': semáforo[tecla]=
            'rojo'
        elif semáforo[tecla]== 'rojo':
            semáforo[tecla]= 'verde'


switchLights(mercado_2)
```

Puede que ya veas el problema con este código, pero imaginemos que escribiste el resto del código de la simulación, de miles de líneas, sin darte cuenta. Cuando finalmente ejecutas la simulación, el programa no se bloquea, ¡pero tus coches virtuales sí!

Como ya has escrito el resto del programa, no tienes ni idea de dónde puede estar el fallo. Tal vez esté en el código que simula los coches o en el código que simula los conductores virtuales. Podría llevar horas rastrear el fallo hasta la función switchLights().

Pero si al escribir switchLights() hubieras añadido una aserción para comprobar que

al menos una de las luces está siempre en rojo, podría haber incluido lo siguiente en la parte inferior de la función:

```
assert 'rojo' in semáforo.valores(), '¡Ninguna de las luces es roja! '+ str(semáforo)
```

Con esta aserción en su lugar, su programa se bloquearía con este mensaje de error:

```
Traceback (llamada más reciente):
      File "carSim.py", line 14, in <module>
        switchLights(mercado_2nd)
      File "carSim.py", line 13, in switchLights
        assert 'rojo' in semáforo.valores(), '¡Ninguna de las luces es roja! '+ str(semáforo)
ssertionError: ¡Ninguna de las dos luces es roja! {'ns': 'amarillo', 'ew': 'green'}
```

La línea importante aquí es el AssertionError❶. Mientras que su programa de estrellarse no es lo ideal, inmediatamente señala que una comprobación de cordura falló: Ninguna de las direcciones del tráfico tiene un semáforo en rojo, lo que significa que el tráfico podría ir en ambas direcciones. Fallando rápido al principio de la ejecución del programa, puedes ahorrarte mucho esfuerzo de depuración en el futuro.

Desactivación de aserciones

Las aserciones se pueden desactivar pasando la opción `-O` cuando se ejecuta Python. Esto es bueno para cuando has terminado de escribir y probar tu programa y no quieres que se ralentice realizando comprobaciones de sanidad (aunque la mayoría de las veces las sentencias `assert` no causan una diferencia de velocidad notable). Las aserciones son para el desarrollo, no para el producto final. Para cuando entregue su
a otra persona para que lo ejecute, debería estar libre de errores y no necesitar las comprobaciones de sanidad. Vea el Apéndice B para más detalles sobre cómo lanzar sus programas probablemente no locos con la opción `-O`.

Registro

Si alguna vez has puesto una sentencia `print()` en tu código para mostrar el valor de alguna variable mientras tu programa se está ejecutando, has usado una forma de *registro* para depurar tu código. El registro es una gran manera de entender lo que está sucediendo en tu programa y en qué orden está sucediendo. El módulo de `registro de` Python hace que sea fácil crear un registro de los mensajes personalizados que escribes. Estos mensajes de registro
describirá cuándo la ejecución del programa ha alcanzado la llamada a la función de registro y listará cualquier variable que especificado en ese . Por otro lado, un mensaje de registro ausente indica que una parte del código se ha omitido y nunca se ha ejecutado.

Uso del módulo de registro

Para permitir que el módulo de registro muestre mensajes de registro en la pantalla mientras se ejecuta el programa, copie lo siguiente en la parte superior del programa (pero debajo de la línea #! python shebang):

```
importar registro
logging.basicConfig(level=logging.DEBUG, format=' %(asctime)s - %(levelname)s
- %(mensaje)s")
```

No necesitas preocuparte demasiado sobre cómo funciona esto, pero básicamente, cuando Python registra un evento, crea un objeto LogRecord que contiene información sobre ese evento. La función basicConfig() del módulo de registro te permite especificar qué detalles sobre el objeto LogRecord quieres ver y cómo quieres que se muestren esos detalles.

Supongamos que has escrito una función para calcular el *factorial* de un número. En matemáticas, el factorial 4 es 1 × 2 × 3 × 4, o 24. El factorial 7 es 1 × 2 × 3 × 4 × 5 × 6 × 7, o 5.040. Abre una nueva ventana del editor de archivos e introduce el siguiente código. un error, pero también introducirás varios mensajes de registro para ayudarte a averiguar qué está fallando. Guarda el programa como *factorialLog.py*.

```
importar registro
logging.basicConfig(level=logging.DEBUG, format=' %(asctime)s - %(levelname)s
-%(mensaje)s") logging.debug('Inicio
del programa")

def factorial(n):
    logging.debug('Inicio de factorial(%s%%)" %
    (n)) total= 1
    para i en rango(n+
        1): total *= i
        logging.debug('i es "+ str(i)+ ", total es "+ str(total))
    logging.debug('Fin de factorial(%s%%)" % (n))
    devolución total

print(factorial(5)) logging.debug('Fin
del programa")
```

Aquí, usamos la función logging.debug() cuando queremos imprimir información de registro. Esta función debug() llamará a basicConfig(), y se imprimirá una línea de información. Esta información estará en el formato que especificamos en basicConfig() e incluirá los mensajes que pasamos a debug(). La dirección

`print(factorial(5))` es parte del programa original, por lo que el resultado se muestra incluso si los mensajes de registro están desactivados.

La salida de este programa tiene el siguiente aspecto:

```
2015-05-23  16:20:12,664 - DEBUG - Inicio del programa
2015-05-23  16:20:12,664 - DEBUG - Inicio de factorial(5)
2015-05-23  16:20:12,665 -DEBUG - i es 0, total es 0
2015-05-23  16:20:12,668 -DEBUG - i es 1, total es 0
2015-05-23  16:20:12,670 -DEBUG - i es 2, total es 0
2015-05-23  16:20:12,673 -DEBUG - i es 3, total es 0
2015-05-23  16:20:12,675 -DEBUG - i es 4, total es 0
2015-05-23  16:20:12,678 -DEBUG - i es 5, total es 0
2015-05-23  16:20:12,680 - DEBUG - Fin de factorial(5)
0
2015-05-23  16:20:12,684 - DEBUG - Fin del programa
```

La función `factorial()` devuelve `0` como factorial de `5`, lo que no es correcto. El bucle `for` debería estar multiplicando el valor en `total` por los números del `1` al `5`. Pero los mensajes de registro mostrados por `logging.debug()` muestran que la variable `i` está comenzando en `0` en lugar de `1`. Dado que cero veces cualquier cosa es cero, el resto de las iteraciones también tienen el valor incorrecto para el `total`. Los mensajes de registro proporcionan un rastro de migas de pan que pueden ayudarle a averiguar cuándo empezaron a ir mal las cosas.

Cambie la línea `for i in range(n+1):` por `for i in range(1, n+1):`, y ejecute de nuevo el programa. La salida tendrá este aspecto:

```
2015-05-23  17:13:40,650 - DEBUG - Inicio del programa
2015-05-23  17:13:40,651 - DEBUG - Inicio de factorial(5)
2015-05-23  17:13:40,651 -DEBUG - i es 1, total es 1
2015-05-23  17:13:40,654 -DEBUG - i es 2, total es 2
2015-05-23  17:13:40,656 -DEBUG - i es 3, total es 6
2015-05-23  17:13:40,659 -DEBUG - i es 4, total es 24
2015-05-23  17:13:40,661 -DEBUG - i es 5, total es 120
2015-05-23  17:13:40,661 - DEBUG - Fin de factorial(5)
120
2015-05-23  17:13:40,666 - DEBUG - Fin del programa
```

La llamada a `factorial(5)` devuelve correctamente `120`. Los mensajes de registro mostraban lo que ocurría dentro del bucle, lo que llevaba directamente al error.

Puedes ver que las llamadas a `logging.debug()` imprimen no sólo las cadenas que se les pasan, sino también una marca de tiempo y la palabra *DEBUG*.

No depurar con print()

Escribir `import logging` y `logging.basicConfig(level=logging.DEBUG, format="%(asctime)s - %(levelname)s - %(message)s")` es algo difícil de manejar. Es posible que desee utilizar llamadas `print()` en su lugar, no caiga en esta tentación! Una vez que hayas terminado de depurar, terminarás gastando mucho tiempo eliminando las llamadas `print()` de tu código para cada mensaje de registro. Puede que incluso eliminar accidentalmente algunas llamadas a `print()` que se estaban utilizando para mensajes que no eran de registro. Lo bueno de los mensajes de registro es que eres libre de llenar tu programa con tantos como quieras, y siempre puedes desactivarlos más tarde añadiendo una única llamada a `logging.disable(logging.CRITICAL)`. A diferencia de `print()`, el módulo `logging` facilita el cambio entre mostrar y ocultar los mensajes de registro.

Los mensajes de registro están pensados para el programador, no para el usuario. Al usuario no le importará el contenido de algún valor del diccionario que necesite ver para ayudar con la depuración; utilice un mensaje de registro para algo así. Para mensajes que el usuario quiera ver, como *Archivo no encontrado* o *Entrada inválida, por favor introduzca un número*, debería usar una llamada a `print()`. Usted no quiere privar al usuario de útiles información después de desactivar los mensajes de registro.

Niveles de registro

Los niveles de registro permiten clasificar los mensajes de registro por importancia. Existen cinco niveles de registro, descritos en la Tabla 10-1 de menor a mayor importancia. Los mensajes se pueden registrar en cada nivel utilizando una función de registro diferente.

Tabla 10-1. Niveles de registro en Python

Nivel	Función de registro	Descripción
DEBUG	logging.debug()	El nivel más bajo. Se utiliza para pequeños detalles. Por lo general, estos mensajes sólo se utilizan para diagnosticar problemas.
INFO	logging.info()	Se utiliza para registrar información sobre eventos generales en su programa o confirmar que las cosas están funcionando en su punto en el programa.
ADVERTENCIA	logging.warning()	Se utiliza para indicar un problema potencial que no impide el funcionamiento del programa pero que podría hacerlo en el futuro .
ERROR	logging.error()	Se utiliza para registrar un error que ha provocado que el programa no haga algo.
CRÍTICA	logging.critical()	El nivel más alto. Se utiliza para indicar un error fatal que ha provocado o está a punto de provocar que el programa deje de ejecutarse por completo.

Su mensaje de registro se pasa como una cadena a estas funciones. Los niveles de registro son sugerencias. En última instancia, es usted quien decide en qué categoría entra su mensaje de registro. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar registro
>>> logging.basicConfig(level=logging.DEBUG, format=' %(asctime)s -
%(nombrenivel)s - %(mensaje)s")
>>> logging.debug('Algunos DETALLES DE depuración.')
2015-05-18 19:04:26,901 - DEBUG - Algunos detalles de depuración.
>>> logging.info('El módulo de registro ESTÁ funcionando.')
2015-05-18 19:04:35,569 - INFO - El módulo de registro está funcionando.
>>> logging.warning('Está a punto de registrarse un mensaje de error.')
2015-05-18 19:04:56,843 - ADVERTENCIA - Un mensaje de error está a punto de ser registrado.
>>> logging.error('SE HA UN ERROR.')
2015-05-18 19:05:07,737 - ERROR - Se ha producido un error.
>>> logging.critical('¡El programa no puede recuperarse!')
2015-05-18 19:05:45,794 - CRÍTICO - ¡El programa no puede recuperarse!
```

La ventaja de los niveles de registro es que puede cambiar la prioridad del registro.

que desea ver. Pasando logging.DEBUG al argumento de la palabra clave level de la función basicConfig() se mostrarán los mensajes de todos los niveles de registro (siendo DEBUG el nivel más bajo). Pero después de desarrollar su programa un poco más, puede estar interesado sólo en los errores. En ese caso, puede establecer el argumento level de basicConfig() a logging.ERROR. Esto mostrará sólo los mensajes de ERROR y CRÍTICO y omitirá los mensajes de DEBUG, INFO y WARNING.

Desactivar el registro

Después de haber depurado tu programa, probablemente no quieras todos estos mensajes de registro abarrotando la . La función logging.disable() los desactiva para que no tengas que entrar en tu programa y eliminar todas las llamadas de registro a mano. Simplemente pasa a logging.disable() un nivel de registro, y suprimirá todos los mensajes de registro en ese nivel o inferior. Así que si quieres desactivar el registro por completo, sólo tienes que añadir logging.disable(logging.CRITICAL) a tu programa. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar registro
>>> logging.basicConfig(level=logging.INFO, format=' %(asctime)s -
%(nombrenivel)s - %(mensaje)s")
>>> logging.critical('¡Error crítico! ¡Error crítico!')
2015-05-22 11:10:48,054 - CRÍTICO - ¡Error crítico! ¡Error crítico!
>>> logging.DISABLE(logging.CRITICAL)
>>> logging.critical('¡Error crítico! ¡Error crítico!')
>>> logging.error('¡Error! ¡Error!')
```

Dado que logging.disable() deshabilitará todos los mensajes después de ella, probablemente querrás añadirla cerca de la línea de código import logging en tu programa. De esta manera, usted puede encontrar fácilmente a comentar o descomentar esa llamada para activar o desactivar los mensajes de registro, según sea necesario.

Registro en un archivo

En lugar de mostrar los mensajes de registro en la pantalla, puede escribirlos en un archivo de texto. La función logging.basicConfig() toma como argumento la palabra clave filename:

```
importar registro
logging.basicConfig(nombrearchivo='miProgramaLog.txt', nivel=logging.DEBUG, formato '=
%(asctime)s - %(levelname)s - %(message)s'')
```

Los mensajes de registro se guardarán en *miProgramaLog.txt*. Aunque los mensajes de registro son útiles, pueden saturar la pantalla y dificultar la lectura de la salida del programa. Escribir los mensajes de registro en un archivo mantendrá su pantalla despejada y almacenará los mensajes para que pueda leerlos después de ejecutar el programa. Puede abrir este archivo de texto en cualquier editor de texto, como Notepad o TextEdit.

Depurador de IDLE

El *depurador* es una característica de IDLE que te permite ejecutar tu programa línea a línea. El depurador ejecutará una sola línea de código y luego esperará a le digas que continúe. Ejecutando tu programa "bajo el depurador" de esta manera, puedes tomarte todo el tiempo que quieras para examinar los valores de las variables en cualquier punto dado durante la vida del programa. Esta es una herramienta valiosa para localizar errores.

Para activar el depurador de IDLE, haz clic en **Debug▸ Debugger** en la ventana del intérprete de comandos interactivo. Esto mostrará la ventana de Control de Depuración, que se parece a la Figura 10-1.

Cuando aparezca la ventana Control de depuración, seleccione las cuatro casillas de verificación **Pila**, **Locales**, **Fuente** y **Globales** para que la ventana muestre el conjunto completo de información de depuración. Mientras se muestra la ventana Control de depuración, cada vez que ejecute un programa desde el editor de archivos, el depurador detendrá la ejecución antes de la primera instrucción y mostrará lo siguiente:

- La línea de código que se va a ejecutar
- Una lista de todas las variables locales y sus valores
- Una lista de todas las variables globales y sus valores

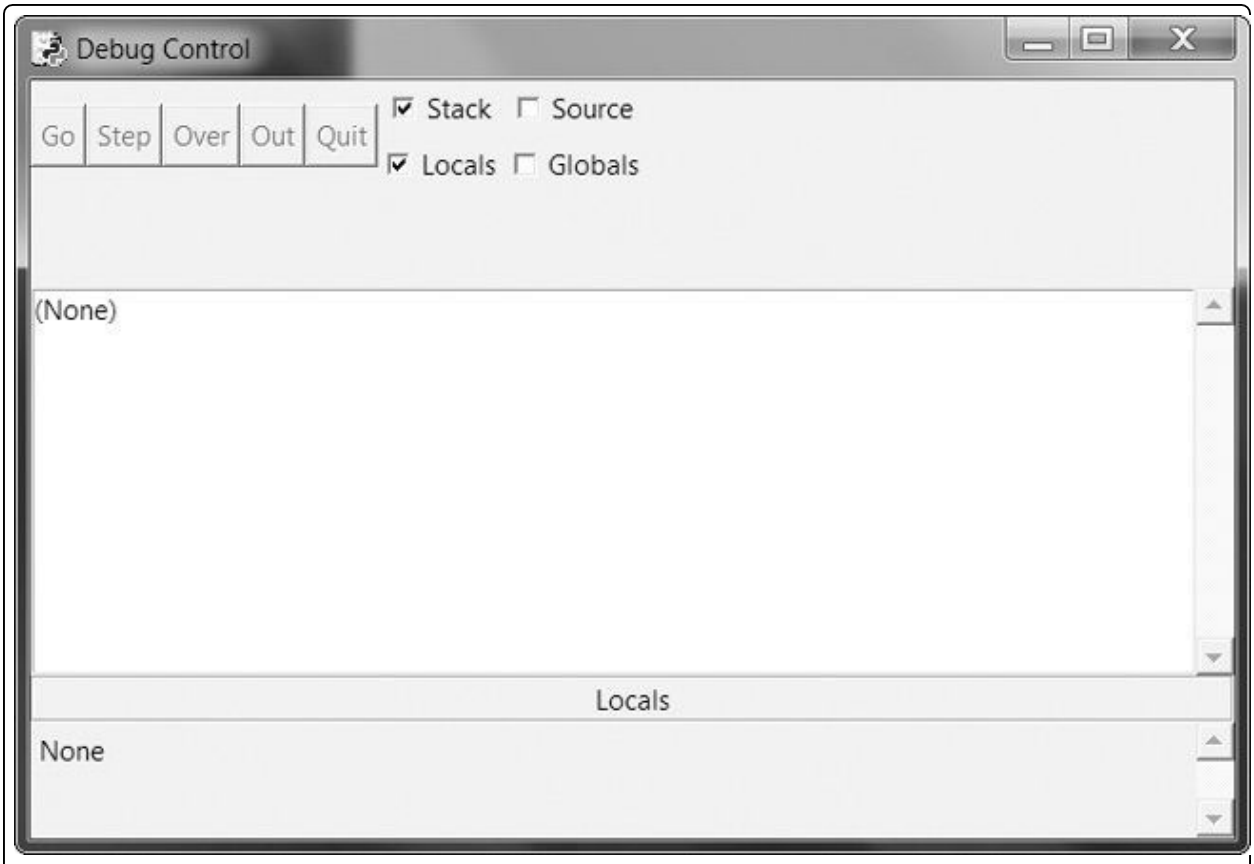


Figura 10-1. Ventana de control de depuración

Observará que en la lista de variables globales hay varias variables que no ha definido, como por ejemplo **__builtins__**, **__doc__**, **__archivo__** etc. Estas son variables que Python establece automáticamente cada vez que ejecuta un programa. El significado de estas variables está más allá del alcance de este libro, y puedes ignorarlas cómodamente.

El programa permanecerá en pausa hasta que pulse uno de los cinco botones de la ventana Control de depuración: Ir, Paso, Sobre, Fuera o Salir.

Ir a

Al hacer clic en el botón Ir, el programa se ejecutará normalmente hasta que termine o alcance un *punto* de interrupción. (Los puntos de interrupción se describen más adelante en este capítulo.) Si ha terminado de depurar y desea que el programa continúe normalmente, haga clic en el botón **Ir**.

Paso

Al hacer clic en el botón Paso, el depurador ejecutará la siguiente línea de código y volverá a hacer una pausa. La lista de variables globales y locales de la ventana de control de depuración se actualizará si sus valores cambian. Si la siguiente línea de código es una llamada a una función, el depurador "entrará" en esa función y saltará a la primera línea de código de esa función.

En

Al hacer clic en el botón Over se ejecutará la siguiente línea de código, de forma similar al botón Step. Sin embargo, si la siguiente línea de código es una llamada a una función, el botón Over "pasará por encima" del código de la función. El código de la función se ejecutará a toda velocidad, y el depurador hará una pausa tan pronto como regrese la llamada a la función. Por ejemplo, si la siguiente línea de código es una llamada a `print()`, no realmente te importa el código dentro de la función incorporada `print()`; sólo quieres que la cadena que le pases se imprima en la pantalla. Por esta razón, usar el botón Over es más común que el botón Step.

En

Al hacer clic en el botón Salir, el depurador ejecutará líneas de código a toda velocidad hasta que regrese de la función actual. Si has entrado en una llamada a función con el botón Step y ahora simplemente quieres seguir ejecutando instrucciones hasta que vuelvas a salir, haz clic en el botón **Out** para "salir" de la llamada a función actual.

Deje de

Si desea detener la depuración por completo y no molestarse en seguir ejecutando el resto del programa, haga clic en el botón **Salir**. El botón Salir terminará inmediatamente el programa. Si desea volver a ejecutar su programa normalmente, seleccione **Depurar▸ Depurador** de nuevo para desactivar el depurador.

Depuración de un programa de suma de números

Abra una nueva ventana del editor de archivos e introduzca el siguiente código:

```
print('Introduce el primer número a
añadir:') first= input()
print('Introduce el segundo número a
añadir:') second= input()
print('Introduce el tercer número a
añadir:') third= input()
print('La suma es '+ primero+ segundo+ tercero)
```

Guárdalo como *buggyAddingProgram.py* y ejecútalo primero sin el depurador activado. El programa mostrará algo como esto:

```
Introduzca el primer número a añadir:
5
Introduzca el segundo número a añadir:
3
Introduzca el tercer número a añadir:
42
La suma es 5342
```

El programa no se ha bloqueado, pero la suma está obviamente mal. Vamos a habilitar la ventana de control de depuración y de nuevo, esta vez bajo el depurador.

Cuando pulsa F5 o selecciona **Ejecutar▸ Ejecutar Módulo** (con **Depurar▸ Depurador** activado y las cuatro casillas de verificación de la ventana de Control de Depuración marcadas), el programa se inicia en un estado de pausa en la línea 1. El depurador siempre hará una pausa en la línea de código que está a punto de ejecutar. La ventana de Control de Depuración se parecerá a la Figura 10-2.



Figura 10-2. La ventana de control de depuración cuando el programa se inicia por primera vez bajo el depurador

Pulse el botón Over una vez para ejecutar la primera llamada a `print()`. Debería usar Over en lugar de Step aquí, ya que no quiere entrar en el código de la función `print()`. La ventana de Control de Depuración se actualizará a la línea 2, y la línea 2 en la ventana del editor de archivos se resaltará, como se muestra en la Figura 10-3. Esto le muestra dónde se encuentra actualmente la ejecución del programa. Esto le muestra donde se encuentra actualmente la ejecución del programa.



Figura 10-3. La ventana Debug Control después de hacer clic en Over

Pulse **Over** de nuevo para ejecutar la llamada a la función `input()`, y los botones de la ventana Debug Control se desactivarán mientras IDLE espera a que escriba algo para la llamada a la `función input()` en la ventana del intérprete de comandos interactivo. Introduzca **5** y pulse Return. Los botones de la ventana Debug Control se volverán a habilitar.

Siga haciendo clic en **Over**, introduciendo **3** y **42** como los dos números siguientes, hasta que el depurador se encuentre en la línea 7, la última llamada a `print()` del programa. La ventana de Control de Depuración debería parecerse a la Figura 10-4. Puede ver en la sección Globals que el campo

Las variables primera, segunda y tercera se establecen con los valores de cadena `'5'`, `'3'` y `'42'` en lugar de los valores enteros `5`, `3` y `42`. Cuando se ejecuta la última línea, estas cadenas se concatenan en lugar de sumarse, lo que provoca el error.

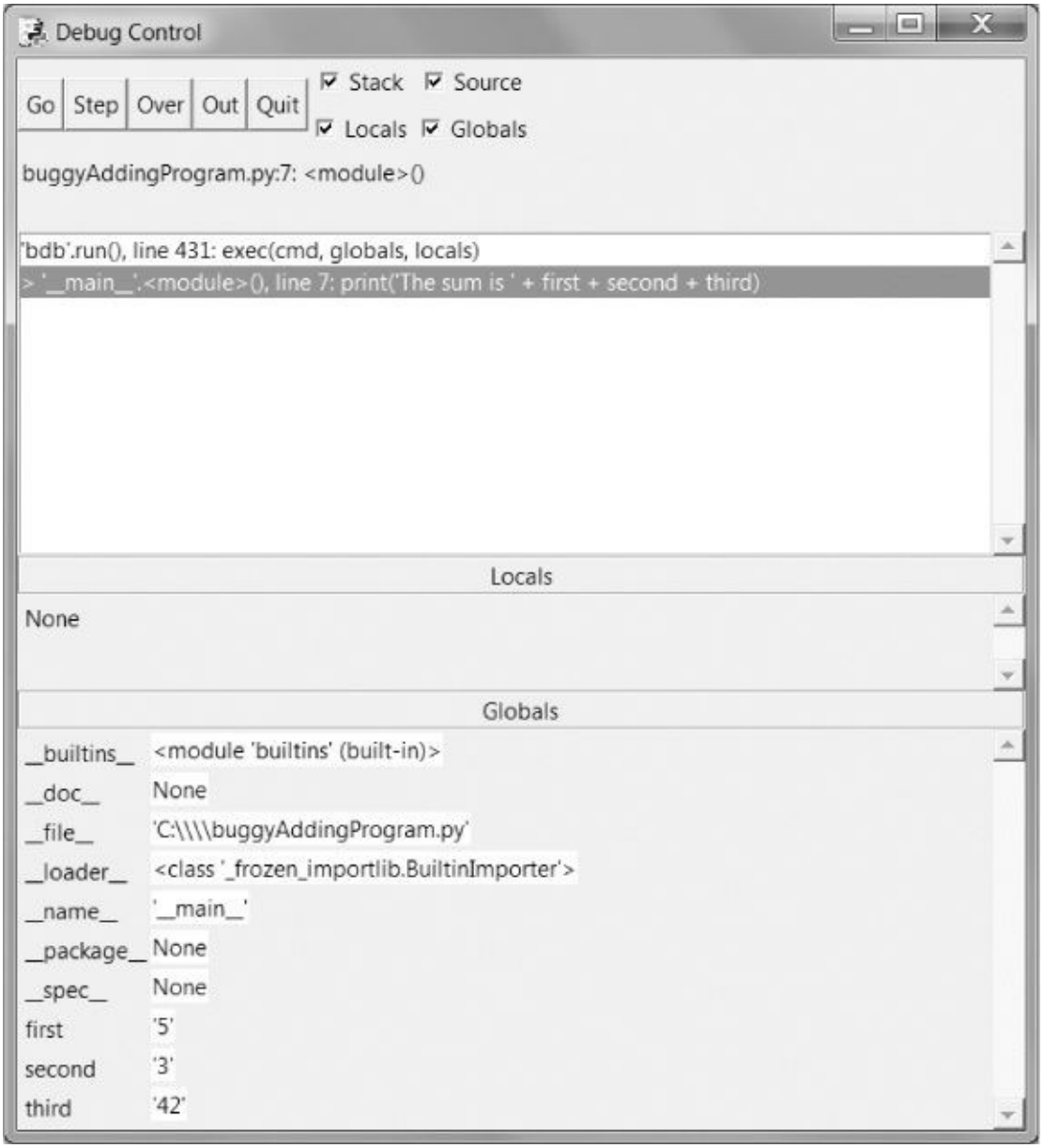


Figura 10-4. La ventana de control de depuración en la última línea. Las variables se establecen como cadenas, causando el error.

Recorrer el programa con el depurador es útil, pero también puede ser lento. A menudo querrás que el programa se ejecute normalmente hasta que llegue a una determinada línea de código. Puede configurar el depurador para hacer esto con puntos de interrupción.

Puntos de interrupción

Un *punto de interrupción* se puede establecer en una línea específica de código y obliga al depurador a hacer una pausa cada vez que la ejecución del programa llega a esa línea. Abre una nueva ventana del editor de archivos e introduce el siguiente programa, que simula lanzar una moneda 1.000 veces. Guárdalo como *coinFlip.py*.

```
import random
    heads= 0
    para i en rango(1, 1001):
❶       si random.randint(0, 1)== 1:
                cabezas= cabezas+ 1
         si  i== 500:
❷               print(' mitad de camino!')
    print('Salieron cabezas '+ str(cabezas)+ ' veces.")
```

La llamada a random.randint(0, 1) ❶ devolverá 0 la mitad de las veces y 1 la otra mitad. Esto puede usarse para simular un lanzamiento de moneda 50/50 donde 1 representa cara. Cuando ejecutas este programa sin el depurador, rápidamente sale algo como lo siguiente:

```
 mitad de camino!
Las cabezas salieron 490 veces.
```

Si ejecutara este programa bajo el depurador, tendría que pulsar el botón Over miles de veces antes de que el programa terminara. Si usted estuviera interesado en el valor de cabezas en el punto medio de la ejecución del programa, cuando 500 de 1000 lanzamientos de moneda se han completado, usted podría establecer un punto de en la línea print('Halfway done!')❷. Para establecer un punto **de** interrupción, haga clic con el botón derecho en la línea en el editor de archivos y seleccione **Establecer punto de interrupción**, como se muestra en la Figura 10-5.



Figura 10-5. Establecimiento de un punto de interrupción

No es conveniente establecer un punto de interrupción en la línea de la sentencia `if`, ya que ésta se ejecuta en cada iteración del bucle. Al establecer el punto de interrupción en el código de la sentencia `if`, el depurador interrumpe sólo cuando la ejecución entra en la cláusula `if`.

La línea con el punto de interrupción se resaltará en amarillo en el editor de archivos. Cuando ejecutes el programa bajo el depurador, comenzará en estado de pausa en la primera línea, como es habitual. Pero si haces clic en Ir, el programa se ejecutará a toda velocidad hasta que llegue a la línea con el punto de interrupción establecido en ella. A continuación, puede hacer clic en Go, Over, Step o Out para continuar normalmente.

Si desea eliminar un punto de interrupción, haga clic con el botón derecho en la línea en el editor de archivos y seleccione **Borrar punto de interrupción** en el menú. El resaltado amarillo desaparecerá, y el depurador no se romperá en esa línea en el futuro.

Resumen

Las aserciones, excepciones, registros y el depurador son herramientas valiosas para encontrar y prevenir errores en tu programa. Las aserciones con la sentencia `assert` de Python son una buena forma de implementar "comprobaciones de sanidad" que te avisan cuando una condición necesaria no se cumple. Las aserciones son sólo para errores que el programa no debería intentar recuperarse y debería fallar rápidamente. De lo contrario, debería lanzar una excepción.

Una excepción puede ser capturada y manejada por las sentencias `try` y `except`. El módulo de `registro` es una buena manera de mirar en su código mientras se está ejecutando y es mucho más cómodo de usar que la función `print()` debido a sus diferentes niveles de registro y la capacidad de registrar en un archivo de texto.

El depurador te permite recorrer tu programa línea a línea. Alternativamente, puede ejecutar su programa a velocidad normal y hacer que el depurador detenga la ejecución cada vez que llegue a una línea con un punto de interrupción establecido. Usando el depurador, puedes ver el estado del valor de cualquier variable en cualquier punto durante la vida del programa.

Estas herramientas y técnicas de depuración te ayudarán a escribir programas que funcionen. La introducción accidental de errores en el código es un hecho de la vida, no importa cuántos años de experiencia en codificación tengas.

Preguntas prácticas

P: 1. Escriba una sentencia assert que dispare un AssertionError si la variable spam es un entero menor que 10.

P: 2. Escribe una sentencia assert que desencadene un AssertionError si las variables eggs y bacon contienen cadenas iguales entre , aunque sus casos sean diferentes (es decir, 'hola' y 'hola' se consideran iguales, y 'adiós' y 'GOODbye' también se consideran iguales).

P: 3. Escribe una sentencia assert que *siempre* dispare un AssertionError.

P: 4. ¿Cuáles son las dos líneas que debe tener tu programa para poder llamar a logging.debug()?

P: 5. ¿Cuales son las dos líneas que debe tener tu programa para que logging.debug() envíe un mensaje de registro a un archivo llamado *programLog.txt*?

P: 6. ¿Cuáles son los cinco niveles de registro?

P: 7. ¿Qué línea de código puedes añadir para desactivar todos los mensajes de registro en tu programa?

P: 8. ¿Por qué es mejor usar mensajes de registro que usar print() para mostrar el mismo mensaje?

P: 9. ¿Cuáles son las diferencias entre los botones Step, Over y Out en la ventana Debug Control?

P: 10. Después de hacer clic en Go en la ventana Debug Control, ¿cuándo se detendrá el depurador?

P: 11. ¿Qué es un punto de interrupción?

P: 12. ¿Cómo se establece un punto de interrupción en una línea de código en IDLE?

Proyecto práctico

Para practicar, escribe un programa que haga lo siguiente.

Depurar el lanzamiento de monedas

El siguiente programa pretende ser un simple juego de adivinar lanzando una moneda. El jugador tiene dos oportunidades (es un juego fácil). Sin embargo, el programa tiene varios errores. Ejecute el programa varias veces para encontrar los errores que impiden que el programa funcione correctamente.

```
import random
guess= ''
while guess not in ('cara', 'cruz'):
    print('¡Adivina el lanzamiento de la moneda! Introduce
    cara o cruz:') guess= input()
toss= random.randint(0, 1) # 0 es cruz, 1 es cara si
toss== guess:
    print(' tienes!')
else:
    print('¡No! ¡Adivina otra vez!')
    guesss= input()
    if toss== guess:
        print('¡Lo has
        acertado!')
    si no:
         print('No. Eres realmente malo en este juego.')
```

Capítulo 11. Web Scraping

En esos raros y aterradores momentos en los que me quedo sin Wi-Fi, me doy cuenta de cuánto de lo que hago en el ordenador es en realidad lo que hago en Internet. Por pura costumbre, me encuentro intentando consultar el correo electrónico, leer los mensajes de Twitter de mis amigos o responder a la pregunta: "¿Tuvo Kurtwood Smith algún papel importante antes de participar *la película RoboCop* original de 1987?"[1][2]

Dado que gran parte del trabajo en un ordenador implica conectarse a Internet, sería estupendo que tus programas pudieran hacerlo. *Web scraping* es el término que se utiliza descargar y procesar contenidos de la Web. Por ejemplo, Google utiliza muchos programas de web scraping para indexar páginas web para su motor de búsqueda. En este capítulo, aprenderás sobre varios módulos que facilitan el scraping de páginas web en Python.

- **navegador web**. Viene con Python y abre un navegador a una página específica.
- **Peticiones**. Descarga archivos y páginas web de Internet.
- **Beautiful Soup**. Analiza HTML, el formato en que están escritas las páginas web.
- **Selenio**. Lanza y controla un navegador web. Selenium es capaz de rellenar formularios y simular clics de ratón en este navegador.

Proyecto: mapit.py con el módulo webbrowser

La función open() del módulo webbrowser puede lanzar un nuevo navegador a una URL especificada. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar navegador
>>> NAVEGADOR.open('http://inventwithpython.com/')
```

Se abrirá una pestaña del navegador web con la URL http://inventwithpython.com/. Esto es lo único que puede hacer el módulo webbrowser. Aún así, la función open() hace posible algunas cosas interesantes. Por ejemplo, es tedioso copiar una dirección al portapapeles y mostrar un mapa de la misma en Google Maps.
Podrías ahorrarte algunos pasos de esta tarea escribiendo un sencillo script que ejecute automáticamente el mapa en tu navegador utilizando el contenido de tu portapapeles.
De este modo, sólo tienes que copiar la dirección en el portapapeles y ejecutar el script, y el mapa se cargará por ti.

Esto es lo que hace su programa:

- Obtiene una dirección de calle a partir de los argumentos de la línea de comandos o del portapapeles.
- Abre el navegador web en la página de Google Maps correspondiente a la

dirección. Esto significa que tu código tendrá que hacer lo siguiente:

- Lee los argumentos de la línea de comandos desde sys.argv.
- Leer el contenido del portapapeles.
- Llama a la función webbrowser.open() para abrir el navegador web. Abre

una nueva ventana del editor de archivos y guárdala como *mapIt.py*.

Paso 1: Averiguar la URL

Basándote en las instrucciones del Apéndice B, configura *mapIt.py* para que cuando lo ejecutes desde la línea de comandos, así...

```
C:> mapit 870 Valencia St, San Francisco, CA 94110
```

... el script utilizará los argumentos de la línea de comandos en lugar del portapapeles. Si no hay argumentos en la línea de comandos, el programa sabrá utilizar el contenido del portapapeles.

Primero hay que averiguar qué URL utilizar para una dirección determinada. Cuando se carga *http://maps.google.com/* en el navegador y se busca una dirección, la URL que aparece en la barra de direcciones tiene este aspecto: *https://www.google.com/maps/place/870+Valencia+St/@37.7590311,-122.4215*

La dirección está en la URL, pero también hay mucho texto adicional. Los sitios web suelen añadir datos adicionales a las URL para ayudar a rastrear a los visitantes o personalizar los sitios. Pero si intentas entrar en *https://www.google.com/maps/place/870+Valencia+St+San+Francisco+CA/*, , verás que la página sigue siendo la correcta. Así que su programa puede ser configurado para abrir un navegador web para *https://www.google.com/maps/place/your_address_string"* (donde *su_cadena_de_dirección* es la dirección que desea asignar).

Paso 2: Manejar los argumentos de la línea de comandos

Haz que tu código tenga este aspecto:

```
#! python3
# mapIt.py - Lanza un mapa en el navegador usando una dirección de la #
línea de comandos o del portapapeles.

import webbrowser, sys if
len(sys.argv)> 1:
    # Obtén la dirección de la línea de
    comandos. address= '
    '.join(sys.argv[1:])

# TODO: Obtener dirección del portapapeles.
```

Después de la línea #! shebang del programa, necesitas importar el módulo webbrowser para lanzar el navegador e importar el módulo sys para leer los posibles argumentos de la línea de comandos. La variable sys.argv almacena una lista del nombre de archivo del programa y los argumentos de la línea de comandos. Si esta lista algo más que el nombre del archivo, entonces len(sys.argv) se evalúa como un número entero mayor que 1, lo que significa que se han proporcionado argumentos de línea de comandos.

Los argumentos de la línea de comandos suelen estar separados por espacios, pero en este casodesea interpretar todos los argumentos como una sola cadena. Dado que sys.argv es una lista de cadenas, puede pasarla al método join(), que devuelve un único valor de cadena. Usted no quiere que el nombre del programa en esta cadena, por lo que en lugar de sys.argv, debe pasar sys.argv[1:] para cortar el primer elemento de la matriz. La cadena final que evalúa esta expresión se almacena en la variable address.

Si ejecutas el programa introduciendo esto en la línea de comandos...

```
mapit 870 Valencia St, San Francisco, CA 94110
```

... la variable sys.argv contendrá este valor de la lista:

```
['mapIt.py', '870', 'Valencia', 'St, ', 'San', 'Francisco, ', 'CA', '94110']
```

La variable de dirección contendrá la cadena "870 Valencia St, San Francisco, CA 94110".

Paso 3: Manejar el contenido del portapapeles e iniciar el navegador

Haz que tu código sea como el siguiente:

```
#! python3
# mapIt.py - Lanza un mapa en el navegador usando una dirección de la #
línea de comandos o del portapapeles.

importar webbrowser, sys, pyperclip
si len(sys.argv)> 1:
    # Obtén la dirección de la línea de
    comandos. address= '
    '.join(sys.argv[1:])
si no:
    # Obtener DIRECCIÓN del portapapeles.
    DIRECCIÓN= pyperclip.paste()

WEBBROWSER.open('https://www.google.com/maps/place/'+ ADDRESS)
```

Si no hay argumentos en la línea de comandos, el programa asumirá que la dirección está almacenada en el portapapeles. Puede obtener el contenido del portapapeles con pyperclip.paste() y almacenarlo en una variable llamada address. Por último, para iniciar un navegador web con la URL de Google Maps, llame a webbrowser.open().

Aunque algunos de los programas que escriba realizarán tareas enormes que le ahorrarán horas, puede ser igual de satisfactorio utilizar un programa que le ahorre convenientemente unos segundos cada vez que realice una tarea común, como obtener el mapa de una dirección. La Tabla 11-1 compara los pasos necesarios para mostrar un mapa con y sin *mapIt.py*.

Tabla 11-1. Obtención de un mapa con y sin mapIt.py

Obtener manualmente un mapa	**Uso de *mapIt.py***
Resalte la dirección.	Resalte la dirección.
Copia la dirección.	Copia la dirección.
Abre el navegador web.	Ejecuta *mapIt.py*.
Vaya a http://maps.google.com/.	
Haz clic en el campo de texto de la dirección.	
Pega la dirección.	

Pulse ENTER.	

¿Ves cómo *mapIt.py* hace esta tarea menos tediosa?

Ideas para programas similares

Siempre que se disponga de una URL, el módulo webbrowser permite a los usuarios ahorrarse el paso de abrir el navegador y dirigirse a un sitio web. Otros programas podrían utilizar esta funcionalidad para hacer lo siguiente:

- Abrir todos los enlaces de una página en pestañas separadas del navegador.
- Abra el navegador para acceder a la URL de su meteorología local.
- Abra varios sitios de redes sociales que consulte con regularidad.

Descarga de archivos de la Web con el módulo de solicitudes

El módulo requests te permite descargar fácilmente archivos de la Web sin tener que preocuparte de problemas complicados como errores de red, problemas de conexión y compresión de datos. El módulo requests no viene con Python, así que tendrás que instalarlo primero. Desde la línea de comandos, ejecute **pip install REQUESTS**. (El Apéndice A tiene detalles adicionales sobre cómo instalar módulos de terceros).

El módulo requests fue escrito porque el módulo urllib2 de Python es demasiado complicado de usar. De hecho, coge un rotulador permanente y tacha todo este párrafo. Olvida que he mencionado urllib2. Si necesitas descargar cosas de la Web, simplemente usa el módulo requests.

A continuación, realice una sencilla prueba para asegurarse de que el módulo de peticiones se ha instalado correctamente. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> SOLICITUDES DE importación
```

Si no aparece ningún mensaje de error, el módulo de solicitudes se ha instalado correctamente.

Descarga de una página web con la función requests.get()

La función requests.get() toma una cadena de una URL para descargar. Llamando a type() sobre el valor de retorno de requests.get(), puedes ver que devuelve un Response, que contiene la respuesta que el servidor web dio para tu petición. Explicaré el objeto Response con más detalle más adelante, pero por ahora, introduce lo siguiente en el intérprete de comandos interactivo mientras tu ordenador está conectado a Internet:

```
>>> solicitudes de importación
>>> ❶res= requests.get('https://automatetheboringstuff.com/files/rj.txt')
>>> tipo(res)
<clase 'requests.models.Response'>
>>> ❷res.status_code== requests.codes.ok
Verdadero
>>> len(res.text)
178981
>>> print(res.text[:250])
El Proyecto Gutenberg EBook de Romeo y Julieta, de William Shakespeare

Este libro electrónico es para uso de cualquier persona en cualquier lugarsin
coste alguno y casi sin restricciones de ningún tipo. Puede , regalarlo o
reutilizarlo con arreglo a las condiciones del Proje
```

La URL va a una página web de texto para toda la obra de *Romeo y Julieta*, proporcionada en el sitio de este libro ❶. Puede saber que la solicitud de esta página web ha tenido éxito comprobando el atributo status_code del objeto Response. Si es igual al valor de requests.codes.ok, entonces todo ha ido bien ❷. (Por cierto, el código de estado para "OK" en el protocolo HTTP es 200.) Es posible que ya esté familiarizado con el código de estado 404 para "No encontrado").

Si la petición ha tenido éxito, la página web descargada se almacena como una cadena en la variable text del objeto Response. Esta variable contiene una gran cadena con toda la obra; la llamada a len(res.text) muestra que más de 178.000 caracteres. Finalmente, llamando a print(res.text[:250]) se muestran sólo los primeros 250 caracteres.

Comprobación de errores

Como ha visto, el objeto Response tiene un atributo status_code que puede ser comprobado con requests.codes.ok para ver si la descarga ha tenido éxito. Una forma más simple de comprobar el éxito es llamar al método raise_for_status() en el objeto Response. Esto lanzará una excepción si hubo un error descargando el archivo y no hará nada si la descarga tuvo éxito. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> RES= REQUESTS.get('http://inventwithpython.com/page_that_does_not_exist')
>>>   RES.RAISE_FOR_STATUS()
Traceback (llamada más reciente):
  File "<pyshell#138>", line 1, in <module>
    res.raise_for_status()
  File "C:\Python34\lib\site-packages\requests\models.py", line 773, in raise_for_status
    raise HTTPError(http_error_msg, response=self)
requests.exceptions.HTTPError: 404 Error de cliente: No encontrado
```

El método raise_for_status() es una buena manera de asegurar que un programa se detenga si ocurre una mala descarga. Esto es algo bueno: Usted quiere que su programa se detenga tan pronto como ocurra algún error inesperado. Si una descarga fallida *no* es un problema para su programa, puede envolver la línea raise_for_status() con sentencias try y except para manejar este caso de error sin bloquearse.

```
solicitudes de importación
res=
requests.get('http://inventwithpython.com/page_that_does_not_exist') try:
    res.raise_for_status() except
Excepción as exc:
    print('Hubo un problema: %s' % (exc))
```

Esta llamada al método raise_for_status() hace que el programa muestre lo siguiente:

```
Ha habido un problema: 404 Error de cliente: No encontrado
```

Llame siempre a raise_for_status() después de llamar a requests.get(). Quiere estar seguro de que la descarga ha funcionado realmente antes de que su programa continúe.

Guardar archivos descargados en el disco duro

A partir de aquí, puedes guardar la página web en un archivo de tu disco duro con la función estándar open() y el método `write()`. Sin embargo, hay algunas pequeñas diferencias. En primer lugardebe abrir el archivo en modo *binario de escritura* pasando la cadena `'wb'` como segundo argumento de open(). Incluso si la página está en texto plano (como el texto de *Romeo y Julieta* que descargaste antes), necesitas escribir datos binarios en lugar de datos de texto para mantener la *codificación Unicode* del texto.

CODIFICACIONES UNICODE

Las codificaciones Unicode están fuera del alcance de este libro, pero puedes aprender más sobre ellas en estas páginas web:

- Joel on Software: The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!): *http://www.joelonsoftware.com/articles/Unicode.html*

  Unicode pragmático: *http://nedbatchelder.com/text/unipain.html*

-

Para escribir la página web en un archivo, puede utilizar un bucle for con la respuesta objeto iter_content().

```
>>> SOLICITUDES DE importación
>>>    RES    =    REQUESTS.get('https://automatetheboringstuff.com/files/rj.txt')
>>>  RES.RAISE_FOR_STATUS()
>>> playFile= open('RomeoAndJuliet.txt', 'wb')
>>> for chunk in RES.iter_content(100000):
        playFile.write(chunk)

100000
78981
>>> playFile.close()
```

El método iter_content() devuelve "trozos" del contenido en cada iteración a través del bucle. Cada trozo es del tipo *bytes*, y puede especificar cuántos bytes contendrá cada trozo. Cien mil bytes es generalmente un buen tamaño, así que pase 100000 como argumento a iter_content().

El archivo *RomeoAndJuliet.txt* existirá ahora en el directorio de trabajo actual. Tenga en cuenta que mientras que el nombre de archivo en el sitio web era *rj.txt*, el archivo en su disco duro tiene un nombre de archivo diferente. El módulo requests simplemente se encarga de descargar el contenido de las páginas web. Una vez que la página se ha descargado, son simplemente datos en su directorio de trabajo actual.

programa. Aunque perdieras la conexión a Internet después de descargar la página web, todos los datos de la página seguirían estando en tu ordenador.

El método `write()` devuelve el número de bytes escritos en el fichero. En el ejemplo anterior, había 100.000 bytes en el primer trozo, y la parte restante del archivo sólo necesitaba 78.981 bytes.

Para repasar, éste es el proceso completo para descargar y guardar un archivo:

1. Llama a `requests.get()` para descargar el archivo.
2. Llame a `open()` con `'wb'` para crear un nuevo archivo en modo binario de escritura.
3. Bucle sobre el método `iter_content()` del objeto `Response`.
4. Llama a `write()` en cada iteración para escribir el contenido en el fichero.
5. Llama a `close()` para cerrar el fichero.

Eso es todo lo que hay en el módulo `requests`. El bucle `for` y el `iter_content()` pueden parecer complicados comparados con el flujo de trabajo `open()`/`write()`/`close()` que has estado usando para escribir archivos de texto, pero es para asegurar que el módulo `requests` no consuma demasiada memoria incluso si descargas archivos masivos. Puedes conocer otras características del módulo requests en http://requests.readthedocs.org/.

HTML

Antes de desmontar páginas web, aprenderás algunos conceptos básicos de HTML. También aprenderá a acceder a las potentes herramientas de desarrollo de su navegador, que le facilitarán enormemente la extracción de información de la Web.

Recursos para aprender HTML

El lenguaje de marcado de hipertexto (HTML) es el formato en el que están escritas las páginas web. Este capítulo asume que tienes alguna experiencia básica con HTML, pero si necesitas un tutorial para principiantes, te sugiero uno de los siguientes sitios:

- *http://htmldog.com/guides/html/beginner/*
- *http://www.codecademy.com/tracks/web/*
- *https://developer.mozilla.org/en-US/learn/html/*

Un repaso rápido

Por si hace tiempo que no ves nada de HTML, aquí tienes un repaso rápido a los conceptos básicos. Un archivo HTML es un archivo de texto plano con la extensión *.html*. El texto de estos archivos está rodeado de *etiquetas*, que son palabras encerradas entre corchetes angulares. Las etiquetas indican al navegador cómo debe formatear la página web. Una etiqueta de inicio y una de cierre pueden encerrar texto para formar un *elemento*. *El texto* (o *HTML interno*) es el contenido entre las etiquetas de inicio y cierre. Por ejemplo, el siguiente HTML mostrará *¡Hola mundo!* en el navegador, con *Hola* en negrita:

```
<strong>Hola</strong>  ¡mundo!
```

Este HTML se verá como la Figura 11-1 en un navegador.



Figura 11-1. Hola mundo! renderizado en el navegador

La etiqueta de apertura indica que el texto adjunto aparecerá en negrita. La etiqueta de cierre indica al navegador dónde termina el texto en negrita.

Hay muchas etiquetas diferentes en HTML. Algunas de estas etiquetas tienen propiedades adicionales en forma de *atributos* dentro de los corchetes angulares. Por ejemplo, la etiqueta

La etiqueta <a> encierra texto que debería ser un enlace. La URL a la que enlaza el texto viene determinada por el atributo href. He aquí un ejemplo:

```
Los libros gratuitos de <a href="http://inventwithpython.com">Python</a> de Al.
```

Este HTML tendrá el aspecto de la Figura 11-2 en un navegador.



Figura 11-2. Enlace mostrado en el navegador

Algunos elementos tienen un atributo id que se utiliza para identificar unívocamente el elemento en la página. A menudo darás instrucciones a tus programas para que busquen un elemento por su atributo id, por lo que averiguar el atributo id de un elemento utilizando las herramientas de desarrollo del navegador es una tarea común al escribir programas de web scraping.

Visualización del código HTML de una página web

Necesitarás ver la fuente HTML de las páginas web con las que trabajarán tus programas. Para ello, haz clic con el botón derecho del ratón (o CTRL-clic en OS X) en cualquier página web de tu navegador y selecciona **Ver fuente** o **Ver fuente de la página** para ver el texto HTML de la página (ver Figura 11-3). Este es el texto que realmente recibe su navegador. El navegador sabe cómo mostrar, o *renderizar*, la página web a partir de este HTML.

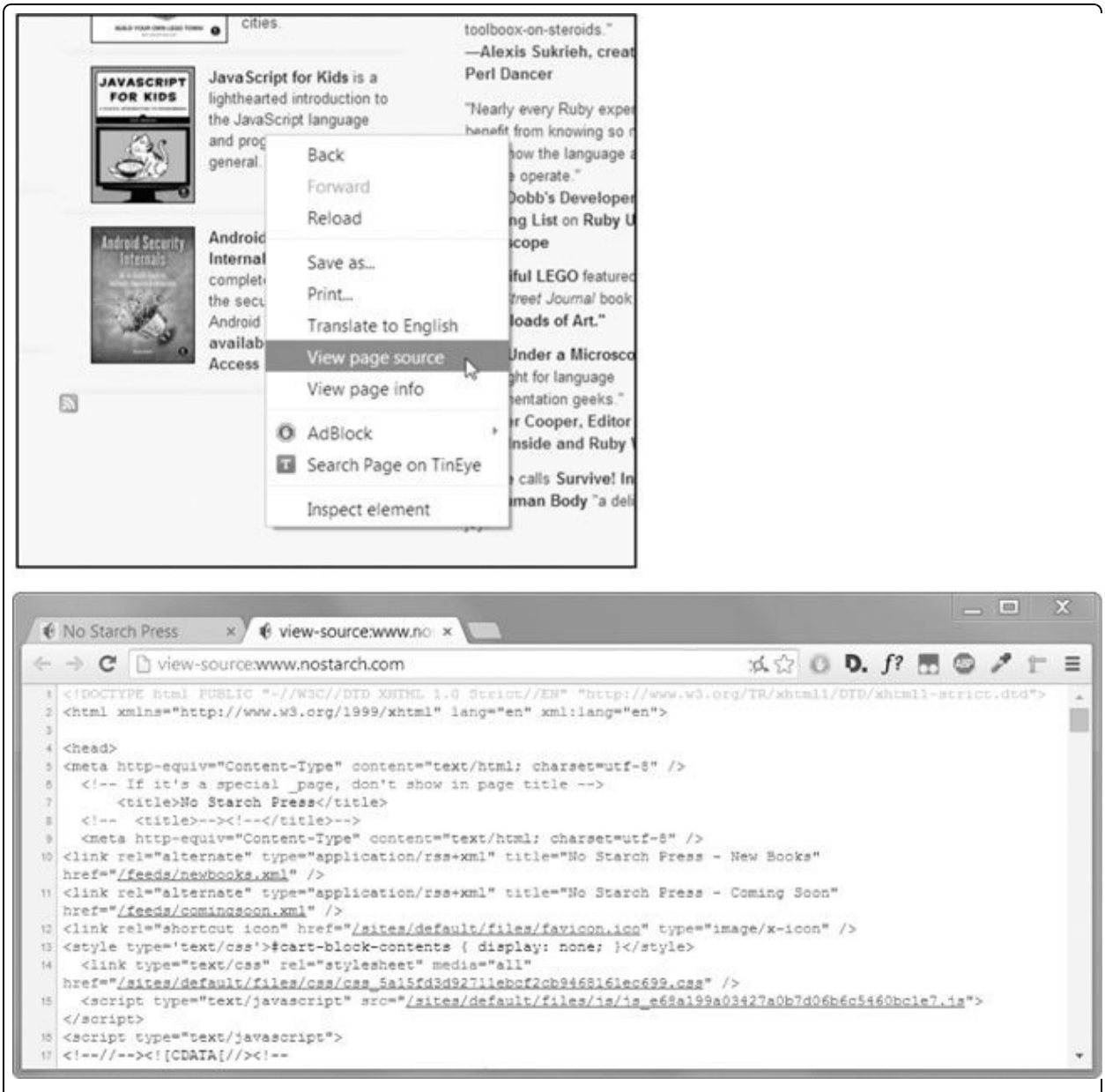


Figura 11-3. Visualización de la fuente de una página web

Le recomiendo encarecidamente que vea el código HTML de algunos de sus sitios favoritos. No pasa nada si no entiendes del todo lo que ves cuando miras el código fuente. No necesitarás dominar el HTML para escribir programas sencillos de web scraping. -Después de todo, no escribirás tus propias páginas web. Sólo necesitas conocimientos suficientes para extraer datos de un sitio existente.

Abrir las herramientas de desarrollo del navegador

Además de ver el código fuente de una página web, puedes examinar el HTML de una página utilizando las herramientas de desarrollo de tu navegador. En Chrome e Internet Explorer para Windows, las herramientas de desarrollo ya están instaladas, y puedes pulsar F12 para que aparezcan (ver Figura 11-4). Si vuelves a pulsar F12, las herramientas de desarrollo desaparecerán. En Chrometambién puedes abrir las herramientas de desarrollo seleccionando Ver▸ Desarrollador▸ Herramientas de desarrollo. En OS X, pulsando⌘-OPTION-I se abrirán las herramientas de desarrollo de Chrome.



Figura 11-4. La ventana Herramientas de desarrollo en el navegador Chrome

En Firefox, puedes abrir el Inspector de Herramientas de Desarrollo Web pulsando CTRL-SHIFT-C en Windows y Linux o pulsando⌘ -OPTION-C en OS X. El diseño es casi idéntico al de las herramientas para desarrolladores de Chrome.

En Safari, abre la ventana de Preferencias y, en el panel Avanzado, marca la opción **Mostrar menú Desarrollo en la barra de menús**. Una vez activada, puedes abrir las herramientas de desarrollo pulsando⌘-OPTION-I.

Después de activar o instalar las herramientas de desarrollo en su navegador, puede hacer clic con el botón derecho del ratón en cualquier parte de la página web y seleccionar **Inspeccionar elemento** en el menú contextual.

para que aparezca el HTML responsable de esa parte de la página. Esto le será útil cuando empiece a analizar HTML para sus programas de web scraping.

NO UTILICE EXPRESIONES REGULARES PARA ANALIZAR HTML

Localizar un fragmento específico de HTML en una cadena parece un caso perfecto para las expresiones regulares. Sin embargo, te aconsejo que no lo hagas. Hay muchas maneras diferentes en que HTML puede ser formateado y aún ser considerado HTML válido, pero tratar de capturar todas estas posibles variaciones en una expresión regular puede ser tedioso y propenso a errores. Un módulo desarrollado específicamente para analizar HTML, como Beautiful Soup, tendrá menos probabilidades de dar lugar a errores.

Puede encontrar un argumento más extenso de por qué no debería analizar HTML con expresiones regulares en http://stackoverflow.com/a/1732454/1893164/.

Uso de las herramientas de desarrollo para buscar elementos HTML

Una vez que su programa haya descargado una página web utilizando el módulo requests, tendrá el contenido HTML de la página como un único valor de cadena. Ahora tienes que averiguar qué parte del HTML corresponde a la información de la página web que te interesa.

Aquí es donde las herramientas de desarrollo del navegador pueden ayudar. Supongamos que quieres escribir un programa que extraiga datos meteorológicos de http://weather.gov/. Antes de escribir el código, investiga un poco. Si visitas el sitio y buscas el código postal 94105, el sitio te llevará a una página que muestra la previsión para esa zona.

¿Y si te interesa obtener la información sobre la temperatura de ese código postal? Haz clic con el botón derecho del ratón en el lugar de la página (o CONTROL-clic en OS X) y selecciona **Inspeccionar elemento** en el menú contextual que aparece. Aparecerá la ventana Herramientas de desarrollo, que muestra el HTML que produce esta parte concreta de la página web. La Figura 11-5 las herramientas de desarrollo abiertas al HTML de la temperatura.

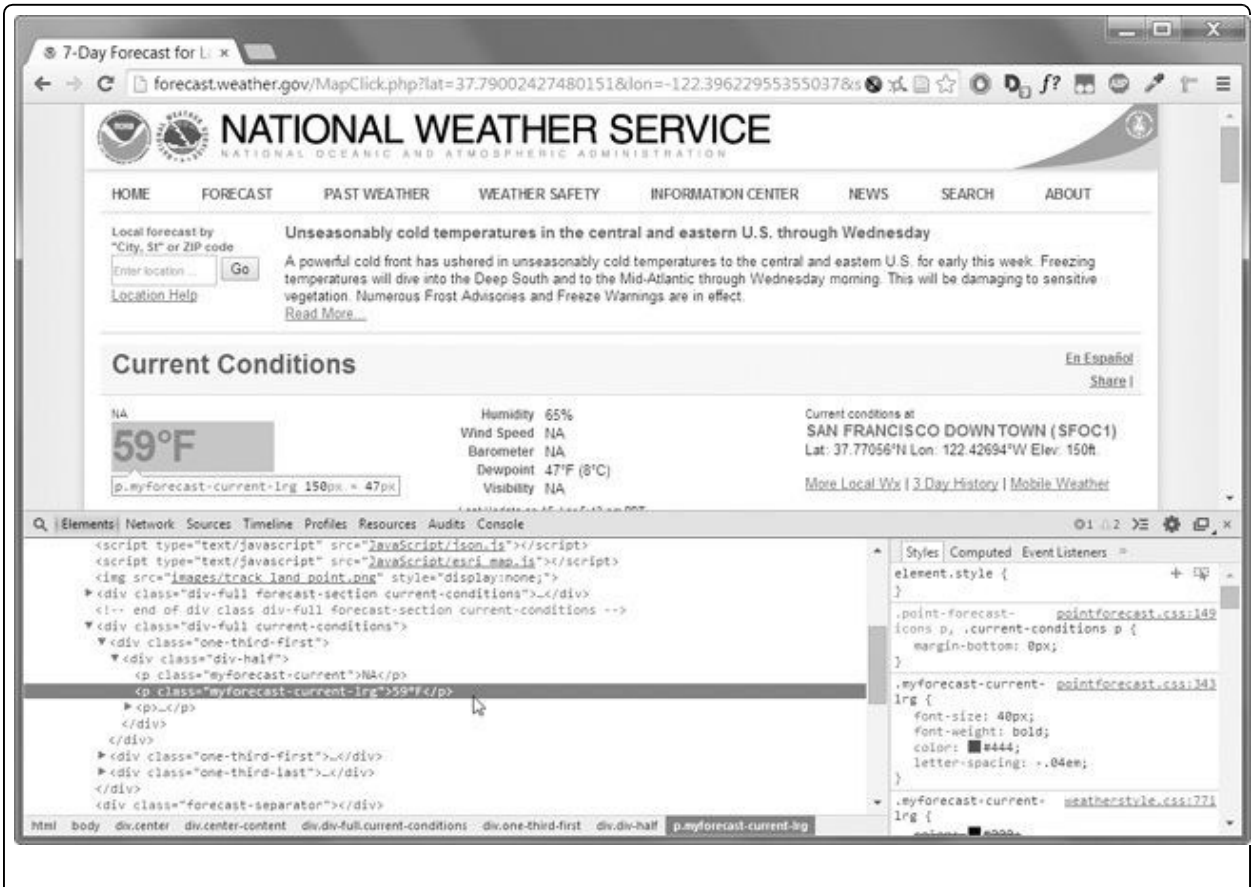


Figura 11-5. Inspección del elemento que contiene el texto de la temperatura con las herramientas de desarrollo

Desde las herramientas para desarrolladores, puede ver que el HTML responsable de la parte de temperatura de la página web es `<p class="myforecast-current-lrg">59°F</p>`. Esto es exactamente lo que estabas buscando. Parece que la información sobre la temperatura está contenida dentro de un elemento `<p>` con la clase `myforecast-current-lrg`. Ahora que ya sabes lo que , el módulo `BeautifulSoup` te ayudará a encontrarlo en la cadena.

Análisis de HTML con el módulo BeautifulSoup

Beautiful Soup es un módulo para extraer información de una página HTML (y es mucho mejor para este propósito que las expresiones regulares). El nombre del módulo BeautifulSoup es bs4 (de Beautiful Soup, versión 4). Para instalarlo, necesitarás ejecutar pip install beautifulsoup4 desde la línea de comandos. (Echa un vistazo al Apéndice A para obtener instrucciones sobre la instalación de módulos de terceros). beautifulsoup4 es el nombre utilizado para la instalación, para importar Beautiful Soup se ejecuta import bs4.

Para este capítulo, los ejemplos de Beautiful Soup *analizarán* (es decir, analizarán e identificarán las partes de) un archivo HTML en el disco duro. Abre una nueva ventana del editor de archivos en IDLE, introduce lo siguiente y guárdalo como *ejemplo.html*. Alternativamente, descárgalo de http://nostarch.com/automatestuff/.

```
<!-- Este es el archivo de ejemplo example.html. -->

<html><head><title>El título del sitio web</title></head>
<body>
<p>Descarga mi libro <strong>Python</strong> desde <a href="http:// inventwithpython.com">mi
página web</a>.</p> <p><a href="http:// inventwithpython.com">mi página web</a>.
<p class="slogan">¡Aprende Python de la  más fácil!</p> <p
<p>Por <span id="author">Al Sweigart</span></p>
</body></html>
```

Como puedes ver, incluso un simple archivo HTML implica muchas etiquetas y atributos diferentes, y las cosas se vuelven rápidamente confusas con sitios web complejos. Afortunadamente, Beautiful Soup hace que trabajar con HTML sea mucho más fácil.

Creación de un objeto BeautifulSoup a partir de HTML

La función bs4.BeautifulSoup() necesita ser invocada con una cadena que contenga el HTML que va a analizar. La función bs4.BeautifulSoup() devuelve un objeto BeautifulSoup. Introduzca lo siguiente en el intérprete de comandos interactivo mientras su ordenador está conectado a Internet:

```
>>> SOLICITUDES DE importación, bs4
>>> RES= REQUESTS.get('http://nostarch.com')
>>>   RES.RAISE_FOR_STATUS()
>>> noStarchSoup= BS4.BeautifulSoup(RES.text)
>>> type(noStarchSoup)
<clase 'bs4.BeautifulSoup'>
```

Este código utiliza requests.get() para descargar la página principal del sitio web No Starch Press y luego pasa el atributo text de la respuesta a bs4.BeautifulSoup(). El objeto BeautifulSoup que devuelve se almacena en una variable llamada noStarchSoup.

También puede cargar un archivo HTML desde su disco duro pasando un objeto File a bs4.BeautifulSoup(). Introduzca lo siguiente en el intérprete de comandos interactivo (asegúrese de que el archivo *example.html* se encuentra en el directorio de trabajo):

```
>>> ejemploArchivo= open('ejemplo.html')
>>>   ejemploSopa   =   BS4.BeautifulSoup(ejemploArchivo)
>>> tipo(ejemploSopa)
<clase 'bs4.BeautifulSoup'>
```

Una vez que tenga un objeto BeautifulSoup, puede utilizar sus métodos para localizar partes específicas de un documento HTML.

Encontrar un elemento con el método select()

recuperar un elemento de página web desde un objeto BeautifulSoup llamando al método select()y pasando una cadena de un *selector* CSS para el elemento que está buscando. Los selectores son como expresiones regulares: Especifican un patrón a buscar, en este , en páginas HTML en lugar de cadenas de texto generales.

Una discusión completa de la sintaxis de los selectores CSS va más allá del alcance de este libro (hay un buen tutorial sobre selectores en los recursos de http://nostarch.com/automatestuff/), pero aquí tienes una breve introducción a los selectores. La Tabla 11-2 muestra ejemplos de los patrones de selectores CSS más comunes.

Tabla 11-2. Ejemplos de selectores CSS

Selector pasado a select() **método**	**Coincidirá...**
soup.select('div')	Todos los elementos denominados
soup.select('#autor')	El elemento con un atributo id de autor
soup.select('.notice')	Todos los elementos que utilizan un atributo de clase CSS denominado notice
soup.select('div span')	Todos los elementos denominados que están dentro de un elemento denominado .
soup.select('div> span')	Todos los elementos denominados que están *directamente* dentro de un elemento denominado , sin ningún otro elemento entre ellos.
soup.select('entrada[nombre]')	Todos los elementos denominados <input> que tengan un atributo name con cualquier valor.
soup.select('input[type="button"]')	Todos los elementos denominados <input> que tengan un atributo denominado tipo con botón de valor

Los distintos patrones de selección pueden combinarse para obtener coincidencias sofisticadas. Por ejemplo, soup.select('p #author') coincidirá con cualquier elemento que tenga el atributo id author, siempre que también esté dentro de un elemento <p>.

El método select(devolverá una lista de objetos Tag, que es como Beautiful Soup representa un elemento HTML. La lista contendrá un objeto Tag por cada coincidencia en el HTML del objeto BeautifulSoup. Los valores de las etiquetas pueden pasarse a

la función str() para mostrar las etiquetas HTML que representan. Los valores de las etiquetas también tienen un atributo attrs que muestra todos los atributos HTML de la etiqueta en forma de diccionario. Utilizando el archivo *example.html* de antes, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar bs4
>>> ejemploArchivo= open('ejemplo.html')
>>> exampleSoup = BS4.BeautifulSoup(exampleFile.read())
>>> ELEMS= exampleSoup.select('#author')
>>> tipo(ELEMS)
<clase 'lista'>
>>> len(ELEMS)
1
>>> type(ELEMS[0])
<clase 'bs4.element.Tag'>
>>> ELEMS[0].getText()
Al Sweigart
>>> str(ELEMS[0])
'<span id="author">Al Sweigart</span>'
>>> ELEMS[0].ATTRS
{'id': 'autor'}
```

Este código extraerá el elemento con id="author" de nuestro HTML de ejemplo. Usamos select('#author') para obtener una lista de todos los elementos con id="author". Almacenamos esta lista de objetos **Tag** en la variable elems, y len(elems) nos dice que hay un objeto **Tag** en la lista; hubo una coincidencia. Llamar a getText() sobre el elemento devuelve el texto del elemento, o HTML interno. El texto de un elemento es el contenido entre las etiquetas de apertura y cierre: en este caso, "Al Sweigart".

Al pasar el elemento a str() se obtiene una cadena con las etiquetas de inicio y cierre y el texto del elemento. Por último, attrs nos da un diccionario con el atributo del elemento, 'id', y el valor del atributo id, 'author'.

También puede extraer todos los elementos <p> del objeto BeautifulSoup. Introduce esto en el intérprete de comandos interactivo:

```
>>> PELEMS= ejemploSopa.select('p')
>>> str(PELEMAS[0])
<p>Descarga mi libro <strong>Python</strong> desde <a href="http:// inventwithpython.com">mi
página web</a>.</p> <p>'
>>> PELEMS[0].getText()
'Descarga mi libro de Python de mi página web'.
>>> str(PELEMAS[1])
```

```
<p class="slogan">¡Aprende Python de forma fácil!</p>'
>>> pElems[1].getText()
Aprenda Python de  sencilla".
>>> str(pElemas[2])
<p>Por <span id="author">Al Sweigart</span></p>'
>>> pElems[2].getText()
Por Al Sweigart
```

Esta vez, select() nos da una lista de tres coincidencias, que almacenamos en pElems. Usando str() en pElems[0], pElems[1], y pElems[2] nos muestra cada elemento como una cadena, y usando getText() en cada elemento nos muestra su texto.

Obtener datos de los atributos de un elemento

El método get() de los objetos Tag facilita el acceso a los valores de los atributos de un elemento. Al método se le pasa una cadena con el nombre de un atributo y devuelve el valor de ese atributo. Usando *ejemplo.html*, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar bs4
>>> sopa= BS4.BeautifulSoup(open('ejemplo.html'))
>>> spanElem = soup.select('span')[0]
>>> str(spanElem)
'<span id="author">Al Sweigart</span>'
>>> spanElem.get('id')
autor
>>> spanElem.get('SOME_NONEXISTENT_ADDR')== None
Verdadero
>>> spanElem.ATTRS
{'id': 'autor'}
```

Aquí usamos select() para encontrar cualquier elemento y luego almacenamos el primer elemento coincidente en spanElem. Al pasar el nombre del atributo 'id' a get() se devuelve el valor del atributo, 'author'.

Proyecto: "Me siento afortunado" Búsqueda en Google

Siempre que busco un tema en Google, no miro sólo un resultado de la búsqueda a la vez. Si hago clic con el botón central en el enlace de un resultado de búsqueda (o hago clic manteniendo pulsada la tecla CTRL), abro los primeros enlaces en varias pestañas nuevas para leerlos más tarde. Hago búsquedas en Google con tanta frecuencia que este flujo de trabajo -abrir el navegador, buscar un tema y hacer clic con el botón central en varios enlaces uno a uno- resulta tedioso. Estaría bien poder escribir un término de búsqueda en la línea de comandos y que mi ordenador abriera automáticamente un navegador con todos los resultados de búsqueda principales en pestañas nuevas. Escribamos un script para hacerlo.

Esto es lo que hace su programa:

- Obtiene las palabras clave de búsqueda a partir de los argumentos de la línea de comandos.
- Recupera la página de resultados de la búsqueda.
- Abre una pestaña del navegador para cada resultado.

Esto significa que su código tendrá que hacer lo siguiente:

- Lee los argumentos de la línea de comandos desde sys.argv.
- Obtener la página de resultados de búsqueda con el módulo de peticiones.
- Encuentre los enlaces de cada resultado de búsqueda.
- Llama a la función webbrowser.open() para abrir el navegador web. Abre

una nueva ventana del editor de archivos y guárdala como *lucky.py*.

Paso 1: Obtener los argumentos de la línea de comandos y solicitar la página de búsqueda

Antes de codificar nada, primero necesitas conocer la URL de la página de resultados de la búsqueda. Si miras la barra de direcciones del navegador después de hacer una búsqueda en Google, puedes ver que la página de resultados tiene una URL como *https://www.google.com/search? q=SEARCH_TERM_HERE*. El módulo de peticiones puede descargar esta página y, a continuación, puedes utilizar Beautiful Soup para encontrar los enlaces de los resultados de la búsqueda en el HTML. Finalmente, usarás el módulo webbrowser para abrir esos enlaces en pestañas del navegador.

Haz que tu código sea como el siguiente:

```
#! python3
# lucky.py - Abre varios resultados de búsqueda en

Google. import requests, sys, webbrowser, bs4

print("Buscando en Google...") # mostrar texto mientras se descarga la página de Google
res= requests.get('http://google.com/search?q='+ ' '.join(sys.argv[1:])) res.raise_for_status()

# TODO: Recuperar los enlaces de los principales resultados de búsqueda.

# TODO: Abrir una pestaña del navegador para cada resultado.
```

El usuario especificará los términos de búsqueda mediante argumentos en la línea de comandos cuando inicie el programa. Estos argumentos se almacenarán como cadenas en una lista en sys.argv.

Paso 2: Buscar todos los resultados

Ahora necesita utilizar Beautiful Soup para extraer los enlaces de los resultados de búsqueda más importantes del HTML descargado. Pero, ¿cómo elegir el selector adecuado? Por ejemplo, no puedes buscar simplemente todas las etiquetas <a>, porque hay muchos enlaces que no te interesan en el HTML. En su lugar, debe inspeccionar la página de resultados de la búsqueda con las herramientas de desarrollo del navegador para intentar encontrar un selector que
elija sólo los enlaces que desee.

Después de hacer una búsqueda en Google de *Beautiful Soup*, puedes abrir las herramientas de desarrollo del navegador e inspeccionar algunos de los elementos de enlace de la página. Parecen increíblemente complicados, algo así: Hermosa Sopa : Le llamábamos Tortuga porque nos enseñó..

No importa que el elemento parezca increíblemente complicado. Sólo tienes que encontrar el patrón que tienen todos los enlaces de resultados de búsqueda. Pero este elemento <a> no tiene nada que lo distinga fácilmente de los elementos <a> de la página que no son resultados de búsqueda.

Haz que tu código sea como el siguiente:

```
#! python3
# lucky.py - Abre varios resultados de búsqueda en

google. import requests, sys, webbrowser, bs4

--snip--

# Recuperar los mejores ENLACES DE
los resultados de búsqueda. soup=
BS4.BeautifulSoup(RES.text)

# Abrir una pestaña del navegador para cada
resultado. LINKELEMS= soup.select('.r a')
```

Sin embargo, si miras un poco más arriba del elemento <a>, hay un elemento como este:

<h3 class="r">. Mirando el resto de la fuente HTML, parece que la clase r se utiliza sólo para los enlaces de resultados de búsqueda. No tienes por qué saber qué es la clase r de CSS ni qué hace. Simplemente vas a utilizarla como marcador del elemento <a> que estás buscando. Puedes crear un objeto BeautifulSoup a partir del texto HTML de la página descargada y luego usar el selector '.r a' para encontrar todos los elementos <a> que estén dentro de un elemento que tenga la clase CSS r.

Paso 3: Abrir los navegadores para cada resultado

Por último, le diremos al programa que abra pestañas del navegador web para nuestros resultados. Añade lo siguiente al final de tu programa:

```
#! python3
# lucky.py - Abre varios resultados de búsqueda en

google. import requests, sys, webbrowser, bs4

--snip--

# Abrir una pestaña del navegador para cada
resultado. linkElems= soup.select('.r
a') numOpen= min(5, len(LINKELEMS))
for i in range(numOpen):
    WEBBROWSER.open('http://google.com'    +    LINKELEMS[i].get('href'))
```

Por defecto, se abren los cinco primeros resultados de la búsqueda en nuevas pestañas mediante el módulo webbrowser. Sin embargo, es posible que el usuario haya buscado algo y haya obtenido menos de cinco resultados. La llamada soup.select() devuelve una lista de todos los elementos que coinciden con su selector '.r a', por lo que el número de pestañas que desea abrir es 5 o la longitud de esta lista (la que sea menor).

La función integrada min() de Python devuelve el menor de los argumentos enteros o flotantes que se le pasen. (También existe una función incorporada max() que devuelve el mayor argumento que se le pase). Puedes utilizar min() para averiguar si hay menos de cinco enlaces en la lista y almacenar el número de enlaces a abrir en una variable llamada numOpen. A continuación, puede ejecutar un bucle for llamando a range(numAbrir).

En cada iteración del bucle, se utiliza webbrowser.open() para abrir una nueva pestaña en el navegador web. Ten en cuenta que el valor del atributo href en los elementos <a> devueltos no tiene la parte inicial http://google.com, por lo que tienes que concatenarla con el valor de cadena del atributo href.

Ahora puedes abrir instantáneamente los cinco primeros resultados de Google para, por ejemplo, *tutoriales de programación en Python*, ¡ejecutando lucky python programming tutorials en la línea de comandos! (Consulta el Apéndice B para saber cómo ejecutar programas fácilmente en tu sistema operativo).

Ideas para programas similares

La ventaja de la navegación por pestañas es que puedes abrir fácilmente enlaces en pestañas nuevas para examinarlos más tarde. Un programa que abra automáticamente varios enlaces a la vez puede ser un buen atajo para hacer lo siguiente:

- Abrir todas las páginas de productos después de buscar en un sitio de compras como Amazon
- Abrir todos los enlaces a las reseñas de un mismo producto
- Abrir los enlaces de resultados a fotos tras realizar una búsqueda en un sitio de fotos como Flickr o Imgur

Proyecto: Descargar todos los cómics de XKCD

Los blogs y otros sitios web que se actualizan con regularidad suelen tener una portada con la entrada más reciente, así como un botón Anterior en la página que te lleva a la entrada anterior. A continuación, esa entrada también tendrá un botón Anterior, y así sucesivamente, creando un rastro desde la página más reciente hasta la primera entrada del sitio. Si quieres una copia del contenido del sitio para leerlo cuando no estés conectado, puedes navegar manualmente por cada página y guardar cada una de ellas. Pero esto es un trabajo bastante aburrido, así que vamos a escribir un programa para hacerlo en su lugar.

XKCD es un popular webcómic geek con un sitio web que se ajusta a esta estructura (véase la Figura 11-6). La página principal en *http://xkcd.com/* tiene un botón Prev que guía al usuario a través de los cómics anteriores. Descargar cada cómic a mano llevaría una eternidad, pero puedes escribir un script para hacerlo en un par de minutos.

Esto es lo que hace tu programa:

- Carga la página de inicio de XKCD.
- Guarda la imagen del cómic en esa página.
- Sigue el enlace Previous Comic.
- Se repite hasta llegar al primer cómic.

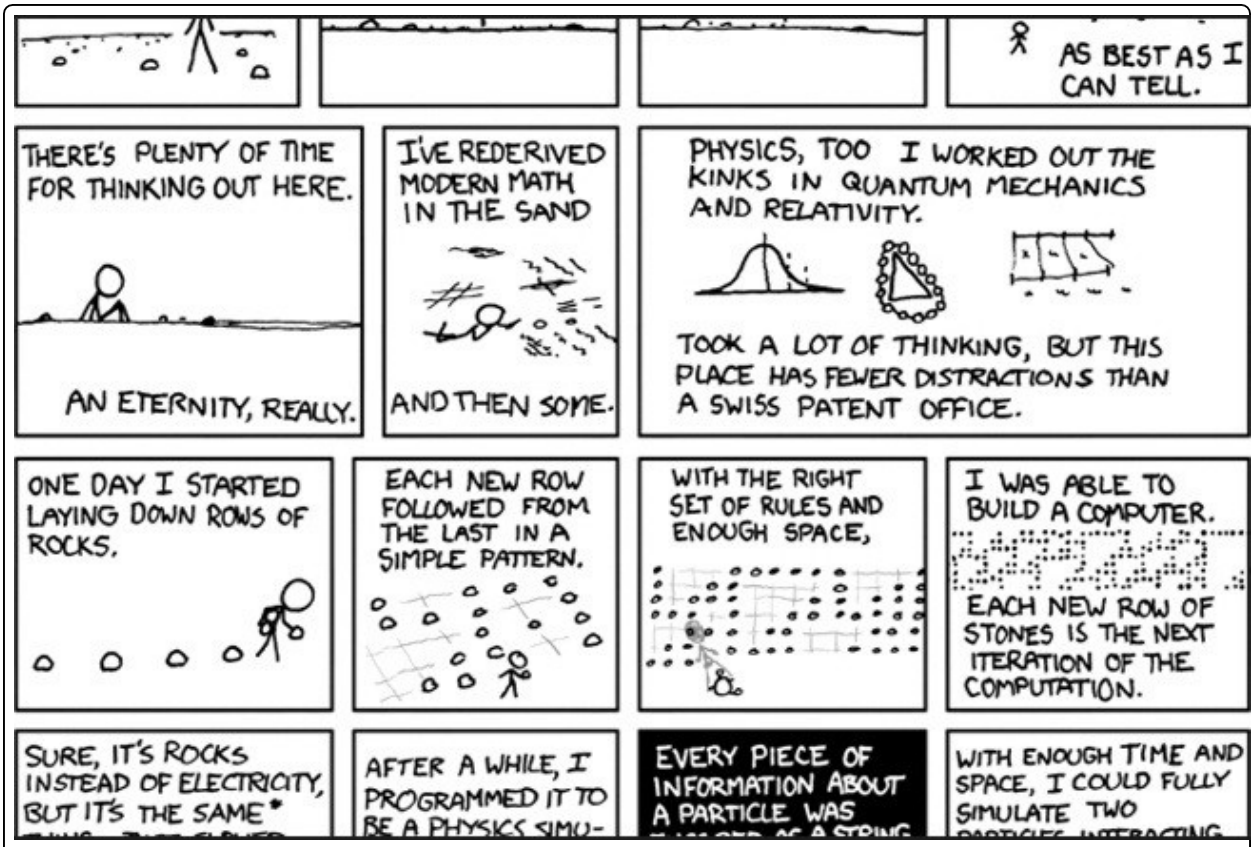


Figura 11-6. XKCD, "un webcómic de romance, sarcasmo, matemáticas y lenguaje"

Esto significa que su código tendrá que hacer lo siguiente:

- Descarga de páginas con el módulo de solicitudes.
- Encuentra la URL de la imagen de cómic de una página utilizando Beautiful Soup.
- Descarga y guarda la imagen del cómic en el disco duro con iter_content().
- Busca la URL del enlace Previous Comic, y repite.

Abre una nueva ventana del editor de archivos y guárdalo como *downloadXkcd.py*.

Paso 1: Diseñar el programa

Si abres las herramientas de desarrollo del navegador e inspeccionas los elementos de la página, encontrarás lo siguiente:

- La URL del archivo de imagen del cómic viene dada por el atributo href de un <img>. elemento.
- El elemento <img> está dentro de un elemento .
- El botón Prev tiene un atributo rel HTML con el valor prev.
- El botón Prev del primer cómic enlaza con la URL http://xkcd.com/#, lo que indica que no hay más páginas anteriores.

Haz que tu código sea como el siguiente:

```
#! python3
# downloadXkcd.py - Descarga todos y cada uno de los cómics XKCD.

import requests, os, bs4

url='http://xkcd.com'      # url inicial
os.makedirs('xkcd', exist_ok=True)      # almacenar comics en
./xkcd while not url.endswith('#'):
    # TODO: Descargar la página.

    # TODO: Encontrar la URL de la imagen del comic.

    # TODO: Descargar la .

    # TODO: Guardar la imagen en ./xkcd.

    # TODO: Obtener la url del boton Prev.

print('Hecho.')
```

Tendrás una variable url que empieza con el valor 'http://xkcd.com' y la actualizarás repetidamente (en un bucle for) con la URL del enlace Prev de la página actual. En cada paso del bucle, descargarás el cómic en url. Sabrás que debes terminar el bucle cuando url termine con '#'.

Descargará los archivos de imagen a una carpeta en el directorio de trabajo actual llamada *xkcd*. La llamada os.makedirs() asegura que esta carpeta existe, y la palabra clave exist_ok=True evita que la función lance un error

si esta carpeta ya existe. El resto del código son sólo comentarios que describen el resto del programa.

Paso 2: Descargar la página web

Implementemos el código para descargar la página. Haz que tu código sea como el siguiente:

```
#! python3
# downloadXkcd.py - Descarga todos y cada uno de los cómics XKCD.

import requests, os, bs4

url='http://xkcd.com'      # url inicial
os.makedirs('xkcd', exist_ok=True)      # almacenar comics en
./xkcd while not url.endswith('#'):
      # Descargar la página. print('Descargando
      página %s...' % url) RES= REQUESTS.get(url)
      RES.RAISE_FOR_STATUS()

      sopa= BS4.BeautifulSoup(RES.text)

      # TODO: Encontrar la URL de la imagen del comic.

      # TODO: Descargar la imagen.

      # TODO: Guardar la imagen en ./xkcd.

      # TODO: Obtener la url del boton Prev.

print('Hecho.')
```

Primero, imprime la url para que el usuario sepa cuál es la URL que el programa está a punto de descargar; luego usa la función request.get() del módulo requests para descargarla. Como siempre, llama inmediatamente al método raise_for_status() del objeto Response para lanzar una excepción y terminar el programa si algo salió mal con la descarga. De lo contrario, se crea un
Objeto BeautifulSoup a partir del texto de la página descargada.

Paso 3: Buscar y descargar la imagen del cómic

Haz que tu código sea como el siguiente:

```
#! python3
# downloadXkcd.py - Descarga todos y cada uno de los cómics XKCD.

import requests, os, bs4

--snip--

    # Busca la URL de la imagen del cómic.
    comicElem= soup.select('#comic img') if
    comicElem== []:
        print('No se ha encontrado la imagen del
    cómic.') else:
        comicUrl= 'http:'+ comicElem[0].get('src') #
        Descarga la imagen.
        print('Descargando imagen %s...' % (comicUrl)) RES=
        REQUESTS.get(comicUrl) RES.RAISE_FOR_STATUS()

    # TODO: Guardar la imagen en ./xkcd.

    # TODO: Obtener la url del boton Prev.

print('Hecho.')
```

Al inspeccionar la página de inicio de XKCD con tus herramientas de desarrollo, sabes que el elemento <img> para la imagen del cómic está dentro de un elemento con el atributo id definido como cómic, por lo que el selector '#comic img' te dará el resultado correcto
elemento <img> del objeto BeautifulSoup.

Algunas páginas XKCD tienen contenido especial que no es un simple archivo de imagen. No pasa nada; las omitirás. Si tu selector no encuentra ningún elementosoup.select('#comic img') devolverá una lista en blanco. Cuando eso ocurre, el programa puede simplemente imprimir un mensaje de error y seguir adelante sin descargar la imagen.

De lo contrario, el selector devolverá una lista que contiene un elemento <img>. Puedes obtener el atributo src de este elemento <img> y pasarlo a requests.get() para descargar el archivo de imagen del cómic.

Paso 4: Guardar la imagen y buscar el cómic anterior

Haz que tu código sea como el siguiente:

```
#! python3
# downloadXkcd.py - Descarga todos y cada uno de los cómics XKCD.

import requests, os, bs4

--snip--

        # Guarda la imagen en ./xkcd.
        imageFile= open(os.path.join('xkcd', os.path.basename(comicUrl)), 'wb') for chunk
        in res.iter_content(100000):
            imageFile.write(chunk) imageFile.close()

    # Obtener la url del botón Prev.
    prevLink = soup.select('a[rel="prev"]')[0]
    url= 'http://xkcd.com'+ prevLink.get('href')

print('Hecho.')
```

En este punto, el archivo de imagen del cómic se almacena en la variable res. Es necesario escribir estos datos de imagen en un archivo del disco duro.

Necesitarás un nombre de archivo para el archivo de imagen local para pasarlo a open(). El comicUrl tendrá un valor como 'http://imgs.xkcd.com/comics/heartbleed_explanation.png'-que te habrás dado cuenta que se parece mucho a una ruta de archivo. Y de hecho, puedes llamar a os.path.basename() con comicUrl, y te devolverá sólo la última parte de la URL, 'heartbleed_explanation.png'. Puedes utilizarlo como nombre de archivo cuando guardes la imagen en tu disco duro. Une este nombre con el nombre de tu carpeta xkcd usando os.path.join() para que tu programa use barras invertidas (\) en Windows y barras inclinadas (/) en OS X y Linux. Ahora que finalmente tienes el nombre del archivo, puedes llamar a open() para abrir un nuevo archivo en modo 'wb' "write binary".

Recuerda que para guardar los archivos que has descargado usando Peticiones, necesitas hacer un bucle sobre el valor de retorno del método iter_content(). El código en el bucle for escribe trozos de los datos de la imagen (como máximo

100.000 bytes cada uno) al archivo y luego lo cierras. La imagen se guardará en el disco duro.

Después, el selector 'a[rel="prev"]' identifica el elemento <a> con el atributo rel configurado como prev, y puedes usar el atributo href de este elemento <a> para obtener la URL del cómic anterior, que se almacena en url. A continuación, el bucle while comienza de nuevo todo el proceso de descarga para este cómic.

La salida de este programa tendrá el siguiente aspecto:

```
Descargando página http://xkcd.com...
Descargando imagen http://imgs.xkcd.com/comics/phone_alarm.png...
Descarga de la página http://xkcd.com/1358/...
Descargando imagen http://imgs.xkcd.com/comics/nro.png...
Descarga de la página http://xkcd.com/1357/...
Descarga de la imagen http://imgs.xkcd.com/comics/free_speech.png...
Descarga de la página http://xkcd.com/1356/...
Descargando imagen http://imgs.xkcd.com/comics/orbital_mechanics.png... Descarga de la
página http://xkcd.com/1355/...
Descarga de la imagen http://imgs.xkcd.com/comics/airplane_message.png... Descarga
de la página http://xkcd.com/1354/...
Descargando la imagen http://imgs.xkcd.com/comics/heartbleed_explanation.png...
--snip--
```

Este proyecto es un buen ejemplo de programa capaz de seguir automáticamente enlaces para extraer grandes cantidades de datos de la Web. Puedes conocer otras funciones de Beautiful Soup en su documentación en *http://www.crummy.com/software/BeautifulSoup/bs4/doc/*.

Ideas para programas similares

La descarga de páginas y el seguimiento de enlaces son la base de muchos programas de rastreo web. Programas similares también podrían hacer lo siguiente:

- Haz una copia de seguridad de todo un sitio siguiendo todos sus enlaces.
- Copia todos los mensajes de un foro web.
- Duplicar el catálogo de artículos a la venta en una tienda online.

Los módulos `requests` y `BeautifulSoup` son geniales siempre y cuando puedas averiguar la URL que necesitas pasar a `requests.get()`. Sin embargo, a veces esto no es tan fácil de encontrar. O quizás el sitio web por el que quieres que navegue tu programa requiere que te identifiques primero. El módulo `selenium` le dará a sus programas el poder de realizar tareas tan sofisticadas.

Control del navegador con el módulo selenium

El módulo selenium permite Python controlar directamente el navegador haciendo clic mediante programación en los enlaces y rellenando la información de acceso, casi como si hubiera un usuario humano interactuando con la página. Selenium te interactuar con páginas web de una forma mucho más avanzada que Requests y Beautiful Soup; pero debido a que lanza un navegador web, es un poco más lento y difícil de ejecutar en segundo plano si, digamos, sólo necesitas descargar algunos archivos de la Web.

El Apéndice A contiene pasos más detallados sobre la instalación de módulos de terceros.

Inicio de un navegador controlado por Selenium

Para estos ejemplos, necesitarás el navegador web Firefox. Este será el navegador que controlarás. Si aún no tienes Firefox, puedes descargarlo gratuitamente desde http://getfirefox.com/.

Importar los módulos para Selenium es un poco complicado. En lugar de importar selenium, necesitas ejecutar from selenium import webdriver. (La razón exacta por la que el módulo selenium está configurado de esta manera está más allá del alcance de este libro). Después de eso, puede iniciar el navegador Firefox con Selenium. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> from selenium import webdriver
>>> navegador= webdriver.Firefox()
>>> tipo(navegador)
<clase 'selenium.webdriver.firefox.webdriver.WebDriver'>
>>> browser.get('http://inventwithpython.com')
```

Observará que cuando se llama a webdriver.Firefox(), se el navegador web Firefox. Llamando a type() en el valor webdriver.Firefox() revela que es del tipo de datos WebDriver. Y llamando a browser.get('http://inventwithpython.com') dirige el navegador a http://inventwithpython.com/. El navegador debería tener un aspecto similar a Figura 11-7.



Figura 11-7. Después de llamar a webdriver.Firefox() *y* get() *en IDLE, aparece el navegador Firefox.*

Encontrar elementos en la página

Los objetos WebDriver tienen bastantes métodos para encontrar elementos en una página. Se dividen en los métodos find_element_* y find_elements_*. Los métodos find_element_* devuelven un único objeto WebElement, que representa el primer elemento de la página que coincide con la consulta. Los métodos find_elements_*
devuelven una lista de objetos WebElement_* para *cada* elemento coincidente de la página.

La Tabla 11-3 muestra varios ejemplos de métodos find_element_* y find_elements_* que se llaman en un objeto WebDriver que está almacenado en el navegador de variables.

Tabla 11-3. Métodos WebDriver *de Selenium para encontrar elementos*

Nombre del método	Objeto/lista WebElement devuelta
browser.find_element_by_class_name(*name*) browser.find_elements_by_class_name(*name*)	Elementos que utilizan el *nombre de* clase CSS
browser.find_element_by_css_selector(*selector*) browser.find_elements_by_css_selector(*selector*)	Elementos que coinciden con el *selector* CSS
browser.find_element_by_id(*id*) browser.find_elements_by_id(*id*)	Elementos con un valor de atributo *id* coincidente
browser.find_element_by_link_text(*text*) browser.find_elements_by_link_text(*text*)	elementos <a> que coincidan completamente con el *texto* proporcionado
browser.find_element_by_partial_link_text(*text*) browser.find_elements_by_partial_link_text(*text*)	Elementos <a> que contienen el *texto* proporcionado
browser.find_element_by_name(*name*) browser.find_elements_by_name(*name*)	Elementos con un valor de atributo *name* coincidente
browser.find_element_by_tag_name(*name*) browser.find_elements_by_tag_name(*name*)	Elementos con un *nombre de* etiqueta coincidente (no se distingue entre mayúsculas y minúsculas; un elemento <a> coincide con <a> y <a>)

A excepción de los métodos *_by_tag_name(), los argumentos de todos los métodos distinguen entre mayúsculas y minúsculas. Si no existen elementos en la página que coincidan con lo que el método está buscando, el módulo selenium lanza una excepción NoSuchElement. Si no desea que esta bloquee su programa, añada sentencias try y except a su código.

Una vez que tenga el objeto WebElement, puede averiguar más sobre él leyendo los atributos o llamando a los métodos de la Tabla 11-4.

Tabla 11-4. Atributos y métodos de WebElement

Atributo o método	Descripción
nombre_etiqueta	El nombre de la etiqueta, como 'a' para un elemento <a>.
get_attribute(*nombre*)	Valor del atributo name del elemento
texto	El texto dentro del elemento, como "hola" en hello.
clear()	En los elementos de campo de texto o área de texto, borra el texto escrito en ellos.
is_displayed()	Devuelve True si el elemento es visible; en caso contrario devuelve False
is_enabled()	Para los elementos de entrada, devuelve True si el elemento está habilitado; en caso contrario, devuelve Falso
is_selected()	Para los elementos de casilla de verificación o botón de radio, devuelve True si el elemento está seleccionado; en caso contrario, devuelve False
ubicación	Un diccionario con las claves 'x' e 'y' para la posición del elemento en la página

Por ejemplo, abra un nuevo editor de archivos e introduzca el siguiente programa:

```
from selenium import webdriver
navegador= webdriver.Firefox()
browser.get('http://inventwithpython.com') try:
    elem = browser.find_element_by_class_name("bookcover")
    print('¡Encontrado elemento <%s> con ese nombre de clase!" %
(elem.tag_name)) except:
    print("No se ha podido encontrar un elemento con ese nombre.")
```

Aquí abrimos Firefox y lo dirigimos a una URL. En esta página, tratamos de encontrar elementos con el nombre de clase 'bookcover', y si se un elemento de este tipoimprimimos su nombre de etiqueta utilizando el atributo tag_name. Si no se encuentra tal elemento,

imprimimos un mensaje diferente.

Este programa mostrará lo siguiente:

```
¡Encontrado elemento <img> con ese nombre de clase!
```

Encontramos un elemento con el nombre de clase 'bookcover' y el nombre de etiqueta "img".

Hacer clic en la página

Los objetos WebElement devueltos por los métodos find_element_* y find_elements_* tienen un método click() que simula un clic del ratón sobre ese elemento. Este método se puede utilizar para seguir un enlace, hacer una selección en un botón de radio, hacer clic en un botón Enviar o activar cualquier otra cosa que pueda ocurrir cuando el ratón haga clic en el elemento. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> from selenium import webdriver
>>> navegador= webdriver.Firefox()
>>> browser.get('http://inventwithpython.com')
>>> linkElem= browser.find_element_by_link_text('Léalo en línea')
>>> type(linkElem)
<clase 'selenium.webdriver.remote.webelement.WebElement'>
>>> linkElem.click() # SIGUE el enlace "Léalo en línea
```

Esto abre Firefox en http://inventwithpython.com/, obtiene el objeto WebElement para el elemento <a> con el texto *Léalo en línea* y, a continuación, simula que se hace clic en él elemento <a>. Es como si usted mismo hiciera clic en el enlace; el navegador sigue entonces ese enlace.

Cumplimentación y envío de formularios

Enviar pulsaciones de teclas a los campos de texto de una página web es cuestión de encontrar la tecla <input> o <textarea> para ese campo de texto y luego llamar al método send_keys(). Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> from selenium import webdriver
>>> navegador= webdriver.Firefox()
>>> browser.get('https://mail.yahoo.com')
>>> emailElem= browser.find_element_by_id('LOGIN-USERNAME')
>>> emailElem.SEND_KEYS('no_mi_email_real')
>>> passwordElem= browser.find_element_by_id('LOGIN-PASSWD')
>>> passwordElem.SEND_KEYS('12345')
>>> passwordElem.submit()
```

Siempre que Gmail no haya cambiado la id de los campos de texto Nombre de usuario y Contraseña desde la publicación de este libro, el código anterior rellenará esos campos de texto con el texto proporcionado. (Siempre puedes utilizar el inspector del navegador para verificar el id.) Llamar al método submit() en cualquier elemento tendrá el mismo resultado que hacer clic en el botón Enviar del formulario en el que se encuentra ese elemento. (También podrías haber llamado emailElem.submit(), y el código habría hecho mismo).

Envío de claves especiales

Selenium tiene un módulo para las teclas del teclado que son imposibles de escribir en un valor de cadena, que funcionan de forma muy parecida a los caracteres de escape. Estos valores se almacenan en atributos en el selenium.webdriver.common.keys. Dado que es un nombre de módulo tan largo, es mucho más fácil ejecutar from selenium.webdriver.common.keys import Keys en la parte superior de su programa; si lo hace, entonces puede simplemente escribir Keys en cualquier lugar donde normalmente tendría que escribir selenium.webdriver.common.keys. La Tabla 11-5 enumera las variables Keys de uso común.

Tabla 11-5. Variables de uso común en el selenium.webdriver.common.keys *Módulo*

Atributos	Significados
Teclas.ABAJO, Teclas.ARRIBA, Teclas.IZQUIERDA, Teclas.DERECHA	Las flechas del teclado
Teclas.ENTER, Teclas.RETURN	Las teclas ENTER y RETURN
Teclas.HOME, Teclas.END, Teclas.PAGE_DOWN, Teclas.PAGE_UP	Las teclas home, end, pagedown y pageup
Teclas.ESCAPE, Teclas.BACK_SPACE, Teclas.DELETE	Las teclas ESC, RETROCESO y SUPR
Teclas.F1, Teclas.F2,..., Teclas.F12	Las teclas F1 a F12 en la parte superior del teclado
Teclas.TAB	La tecla TAB

Por ejemplo, si el cursor no está actualmente en un campo de texto, pulsando las teclas HOME y END el navegador se desplazará a la parte superior e inferior de la página, respectivamente. Introduzca lo siguiente en el intérprete de comandos interactivo y observe cómo las llamadas a send_keys() desplazan la página:

```
>>> from selenium import webdriver
>>> from selenium.webdriver.common.KEYS import CLAVES
>>> navegador= webdriver.Firefox()
>>> browser.get('http://nostarch.com')
>>> htmlElem= browser.find_element_by_tag_name('html')
>>> htmlElem.SEND_KEYS(CLAVES.FIN)       # se DESPLAZA hasta abajo
>>> htmlElem.SEND_KEYS(TECLAS.HOME)      # se DESPLAZA a la parte superior
```

La etiqueta <html> es la etiqueta base de los archivos HTML: El contenido completo del archivo HTML

está encerrado dentro de las etiquetas <html> y </html>. Llamar a `browser.find_element_by_tag_name('html')` es un buen lugar para enviar claves a la página web general. Esto sería útil si, por ejemplo, se carga nuevo contenido una vez que te has desplazado hasta el final de la página.

Botones del navegador

Selenium puede simular clics en varios botones del navegador también a través de los siguientes métodos:

- **browser.back()**. Hace clic en el botón Atrás.
- **browser.forward()**. Pulsa el botón Avanzar.
- **browser.REFRESH()**. Pulsa el botón Actualizar/Recargar.
- **browser.quit()**. Pulsa el botón Cerrar ventana.

Más información sobre el selenio

Selenium puede hacer mucho más que las funciones descritas aquí. Puede modificar las cookies del navegador, hacer capturas de pantalla de páginas web y ejecutar JavaScript personalizado. Para obtener más información sobre estas funciones, puede visitar la documentación de Selenium en http://selenium-python.readthedocs.org/.

Resumen

La mayoría de las tareas aburridas no se limitan a los archivos de tu ordenador. Ser capaz de descargar páginas web mediante programación extenderá tus programas a Internet. El módulo requests hace que la descarga sea sencilla, y con algunos conocimientos básicos de conceptos HTML y selectores, puedes utilizar el módulo BeautifulSoup para analizar las páginas que descargues.

Pero para automatizar completamente cualquier tarea basada en web, necesita el control directo de su navegador web a través del módulo selenium. El módulo de selenio le permitirá iniciar sesión en sitios web y rellenar formularios automáticamente. Dado que un navegador web es la forma más común de enviar y recibir información a través de Internet, esta es una gran habilidad para tener en su caja de herramientas de programador.

Preguntas prácticas

P: 1. Describa brevemente las diferencias entre el navegador web, requests, BeautifulSoup y módulos de selenio.

P: 2. ¿Qué tipo de objeto devuelve requests.get()? ¿Cómo se puede acceder al contenido descargado como un valor de cadena?

P: 3. ¿Qué método de comprobación verifica que la descarga ha funcionado?

P: 4. ¿Cómo se puede obtener el código de estado HTTP de una respuesta de solicitud?

P: 5. ¿Cómo se guarda en un archivo una respuesta a una solicitud?

P: 6. ¿Cuál es el atajo de teclado para abrir las herramientas de desarrollo de un navegador?

P: 7. ¿Cómo se puede ver (en las herramientas para desarrolladores) el HTML de un elemento concreto de una página web?

P: 8. ¿Cuál es la cadena del selector CSS que encontraría el elemento con un atributo id de main?

P: 9. ¿Cuál es la cadena del selector CSS que encontraría los elementos con una clase CSS de resaltado?

P: 10. ¿Cuál es la cadena del selector CSS que encontraría todos los elementos dentro de otro ? ¿elemento?

P: 11. ¿Cuál es la cadena del selector CSS que encontraría el elemento <button> con un atributo value establecido en ¿favorito?

Q: 12. Digamos que tienes un objeto Tag de Beautiful Soup almacenado en la variable spam para el elemento ¡Hola mundo! . ¿Cómo podrías obtener una cadena '¡Hola mundo!' del objeto Tag?

P: 13. ¿Cómo almacenarías todos los atributos de un objeto Beautiful Soup Tag en una variable llamada ¿enlaceElem?

Q: 14. Ejecutar import selenium no funciona. ¿Cómo importar correctamente el módulo selenium?

P: 15. ¿Cuál es la diferencia entre los métodos find_element_* y find_elements_*?

Q: 16. ¿Qué métodos tienen los objetos WebElement de Selenium para simular los clics del ratón y las teclas del teclado?

P: 17. Podría llamar a send_keys(Keys.ENTER) en el objeto WebElement del botón Submit, pero ¿cuál es una forma más fácil de enviar un formulario con Selenium?

P: 18. ¿Cómo se puede simular la pulsación de los botones Adelante, Atrás y Actualizar de un navegador con Selenium?

Proyectos prácticos

Para practicar, escribe programas que realicen las siguientes tareas.

Emailer de línea de comandos

Escribe un programa que tome una dirección de correo electrónico y una cadena de texto en la línea de comandos y luego, utilizando Selenium, inicie sesión en tu cuenta de correo electrónico y envíe un correo electrónico con la cadena a la dirección proporcionada. (Es posible que desee configurar una cuenta de correo electrónico independiente para este programa).

Esta sería una buena manera de añadir una función de notificación a tus programas. También podrías escribir un programa similar para enviar mensajes desde una cuenta de Facebook o Twitter.

Descargador de imágenes

Escribe un programa que vaya a un sitio de intercambio de fotos como Flickr o Imgur, una categoría de fotos y luego descargue todas las imágenes resultantes. Podrías escribir un programa que funcione con cualquier sitio de fotos que tenga una función de búsqueda.

2048

2048 es un juego sencillo en el que combinas fichas deslizándolas hacia arriba, abajo, izquierda o derecha con las flechas del teclado. Puedes conseguir una puntuación bastante alta deslizando repetidamente hacia arriba, derecha, abajo e izquierda una y otra . Escribe un programa que abra el juego en *https://gabrielecirulli.github.io/2048/* y siga enviando pulsaciones de las teclas arriba, derecha, abajo e izquierda para jugar automáticamente.

Verificación de enlaces

Escriba un programa que, dada la URL de una página web, intente descargar todas las páginas enlazadas en la página. El programa debe marcar las páginas que tengan un código de estado 404 "No encontrado" e imprimirlas como enlaces rotos.

---

[2] La respuesta es no.

Capítulo 12. Trabajar con hojas de cálculo Excel

Excel es una popular y potente aplicación de hoja de cálculo para Windows. El módulo openpyxl permite a tus programas Python leer y modificar archivos de hojas de cálculo Excel. Por ejemplo, puede que tengas la aburrida tarea de copiar ciertos datos de una hoja de cálculo y pegarlos en otra. O puede que
tener que recorrer miles de filas y seleccionar sólo un puñado de ellas para hacer pequeñas modificaciones basadas en algún criterio. O puede que tengas que revisar cientos de hojas de cálculo de presupuestos de departamentos, buscando los que estén en números rojos. Este es exactamente el tipo de tareas aburridas y sin sentido que Python puede hacer por ti.

Aunque Excel es un software propietario de Microsoft, existen alternativas gratuitas que funcionan en Windows, OS X y Linux. Tanto LibreOffice Calc como OpenOffice Calc trabajan con el formato de archivo *.xlsx* de Excel para hojas de cálculo, lo que significa que el módulo openpyxl también puede trabajar con hojas de cálculo de estas aplicaciones. Puede descargar el software de *https://www.libreoffice.org/* y http://www.openoffice.org/, respectivamente. Incluso si ya tiene Excel instalado en su ordenador, puede que estos programas le resulten más fáciles de usar. Las capturas de pantalla de este , sin embargo, son todas de Excel 2010 en Windows 7.

Documentos Excel

En primer lugar, repasemos algunas definiciones básicas: Un documento de hoja de cálculo Excel se denomina *libro de trabajo*. Un solo libro se guarda en un archivo con la extensión *.xlsx*. Cada libro puede contener varias *hojas* (también llamadas *hojas de trabajo*). La hoja que el usuario está viendo en ese momento (o la última que vio antes de cerrar Excel) se denomina *hoja activa*.

Cada hoja tiene *columnas* (cuyas letras empiezan por *A*) y *filas* (cuyas cifras empiezan por 1). Una casilla situada en una columna y una fila determinadas se denomina *celda*. Cada celda puede contener un número o un valor de texto. La cuadrícula de celdas con datos constituye una hoja.

Instalación del módulo openpyxl

Python no viene con OpenPyXL, así que tendrás que instalarlo. Siga las instrucciones para instalar módulos de terceros en el Apéndice A; el nombre del módulo es openpyxl. Para comprobar si está instalado correctamente, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
```

Si el módulo se instaló correctamente, esto no debería producir ningún mensaje de error. Recuerda importar el módulo openpyxl antes de ejecutar los ejemplos de shell interactivo de este , o recibirás un error NameError: name 'openpyxl' is not defined.

Este libro cubre la versión 2.1.4 de OpenPyXL, pero el equipo de OpenPyXL publica nuevas versiones regularmente. Sin embargo, no se preocupe: Las nuevas versiones deberían seguir siendo compatibles con las instrucciones de este libro durante bastante tiempo. Si tienes una versión más reciente y quieres ver qué características adicionales pueden estar disponibles para ti, puedes consultar la documentación completa de OpenPyXL en http://openpyxl.readthedocs.org/.

Lectura de documentos Excel

Los ejemplos de este capítulo utilizarán una hoja de cálculo llamada *ejemplo.xlsx* almacenada en la carpeta raíz. Puedes crear la hoja de cálculo tú mismo o descargarla de http://nostarch.com/automatestuff/. La Figura 12-1 muestra las pestañas de las tres hojas predeterminadas denominadas *Hoja1*, *Hoja2* y *Hoja3* que Excel proporciona automáticamente para los libros nuevos. (El número de hojas por defecto creadas puede variar entre sistemas operativos y programas de hojas de cálculo).

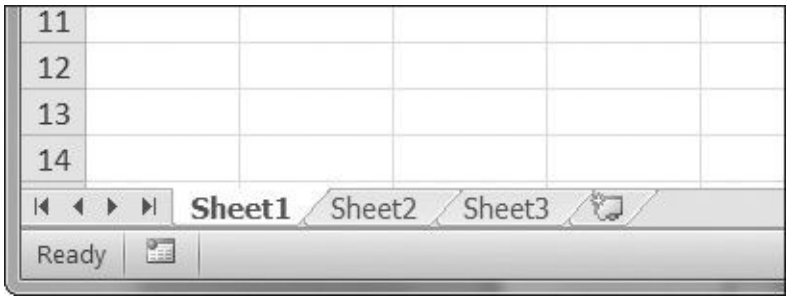


Figura 12-1. Las pestañas de las hojas de un libro se encuentran en la esquina inferior izquierda de Excel.

La hoja 1 del archivo de ejemplo debería parecerse a la Tabla 12-1. (Si no ha descargado *example.xlsx* del sitio web, deberá introducir usted mismo estos datos en la hoja).

Tabla 12-1. Hoja de cálculo example.xlsx

	A	B	C
1	4/5/2015 1:34:02 PM	Manzanas	73
2	4/5/2015 3:41:23 AM	Cerezas	85
3	4/6/2015 12:46:51 PM	Peras	14
4	4/8/2015 8:59:43 AM	Naranjas	52
5	4/10/2015 2:07:00 AM	Manzanas	152
6	4/10/2015 6:10:37 PM	Plátanos	23
7	4/10/2015 2:40:46 AM	Fresas	98

Ahora que tenemos nuestra hoja de cálculo de ejemplo, veamos cómo podemos manipularla con el módulo openpyxl.

Abrir documentos Excel con OpenPyXL

Una vez que hayas importado el módulo openpyxl, podrás utilizar la función openpyxl.load_workbook(). Introduce lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
>>> wb= openpyxl.load_workbook('ejemplo.xlsx')
>>> tipo(wb)
<clase 'openpyxl.workbook.workbook.Workbook'>
```

La función openpyxl.load_workbook() toma el nombre del archivo y devuelve un valor del tipo de datos Workbook. Este objeto Workbook representa el archivo Excel, un poco como un objeto File representa un archivo de texto abierto.

Recuerde que *ejemplo.xlsx* debe estar en el directorio de trabajo actual para poder trabajar con él. Puede averiguar cuál es el directorio de trabajo actual importando os y utilizando os.getcwd(), y puede cambiar el directorio de trabajo actual utilizando os.chdir().

Obtener hojas del cuaderno de trabajo

Puede obtener una lista de todos los nombres de hoja del libro de trabajo llamando a la función get_sheet_names(). Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
>>> wb= openpyxl.load_workbook('ejemplo.xlsx')
>>> wb.get_nombres_hoja()
['Hoja1', 'Hoja2', 'Hoja3']
>>> hoja= wb.get_sheet_by_name('Hoja3')
>>> hoja
<Hoja de trabajo "Hoja3">
>>> type(sheet) <class 'openpyxl.worksheet.worksheet.Worksheet'>
>>> título.hoja
Hoja3
>>> otraHoja= wb.get_active_sheet()
>>> otraHoja
<Hoja de trabajo "Hoja1">
```

Cada hoja está representada por un objeto Hoja de trabajoque puede obtener pasando la cadena del nombre de la hoja al método get_sheet_by_name() del libro de trabajo. Finalmente, puede llamar al método get_active_sheet() de un objeto Workbook para obtener la hoja activa del libro de trabajo. La hoja activa es la hoja que está en la parte superior cuando el libro se abre en Excel. Una vez que tenga el objeto Hoja de trabajo, puede
obtiene su nombre del atributo title.

Obtener celdas de las hojas

Una vez que tenga un objeto Hoja de cálculo, puede acceder a un objeto Celda por su nombre. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
>>> wb= openpyxl.load_workbook('ejemplo.xlsx')
>>> hoja= wb.get_sheet_by_name('Hoja1')
>>> hoja['A1']
<Celda Hoja1.A1>
>>> hoja['A1'].valor
datetime.datetime(2015, 4, 5, 13, 34, 2)
>>> c= hoja['B1']
>>> c.valor
Manzanas
>>> 'Fila '+ str(c.fila)+ ', Columna '+ c.columna+ ' es '+ c.valor
'Fila 1, Columna B es Manzanas'
>>> 'Celda '+ c.coordenada+ ' es '+ c.valor
'Celda B1 es Manzanas'
>>> hoja['C1'].valor
73
```

El objeto Cell tiene un atributo value que contiene, como era de esperar, el valor almacenado en esa celda. Los objetos Cell también tienen atributos de fila, columna y coordenada que proporcionan información sobre la ubicación de la celda.

Aquí, accediendo al atributo valor de nuestro objeto Cell para la celda B1 nos da la cadena 'Manzanas'. El atributo fila nos da el entero 1, el atributo columna nos da 'B', y el atributo coordenada nos da 'B1'.

OpenPyXL interpretará automáticamente las fechas en la columna A y las devolverá como valores datetime en lugar de cadenas. El tipo de datos datetime se explica con más detalle en el Capítulo 16.

Especificar una columna por letra puede ser complicado de programar, sobre todo porque después de la columna Z, las columnas empiezan utilizando dos letras: AA, AB, AC, etc. Como alternativa, también puede obtener una celda utilizando el método cell() de la hoja y pasando enteros para sus argumentos de palabra clave de fila y columna. El primer entero de fila o columna es 1, no 0. Continúe con el ejemplo del shell interactivo introduciendo lo siguiente:

```
>>> hoja.celda(fila=1, columna=2)
<Celda Hoja1.B1>
>>> hoja.celda(fila=1, columna=2).valor
```

```
Manzanas
>>> para i en rango(1, 8, 2):
        print(i, hoja.celda(fila=i, columna=2).valor)

1 Manzanas
3 Peras
5 Manzanas
7 Fresas
```

Como puedes ver, usando el método cell() de la hoja y pasándole row=1 y column=2 obtienes un objeto Cell para la celda B1, igual que especificando sheet['B1']. Entonces, usando el método cell() y sus argumentos de palabra clave, puedes escribir un bucle for para imprimir los valores de una serie de celdas.

Digamos que desea bajar por la columna B e imprimir el valor en cada celda con un número de fila impar. Pasando 2 para el parámetro "step" de la función range(), puede obtener celdas de cada segunda fila (en este caso, todas las filas impares). La variable **i** del bucle for se pasa como argumento de la palabra clave fila al método cell(), mientras que 2 se pasa siempre como argumento de la palabra clave columna. Tenga en cuenta que
se pasa el entero 2, no la cadena 'B'.

Puede determinar el tamaño de la hoja con los métodos get_highest_row() y get_highest_column() del objeto Worksheet. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
>>> wb= openpyxl.load_workbook('ejemplo.XLSX')
>>> hoja= wb.GET_SHEET_BY_NAME('Hoja1')
>>> hoja.GET_HIGHEST_ROW()
7
>>> hoja.GET_COLUMNA_MÁS_ALTA()
3
```

Tenga en cuenta que el método get_highest_column() devuelve un número entero en lugar de la letra que aparece en Excel.

Conversión entre letras de columna y números

Para convertir de letras a números, llame a la openpyxl.cell.column_index_from_string(). Para convertir de números a letras, llama a la función openpyxl.cell.get_column_letter(). Introduce lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
>>> from openpyxl.cell import get_column_letter, column_index_from_string
>>> get_column_letter(1)
'A'
>>> get_column_letter(2) 'B'
>>> get_column_letter(27)
AA
>>> get_column_letter(900)
AHP
>>> wb= openpyxl.load_workbook('ejemplo.XLSX')
>>> hoja= wb.GET_SHEET_BY_NAME('Hoja1')
>>> get_column_letter(sheet.GET_HIGHEST_COLUMN())
'C'
>>> COLUMN_INDEX_FROM_STRING('A')
1
>>> COLUMN_INDEX_FROM_STRING('AA')
27
```

Después de importar estas dos funciones del módulo openpyxl.cell, puedes llamar a get_column_letter() y pasarle un entero como 27 para averiguar cuál el nombre de la letra de la 27ª columna. La función column_index_string() hace lo contrario: Le pasas el nombre de la letra de una columna y te dice qué número tiene esa columna. No necesitas tener un libro de trabajo cargado para usar estas funciones. Si lo deseas, puedes cargar un libro de trabajo, obtener un objeto Hoja de cálculo y llamar a un método del objeto Hoja de cálculo como get_highest_column() para obtener un número entero. Luego, puede pasar ese entero a get_column_letter().

Obtención de filas y columnas de las hojas

Puede cortar objetos de hoja de cálculo para obtener todos los objetos celda de una fila, columna o área rectangular de la hoja de cálculo. A continuación, puede realizar un bucle sobre todas las celdas del corte. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
    >>> wb= openpyxl.load_workbook('ejemplo.xlsx')
    >>> hoja= wb.get_sheet_by_name('Hoja1')
    >>> tupla(hoja['A1':'C3'])
    ((<Celda Hoja1.A1>, <Celda Hoja1.B1>, <Celda Hoja1.C1>), (<Celda Hoja1.A2>,
    <Celda Hoja1.B2>, <Celda Hoja1.C2>), (<Celda Hoja1.A3>, <Celda Hoja1.B3>,
    <Celda Hoja1.C3>))

>>> ❶for filaObjetosDeCelda en hoja['A1':'C3']:
❷           for cellObj in rowOfCellObjects:
                print(cellObj.coordinate, cellObj.value)
            print('--- FIN DE FILA ---')
    A1 2015-04-05 13:34:02
    B1
    Manzanas
    C1 73
    --- FIN DE LA FILA.
    A2 2015-04-05 03:41:23
    B2 Cerezas
    C2 85
    --- FIN DE LA FILA.
    A3 2015-04-06 12:46:51
    B3 Peras
    C3 14
    --- FIN DE LA FILA.
```

Aquí, especificamos que queremos los objetos Cell en el área rectangular de A1 a C3, y obtenemos un objeto Generator que contiene los objetos Cell en esa área. Para ayudarnos a visualizar este objeto Generador, podemos utilizar tuple() sobre él para mostrar sus objetos Cell en una tupla.

Esta tupla contiene tres tuplas: una por cada fila, desde la parte superior del área hasta la parte inferior. Cada una de estas tres tuplas internas contiene los objetos Cell de una fila de nuestra área deseada, desde la celda situada más a la izquierda hasta la derecha. Así que, en total, nuestra porción de la hoja contiene todos los objetos Cell del área de A1 a C3, empezando por la celda superior izquierda y terminando por la celda inferior derecha.

Para imprimir los valores de cada celda del área, utilizamos dos bucles for. El bucle for recorre cada fila del corte❶. A continuación, para cada , el bucle for anidado

pasa por cada celda de esa fila❷.

Para acceder a los valores de las celdas de una fila o columna determinada, también puede utilizar el atributo filas y columnas de un objeto Hoja de cálculo. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
>>> wb= openpyxl.load_workbook('ejemplo.XLSX')
>>> hoja= wb.GET_ACTIVE_SHEET()
>>> hoja.COLUMNAS[1]
(<Celda Hoja1.B1>, <Celda Hoja1.B2>, <Celda Hoja1.B3>, <Celda Hoja1.B4>,
<Celda Hoja1.B5>, <Celda Hoja1.B6>, <Celda Hoja1.B7>)
>>> for cellObj in hoja.COLUMNAS[1]: print(cellObj.valor)

Manzanas
Cerezas Peras
Naranjas
Manzanas
Plátanos
Fresas
```

Utilizando el atributo rows en un objeto Worksheet obtendrá una tupla de tuplas. Cada una de estas tuplas internas representa una fila, y contiene los objetos Celda de esa fila. El atributo columns también te dará una tupla de tuplas, con cada una de las tuplas internas conteniendo los objetos Cell en una columna particular . Para *ejemplo.xlsx*, dado que hay 7 filas y 3 columnas, filas nos da una tupla de 7 tuplas (cada una contiene 3 objetos Celda), y columnas nos da una tupla de 3 tuplas (cada una contiene 7 objetos Celda).

Para acceder a una tupla concreta, puedes referirte a ella por su índice en la tupla mayor. Por ejemplo, para obtener la tupla que representa la columna B, se utiliza sheet.columns[1]. Para obtener la tupla que contiene los objetos Cell de la columna A, utilizarás sheet.columns[0]. Una vez que tienes una tupla que representa una fila o columna, puedes recorrer sus objetos Cell e imprimir sus valores.

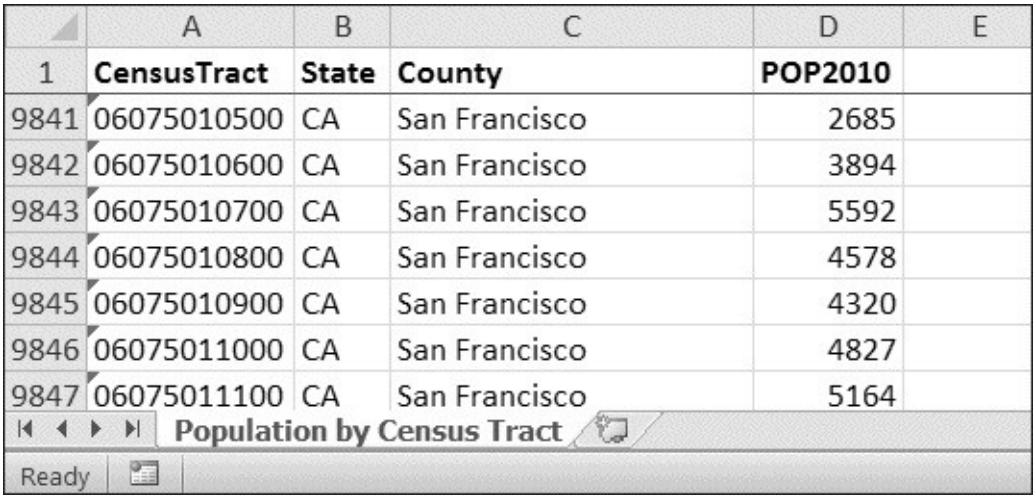
Libros, hojas y celdas

A modo de repaso rápido, he aquí un resumen de todas las funciones, métodos y tipos de datos que intervienen en la lectura de una celda de un archivo de hoja de cálculo:

1. Importa el módulo openpyxl.
2. Llama a la función openpyxl.load_workbook().
3. Obtener un objeto Workbook.
4. Llame al método get_active_sheet() o get_sheet_by_name() del libro de trabajo.
5. Obtener un objeto Hoja de cálculo.
6. Utilice la indexación o el método de hoja cell() con argumentos de palabra clave de fila y columna.
7. Obtener un objeto Cell.
8. Lee el atributo value del objeto Cell.

Proyecto: Lectura de datos de una hoja de cálculo

Supongamos que tiene una hoja de cálculo con los datos del censo de EE.UU. de 2010 y se enfrenta a la aburrida tarea de recorrer sus miles de filas para contar tanto la población total como el número de secciones censales de cada condado. (Una sección censal es simplemente un área geográfica definida a efectos del censo). Cada fila representa una única sección censal. Llamaremos al archivo de hoja de cálculo *censuspopdata.xlsx*, y puede descargarlo de http://nostarch.com/automatestuff/. Su contenido se parece al de la Figura 12-2.



	A	B	C	D	E
1	**CensusTract**	**State**	**County**	**POP2010**	
9841	06075010500	CA	San Francisco	2685	
9842	06075010600	CA	San Francisco	3894	
9843	06075010700	CA	San Francisco	5592	
9844	06075010800	CA	San Francisco	4578	
9845	06075010900	CA	San Francisco	4320	
9846	06075011000	CA	San Francisco	4827	
9847	06075011100	CA	San Francisco	5164	

Figura 12-2. Hoja de cálculo censuspopdata.xlsx

Aunque Excel puede calcular la suma de varias celdas seleccionadas, tendrías que seleccionar las celdas de cada uno de los más de 3.000 condados. Aunque sólo se tarda unos segundos en calcular a mano la población de un condado, se tardaría horas en hacerlo para toda la hoja de cálculo.

En este proyecto, escribirá una secuencia de comandos que puede leer desde el archivo de hoja de cálculo del censo y calcular estadísticas para cada condado en cuestión de segundos.

Esto es lo que hace su programa:

- Lee los datos de la hoja de cálculo Excel.
- Cuenta el número de secciones censales de cada condado.
- Cuenta la población total de cada condado.
- Imprime los resultados.

Esto significa que su código tendrá que hacer lo siguiente:

- Abra y lea las celdas de un documento Excel con el módulo openpyxl.
- Calcular todos los datos de tracto y población y almacenarlos en una estructura de datos.
- Escribe la estructura de datos en un archivo de texto con extensión *.py* utilizando la función pprint módulo.

Paso 1: Leer los datos de la hoja de cálculo

En la hoja de cálculo *censuspopdata.xlsx* sólo hay una hoja, denominada "Población por sección censal", y cada fila contiene los datos de una única sección censal. Las columnas son el número de sección (A), la abreviatura del estado (B), el nombre del condado (C) y la población de la sección (D).

Abra una nueva ventana del editor de archivos e introduzca el siguiente código. Guarde el archivo como *readCensusExcel.py*.

```
#! python3
    # readCensusExcel.py - Tabula la población y el número de secciones censales # de
    cada condado.

❶ import openpyxl, pprint print("Abriendo
    libro...')
❷ wb= openpyxl.load_workbook('censuspopdata.xlsx')
➲   hoja= wb.get_sheet_by_name("Population by Census Tract')
    countyData= {}

    # TODO: Rellenar countyData con la población y los tramos de cada condado.
    print('Leyendo filas...')
❹ para fila en rango(2, hoja.get_highest_row()+ 1):
        # Cada fila de la hoja de cálculo contiene los datos de una sección
        censal. state   = hoja['B'+ str(fila)].valor
        condado= hoja['C'+ str(fila)].valor pop
                = hoja['D'+ str(fila)].valor

    # TODO: Abrir un nuevo fichero de texto y escribir en el contenido de countyData.
```

Este código importa el módulo openpyxl, así como el módulo pprint que utilizarás para imprimir los datos finales del condado ❶. A continuación, abre el archivo *censuspopdata.xlsx*❷, obtiene la hoja con los datos del censo❸, y comienza a iterar sobre sus filas
❹.

Observe que también ha creado una variable llamada countyData, que contendrá las poblaciones y el número de tramos que calcule para cada condado. Sin embargo, antes de almacenar nada en ella, debes determinar exactamente cómo estructurarás los datos que contiene.

Paso 2: Rellenar la estructura de datos

La estructura de datos almacenada en countyData será un diccionario cuyas claves serán las abreviaturas de los estados. Cada abreviatura de estado se asignará a otro diccionario, cuyas claves son cadenas de los nombres de los condados de ese estado. A su vez, cada nombre de condado se asignará a un diccionario con sólo dos claves, **"tracts"** y **"pop"**. Estas claves corresponden al número de secciones censales y a la población del condado. Por ejemplo,
el diccionario tendrá un aspecto similar a éste:

```
{'AK': {'Aleutians East': {'pop': 3141, 'tracts': 1},
   'Aleutians West': {'pop': 5561, 'tracts': 2},
  'Anchorage': {'pop': 291826, 'tracts': 55},
  'Bethel': {'pop': 17013, 'tracts': 3},
  'Bristol Bay': {'pop': 997, 'tracts': 1},
  --snip--
```

Si el diccionario anterior estuviera almacenado en countyData, las siguientes expresiones se evaluarían así:

```
>>> countyData['AK']['Anchorage']['pop']
291826
>>> countyData['AK']['Anchorage']['tracts'] 55
```

De forma más general, las claves del diccionario countyData tendrán el siguiente aspecto:

```
countyData[state abbrev][county]['tracts']
countyData[state abbrev][county]['pop']
```

Ahora que sabes cómo se estructurará countyData, puedes escribir el código que lo llenará con los datos del condado. Añade el siguiente código al final de tu programa:

```
#! python 3
    # readCensusExcel.py - Tabula la población y el número de secciones censales # de
    cada condado.

    --snip--
    for row in range(2, sheet.get_highest_row()+ 1):
        # Cada fila de la hoja de cálculo contiene los datos de una sección
        censal. state   = hoja['B'+ str(fila)].valor
        condado= hoja['C'+ str(fila)].valor pop
                = hoja['D'+ str(fila)].valor

        # Asegúrese de que la clave para ESTE estado EXISTE.
❶       countyData.setdefault(state,   {})
```

```
            # Asegúrese de que la clave para ESTE condado en ESTE estado EXISTE.
❷           countyData[state].setdefault(county, {'TRACTS': 0, 'pop': 0})

            # Cada fila REPRESENTA una sección censal, así que incrementa de uno en uno.
➲           countyData[state][county]['TRACTS'] 1   +=
            # Aumente la población del condado por la población de ESTA sección censAL.
❹           countyData[state][county]['pop']+= int(pop)

        # TODO: Abrir un nuevo fichero de texto y escribir en el contenido de countyData.
```

Las dos últimas líneas de código realizan el trabajo de cálculo real, incrementando el valor de `tracts` ❸ y aumentando el valor de `pop` ❹ para el condado actual en cada iteración del bucle `for`.

El otro código está ahí porque no se puede añadir un diccionario de condado como valor para una clave de abreviatura de estado hasta que la propia clave exista en `countyData`. (Es decir, `countyData['AK']['Anchorage']['tracts']+= 1` provocará un error si la clave `'AK'` aún no existe). Para asegurarte de que la clave de abreviatura de estado existe en tu estructura de datos, tienes que llamar al método `setdefault()` para establecer un valor si aún no existe uno para `el estado`❶.

Al igual que el diccionario `countyData` necesita un diccionario como valor para cada clave de abreviatura de estado, cada uno de *esos* diccionarios necesitará su propio diccionario como valor para cada clave de condado ❷. Y cada uno de *esos* diccionarios necesitará a su vez claves `'tracts'` y `'pop'` que empiecen por el valor entero `0`. (Si alguna vez pierdes la noción de la estructura de diccionarios, vuelve a mirar el diccionario de ejemplo al principio de esta sección).

Dado que `setdefault()` no hará nada si la clave ya existe, puede llamarla en cada iteración del bucle `for` sin problema.

Paso 3: Escribir los resultados en un archivo

Una vez finalizado el bucle for, el diccionario countyData contendrá toda la información sobre población y tramos codificada por condado y estado. En este punto, podría programar más código para escribir esto en un archivo de texto o en otra hoja de cálculo de Excel. Por ahora, vamos a utilizar la función pprint.pformat() para escribir el valor del diccionario countyData como una cadena masiva en un archivo llamado *census2010.py*. Añada la función
siguiente código al final de su programa (asegurándose de mantenerlo sin sangría para que quede fuera del bucle for):

```
#! python 3
# readCensusExcel.py - Tabula la población y el número de secciones censales # de
cada condado.
--snip--

for row in range(2, sheet.get_highest_row()+ 1):
--snip--

# Abre un nuevo fichero de texto y escribe en el contenido de countyData.
print('Escribiendo RESULTADOS...')
resultFile= open('CENSUS2010.py', 'w') resultFile.write('allData= '+
pprint.pformat(countyData)) resultFile.close()
print('Hecho.')
```

La función pprint.pformat() produce una cadena que a su vez está formateada como código Python válido. Al enviarla a un archivo de texto llamado *census2010.py*, ¡has generado un programa Python a partir de tu programa Python! Esto puede parecer complicado, pero la ventaja es que ahora puede importar *census2010.py* como cualquier otro módulo de Python. En el intérprete de comandos interactivo, cambie el directorio de trabajo actual a la carpeta con el archivo *census2010.py* recién creado (en mi portátil, es *C:\Python34*) y, a continuación, impórtelo:

```
>>> importar os
>>> os.chdir('C:\\Python34')
>>> importar censo2010
>>> CENSUS2010.allData['AK']['Anchorage']
{'pop': 291826, 'tracts': 55}
>>> anchoragePop = CENSUS2010.allData['AK']['Anchorage']['pop']
>>> print('La población de Anchorage en 2010 ERA '+ str(anchoragePop))
La población de Anchorage en 2010 era de 291826 habitantes.
```

El programa *readCensusExcel.py* era código desechable: Una vez que tengas sus resultados guardados en *census2010.py*no necesitarás volver a ejecutar el programa.

Siempre que necesites los datos de los condados, puedes ejecutar `import censuss2010`.

Calcular estos datos a mano habría llevado horas; este programa lo hizo en unos segundos. Utilizando OpenPyXL, no tendrá ningún problema para extraer información guardada en una hoja de cálculo Excel y realizar cálculos con ella. Puedes descargar el programa completo en http://nostarch.com/automatestuff/.

Ideas para programas similares

Muchas empresas y oficinas utilizan Excel para almacenar diversos tipos de datos, y no es raro que las hojas de cálculo se vuelvan grandes y difíciles de manejar. Cualquier programa que analice una hoja de cálculo de Excel tiene una estructura similar: Carga el archivo de la hoja de cálculo, prepara algunas variables o estructuras de datos y, a continuación, recorre cada una de las filas de la hoja de cálculo. Un programa de este tipo podría hacer lo siguiente:

- Comparar datos entre varias filas de una hoja de cálculo.
- Abra varios archivos de Excel y compare datos entre hojas de cálculo.
- Comprueba si una hoja de cálculo tiene filas en blanco o datos no válidos en alguna celda y avisa al usuario en caso afirmativo.
- Lea datos de una hoja de cálculo y utilícelos como entrada para sus programas Python.

Redacción de documentos Excel

OpenPyXL también proporciona formas de escribir datos, lo que significa que tus programas pueden crear y editar archivos de hojas de cálculo. Con Python, es sencillo crear hojas de cálculo con miles de filas de datos.

Crear y guardar documentos Excel

Llama a la función `openpyxl.Workbook()` para crear un nuevo `Workbook` en blanco objeto. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
>>> wb= openpyxl.Workbook()
>>> wb.GET_NOMBRES_HOJA()
['Hoja']
>>> hoja= wb.GET_ACTIVE_SHEET()
>>> título.hoja
Hoja
>>> sheet.title= 'Hoja HUEVOS con Tocino y Jamón'
>>> wb.GET_NOMBRES_HOJA()
['Hoja de Huevos con Tocino y Jamón']
```

El libro de trabajo comenzará con una única hoja llamada *Hoja*. Puede cambiar el nombre de la hoja almacenando una nueva cadena en su atributo `title`.

Cada vez que modifique el objeto `Workbook` o sus hojas y celdas, el archivo de hoja de cálculo no se guardará hasta que llame al método `save()` workbook. Introduzca lo en el intérprete de comandos interactivo (con *ejemplo.xlsx* en el directorio de trabajo actual):

```
>>> importar openpyxl
>>> wb= openpyxl.load_workbook('ejemplo.XLSX')
>>> hoja= wb.GET_ACTIVE_SHEET()
>>> sheet.title= 'Spam Spam Spam'
>>> wb.save('ejemplo_copia.XLSX')
```

Aquí, cambiamos el nombre de nuestra hoja. Para guardar nuestros cambios, pasamos un nombre de archivo como cadena al método `save()`. Pasando un nombre de archivo diferente al original, como `"ejemplo_copia.xlsx"`, guardamos los cambios en una copia de la hoja de cálculo.

Siempre que edites una hoja de cálculo que hayas cargado desde un archivo, debes guardar la nueva hoja de cálculo editada con un nombre de archivo distinto del original. De esta forma, seguirás teniendo el archivo original de la hoja de cálculo para trabajar con él en caso de que un error en el código provoque que el nuevo archivo guardado contenga datos incorrectos o corruptos.

Creación y eliminación de hojas

Las hojas pueden añadirse y eliminarse de un libro de trabajo con la función create_sheet() y remove_sheet(). Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
>>> wb= openpyxl.Workbook()
>>> wb.GET_NOMBRES_HOJA()
['Hoja']
>>> wb.CREAR_HOJA()
<Hoja de trabajo "Hoja1">
>>> wb.GET_NOMBRES_HOJA()
['Hoja', 'Hoja1']
>>> wb.CREATE_SHEET(index=0, title='Primera hoja')
<Hoja de trabajo "Primera hoja">
>>> wb.GET_NOMBRES_HOJA()
['Primera hoja', 'Hoja', 'Hoja1']
>>>  wb.CREATE_SHEET(index=2,  title='Hoja  Medio')
<Hoja de trabajo "Hoja intermedia">
>>> wb.GET_NOMBRES_HOJA()
['Primera hoja', 'Hoja', 'Hoja intermedia', 'Hoja1']
```

El método create_sheet() devuelve un nuevo objeto Hoja de cálculo llamado *HojaX*, que por defecto se establece como la última hoja del libro de trabajo. Opcionalmente, el índice y el nombre de la nueva hoja se pueden especificar con los argumentos de palabra clave índice y título.

Continúe con el ejemplo anterior introduciendo lo siguiente:

```
>>> wb.GET_NOMBRES_HOJA()
['Primera hoja', 'Hoja', 'Hoja intermedia', 'Hoja1']
>>> wb.REMOVE_SHEET(wb.GET_SHEET_BY_NAME('Hoja_del_medio'))
>>> wb.REMOVE_SHEET(wb.GET_SHEET_BY_NAME('Hoja1'))
>>> wb.GET_NOMBRES_HOJA()
['Primera hoja', 'Hoja']
```

El método remove_sheet() toma como argumento un objeto Worksheet, no una cadena con el nombre de la hoja. Si sólo conoce el nombre de una hoja que desea eliminar, llame a get_sheet_by_name() y pase su valor de retorno a remove_sheet().

Recuerde llamar al método save() para guardar los cambios después de añadir o eliminar hojas del libro de trabajo.

Escribir valores en celdas

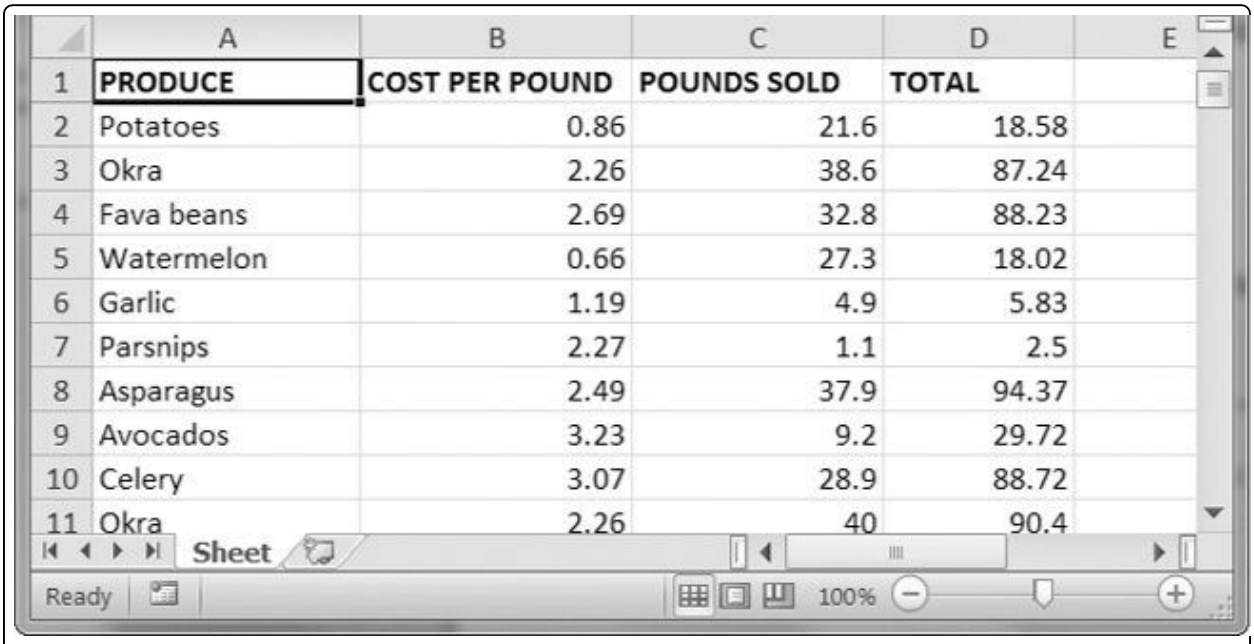
Escribir valores en las celdas es muy parecido a escribir valores en las claves de un diccionario. Introduzca esto en el intérprete de comandos interactivo:

```
>>> importar openpyxl
>>> wb= openpyxl.Workbook()
>>> hoja= wb.GET_SHEET_BY_NAME('Hoja')
>>> hoja['A1']= '¡Hola mundo!'
>>> hoja['A1'].valor
"¡Hola mundo!
```

Si tiene la coordenada de la celda como cadena, puede utilizarla como una clave de diccionario en el objeto Hoja de cálculo para especificar en qué celda escribir.

Proyecto: Actualización de una hoja de cálculo

En este proyecto, escribirás un programa para actualizar celdas en una hoja de cálculo de ventas de productos agrícolas. Tu programa buscará en la hoja de cálculo, encontrará tipos específicos de productos y actualizará sus precios. Descarga esta hoja de cálculo de http://nostarch.com/automatestuff/. La Figura 12-3 muestra el aspecto de la hoja de cálculo.



	A	B	C	D	E
1	PRODUCE	COST PER POUND	POUNDS SOLD	TOTAL	
2	Potatoes	0.86	21.6	18.58	
3	Okra	2.26	38.6	87.24	
4	Fava beans	2.69	32.8	88.23	
5	Watermelon	0.66	27.3	18.02	
6	Garlic	1.19	4.9	5.83	
7	Parsnips	2.27	1.1	2.5	
8	Asparagus	2.49	37.9	94.37	
9	Avocados	3.23	9.2	29.72	
10	Celery	3.07	28.9	88.72	
11	Okra	2.26	40	90.4	

Figura 12-3. Hoja de cálculo de ventas de productos agrícolas

Cada fila representa una venta individual. Las columnas son el tipo de producto vendido (A), el coste por libra de ese producto (B), el número de libras vendidas (C) y los ingresos totales de la venta (D). La columna TOTAL se ajusta con la fórmula de Excel *=ROUND(B3*C3, 2)*, que multiplica el coste por libra por el número de libras vendidas y redondea el resultado al céntimo más próximo. Con esta fórmula, las celdas de la columna TOTAL se actualizarán automáticamente si se produce un cambio en las columnas B o C.

Ahora imagine que los precios del ajo, el apio y los limones se introdujeron incorrectamente, dejándole con la aburrida tarea de recorrer miles de filas en esta hoja de cálculo para actualizar el coste por libra de cualquier fila de ajo, apio y limón. No puede hacer una simple búsqueda y sustitución del precio porque podría haber otros artículos con el mismo precio que no quiere "corregir" por error. Para miles de filas, esto llevaría horas hacerlo a mano. Pero puede escribir un programa que lo haga en segundos.

Su programa hace lo siguiente:

- Lazos sobre todas las filas.
- Si la fila es de ajo, apio o limones, cambia el precio. Esto significa

que su código tendrá que hacer lo siguiente:

- Abra el archivo de hoja de cálculo.
- Para cada fila, compruebe si el valor de la columna A es Apio, Ajo o Limón.
- Si lo es, actualice el precio en la columna B.
- Guarda la hoja de cálculo en un archivo nuevo (para no perder la hoja antigua, por si acaso).

Paso 1: Configurar una estructura de datos con la información de actualización

Los precios que debe actualizar son los siguientes:

Apio	1.19
Ajo	3.07
Limón	1.27

Podrías escribir código como este:

```
if produceName== 'Apio':
    cellObj= 1.19
if produceName== 'Ajo':
    cellObj= 3.07
if produceName== "Limón":
    cellObj= 1.27
```

Tener los datos de los productos y los precios actualizados codificados de esta manera es poco elegante. Si necesitara actualizar la hoja de cálculo de nuevo con precios diferentes o productos diferentes, tendría que cambiar gran parte del código. Cada vez que cambias el código, corres el riesgo de introducir errores.

Una solución más flexible es almacenar la información del precio corregido en un diccionario y escribir su código para utilizar esta estructura de datos. En una nueva ventana del editor de archivos, introduzca el siguiente código:

```
#! python3
# updateProduce.py - Corrige los costes en la hoja de ventas de productos

import openpyxl

wb= openpyxl.load_workbook('producirVentas.xlsx') hoja
=   wb.get_sheet_by_name("Hoja")

# Los tipos de producto y sus precios actualizados
PRICE_UPDATES= {"Ajo": 3.07,
                Apio': 1,19,
                "Limón":   1.27}

# TODO: Recorrer las filas y actualizar los precios.
```

Guárdalo como *updateProduce.py*. Si necesita actualizar la hoja de cálculo de nuevo,

tendrá que actualizar sólo el diccionario PRICE_UPDATES, no cualquier otro código.

Paso 2: Comprobar todas las filas y actualizar los precios incorrectos

La siguiente parte del programa recorrerá todas las filas de la hoja de cálculo. Añade el siguiente código al final de *updateProduce.py*:

```
#! python3
    # updateProduce.py - Corrige los costes en la hoja de ventas de productos.

    --snip--

    # Recorre las FILAS y actualiza los PRECIOS.
❶ for rowNum in range(2, sheet.GET_HIGHEST_ROW()):              # saltar la primera fila
❷         produceNombre= hoja.celda(fila=NumFila, columna=1).valor
➲         si produceNombre en PRICE_UPDATES:
                sheet.cell(row=rowNum, column=2).value= PRICE_UPDATES[produceName]

❹ wb.save('UPDATEDPRODUCESALES.XLSX')
```

Hacemos un bucle a través de las filas empezando por la fila 2, ya que la fila 1 es sólo la cabecera ❶. La celda de la columna 1 (es decir, la columna A) se almacenará en la variable produceNombre

❷. Si produceName existe como clave en el diccionario PRICE_UPDATES❸, sabrá que se trata de una fila cuyo precio debe corregirse. El precio correcto estará en PRICE_UPDATES[nombre_producto].

Fíjese en lo limpio que queda el al utilizar PRICE_UPDATES. Sólo necesaria una sentencia `if`, en lugar de código como `if nombreproducto== "Ajo":`, para actualizar cada tipo de producto. Y como el código utiliza el diccionario PRICE_UPDATES en lugar de codificar los nombres de los productos y los costes actualizados en el bucle for, sólo se modifica el diccionario PRICE_UPDATES y no el código si la hoja de cálculo de ventas de productos necesita cambios adicionales.

Después de recorrer toda la hoja de cálculo y hacer los cambios, el código guarda el objeto Workbook en *updatedProduceVentas.xlsx* ❹. No sobrescribe la hoja de cálculo antigua por si hay un error en tu programa y la hoja de cálculo actualizada es incorrecta. Después de comprobar que la hoja de cálculo actualizada parece correcta,
puedes borrar la hoja de cálculo antigua.

Puede descargar el código fuente completo de este programa en
http://nostarch.com/automatestuff/.

Ideas para programas similares

Dado que muchos trabajadores de oficina utilizan hojas de cálculo Excel todo el tiempo, un programa que pueda editar y escribir automáticamente archivos Excel podría ser realmente útil. Un programa de este tipo podría hacer lo siguiente:

- Leer datos de una hoja de cálculo y escribirlos en partes de otras hojas de cálculo.
- Lea datos de sitios web, archivos de texto o el portapapeles y escríbalos en una hoja de cálculo.
- Limpia" automáticamente los datos de las hojas de cálculo. Por ejemplo, podría utilizar expresiones regulares para leer múltiples formatos de números de teléfono y editarlos a un formato único y estándar.

Configuración del estilo de fuente de las celdas

Estilizar determinadas celdas, filas o columnas puede ayudarte a resaltar áreas importantes de la hoja de cálculo. En la hoja de cálculo de productos agrícolas, por ejemplo, su programa podría aplicar texto en negrita a las filas de patatas, ajos y chirivías. O tal vez desee poner en cursiva todas las filas con un coste por libra superior a 5 dólares. Estilizar partes de una hoja de cálculo grande a mano sería tedioso, pero tus programas pueden hacerlo al instante.

Para personalizar los estilos de fuente en las celdas, importante, importar las funciones Font() y Style() del módulo openpyxl.styles.

```
from openpyxl.styles import Fuente, Estilo
```

Esto le permite escribir Font() en lugar de openpyxl.styles.Font(). (Consulte Importar módulos para revisar este estilo de sentencia import).

A continuación se muestra un ejemplo que crea un nuevo libro de trabajo y establece la celda A1 para que tenga una fuente de 24 puntos en cursiva. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
    >>> from openpyxl.STYLES import Fuente, Estilo
    >>> wb= openpyxl.Workbook()
    >>> hoja= wb.GET_SHEET_BY_NAME('Hoja')

>>> ❶italic24Font= Font(size=24, italic=True)

>>> ❷styleObj= Style(font=italic24Font)

➲  >>> hoja['A1'].style= styleObj
    >>> hoja['A1']= '¡Hola mundo!'
    >>> wb.save('styled.XLSX')
```

OpenPyXL representa la colección de ajustes de estilo para una celda con un Style, que se almacena en el atributo style del objeto Cell. El estilo de una se puede establecer asignando el objeto Style al atributo style.

En este ejemplo, Font(size=24, italic=True) devuelve un objeto Font, que se almacena en italic24Font❶. Los argumentos de palabra clave de Font(), tamaño y cursiva, configuran los atributos de estilo del objeto Font. Este objeto Font se pasa entonces a la llamada Style(font=italic24Font), que devuelve el valor que almacenó en styleObj❷. Y cuando styleObj se asigna al estilo de la celda ❸, toda esa información de estilo de fuente se aplica a la celda A1.

Objetos tipográficos

Los atributos de estilo de los objetos Font afectan a la forma en que se muestra el texto en las celdas. Para establecer atributos de estilo de fuente, se pasan argumentos de palabra clave a Font(). La Tabla 12-2 muestra los posibles argumentos de palabra clave para la función Font().

Tabla 12-2. Argumentos de palabra clave para atributos de estilo de *fuente*

Argumento clave	Tipo de datos	Descripción
nombre	Cadena	El nombre de la fuente, como "Calibri" o "Times New Roman"
talla	Entero	El tamaño del punto
negrita	Booleano	Verdadero, para fuente en negrita
cursiva	Booleano	True, para letra cursiva

Puede llamar a Font() para crear un objeto Font y almacenarlo en una variable. Luego se pasa a Style(), se almacena el objeto Style resultante en una variable, y se asigna esa variable al atributo style de un objeto Cell. Por ejemplo, este código crea varios estilos de fuente:

```
>>> importar openpyxl
>>> from openpyxl.styles import Fuente, Estilo
>>> wb= openpyxl.Workbook()
>>> hoja= wb.get_sheet_by_name('Hoja')

>>> fontObj1= Font(name='Times New Roman', bold=True)
>>> styleObj1= Estilo(font=fontObj1)
>>> hoja['A1'].style/styleObj
>>> hoja['A1']= 'Negrita Times New Roman'

>>> fontObj2= Fuente(size=24, italic=True)
>>> styleObj2= Estilo(font=fontObj2)
>>> hoja['B3'].style/styleObj
>>> sheet['B3']= '24 pt Italic'

>>> wb.save('estilos.xlsx')
```

, almacenamos un objeto Fuente en fontObj1 y lo utilizamos para crear un objeto Estilo, que almacenamos en styleObj1, y luego establecemos el atributo style del objeto Celda A1 a styleObj. Repetimos el proceso con otro objeto Fuente y otro objeto Estilo

para establecer el estilo de una segunda celda. Después de ejecutar este código, los estilos de las celdas A1 y B3 de la hoja de cálculo se establecerán con estilos de fuente personalizados, como se muestra en
Figura 12-4.

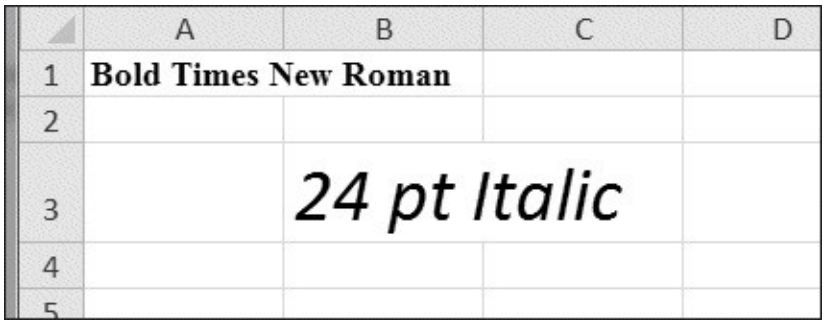


Figura 12-4. Una hoja de cálculo con estilos de fuente personalizados

Para la celda A1, establecemos el nombre de la fuente a `"Times New Roman"` y establecemos `negrita` a `true`, de modo que nuestro texto aparece en negrita Times New Roman. No especificamos un tamaño, por lo que se el predeterminado de openpyxl, 11. En la celda B3, nuestro texto está en cursiva, con un tamaño de 24; no especificamos un nombre de fuente, por lo que se utiliza la predeterminada de openpyxl, Calibri.

Fórmulas

Las fórmulas, que comienzan con un signo igual, pueden configurar celdas para que contengan valores calculados a partir de otras celdas. En esta sección, utilizarás el módulo openpyxl para añadir fórmulas a las celdas mediante programación, como cualquier valor normal. Por ejemplo:

```
>>> hoja['B9']= '=SUMA(B1:B8)'
```

Esto almacenará *=SUMA(B1:B8)* como valor en la celda B9. Esto establece la celda B9 en una fórmula que calcula la suma de los valores de las celdas B1 a B8. Puedes ver esto en acción en la Figura 12-5.

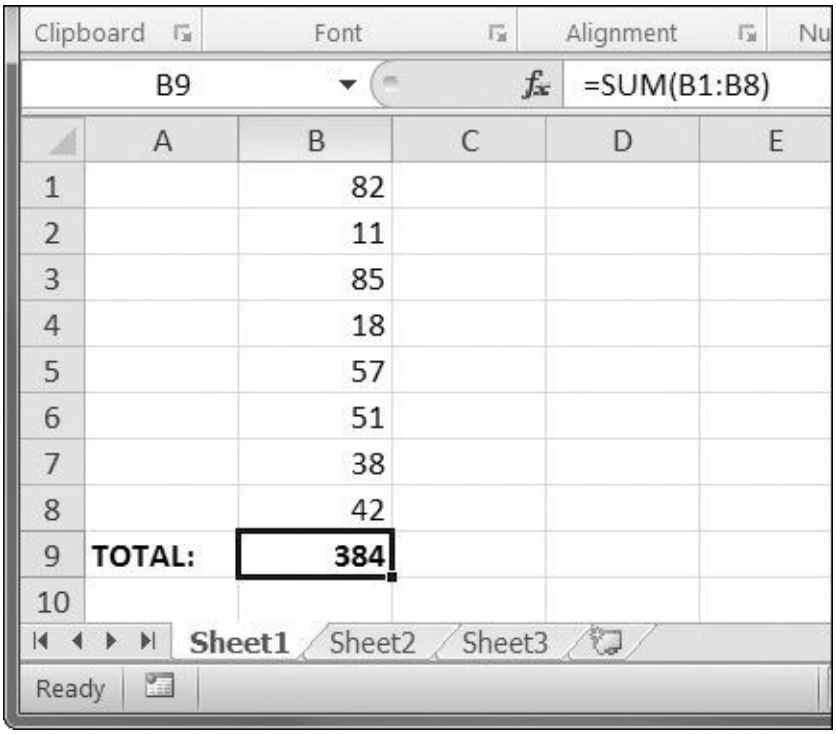


Figura 12-5. La celda B9 contiene la fórmula =SUMA(B1:B8), que suma las celdas B1 a B8.

Una fórmula se establece como cualquier otro valor de texto en una celda. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
>>> wb= openpyxl.Workbook()
>>> hoja= wb.GET_ACTIVE_SHEET()
>>> hoja['A1']= 200
>>> hoja['A2']= 300
>>> hoja['A3']= '=SUMA(A1:A2)'
>>> wb.save('escribirFormula.XLSX')
```

Las celdas A1 y A2 se ajustan a 200 y 300, respectivamente. El valor de la celda A3 establece con una fórmula que suma los valores de A1 y A2. Cuando la hoja de cálculo se abre en Excel, A3 mostrará su valor como 500.

También puede leer la fórmula en una celda como lo haría con cualquier valor. Sin embargo, si desea ver el resultado del *cálculo* de la fórmula en lugar de la fórmula literal, debe pasar True para el argumento de la palabra clave data_only a load_workbook(). Esto significa que un objeto Workbook puede mostrar las fórmulas o el resultado de las fórmulas, pero no ambos. (Pero puedes tener múltiples objetos Workbook cargados para el mismo archivo de hoja de cálculo). Introduce lo siguiente en el shell interactivo para ver la diferencia entre cargar un libro de trabajo con y sin el argumento de la palabra clave data_only:

```
>>> importar openpyxl
>>> wbFormulas = openpyxl.load_workbook('escribirFormula.xlsx')
>>> hoja= wbFormulas.get_active_sheet()
>>> hoja['A3'].valor
'=SUM(A1:A2)'

>>> wbDataOnly= openpyxl.load_workbook('writeFormula.xlsx', data_only=True)
>>> hoja= wbDataOnly.get_active_sheet()
>>> hoja['A3'].valor
500
```

Aquí, cuando se llama a load_workbook() con data_only=True, la celda A3 muestra 500, el resultado de la fórmula *=SUMA(A1:A2)*, en lugar del texto de la fórmula.

Las fórmulas de Excel ofrecen un nivel de programabilidad para las hojas de cálculo, pero pueden volverse rápidamente inmanejables para tareas complicadas. Por ejemplo, aunque estés profundamente familiarizado con las fórmulas de Excel, es un quebradero de cabeza intentar descifrar qué *=IFERROR(TRIM(IF(LEN(VLOOKUP(F7, Hoja2!$A$1:$B$10000, 2, FALSE))>0,SUBSTITUTE(VLOOKUP(F7, Hoja2!$A$1:$B$10000, 2, FALSE), " ", ""),"")), "")* realmente lo hace. El código Python es mucho más legible.

Ajuste de filas y columnas

En Excel, ajustar el tamaño de filas y columnas es tan fácil como hacer clic y arrastrar los bordes de la cabecera de una fila o columna. Pero si necesitas ajustar el tamaño de una fila o columna basándote en el contenido de sus celdas o si quieres ajustar tamaños en un gran número de archivos de hoja de cálculo, será mucho más rápido escribir un programa Python para hacerlo.

Las filas y columnas también pueden ocultarse por completo. O pueden "congelarse" para que siempre estén visibles en la pantalla y aparezcan en todas las páginas cuando se imprima la hoja de cálculo (lo que resulta útil para los encabezados).

Ajustar el alto de fila y el ancho de columna

Los objetos de hoja de cálculo tienen atributos row_dimensions y column_dimensions que controlan la altura de las filas y la anchura de las columnas. Introduzca esto en el intérprete de comandos interactivo:

```
>>> importar openpyxl
>>> wb= openpyxl.Workbook()
>>> hoja= wb.GET_ACTIVE_SHEET()
>>> hoja['A1']= 'Fila alta'
>>> hoja['B2']=  "Columna ancha'
>>> sheet.ROW_DIMENSIONS[1].height= 70
>>> sheet.COLUMN_DIMENSIONS['B'].width= 20
>>> wb.save('DIMENSIONES.XLSX')
```

row_dimensions y column_dimensions de una hoja son valores de tipo diccionario; row_dimensions contiene objetos RowDimension y column_dimensions contiene objetos ColumnDimension. En row_dimensions, puede acceder a uno de los objetos utilizando el número de la fila (en este caso, 1 o 2). En column_dimensions, puede acceder a uno de los objetos utilizando la letra de la columna (en este caso, A o B).

La hoja de cálculo *dimensiones.xlsx* tiene el aspecto de la Figura 12-6.

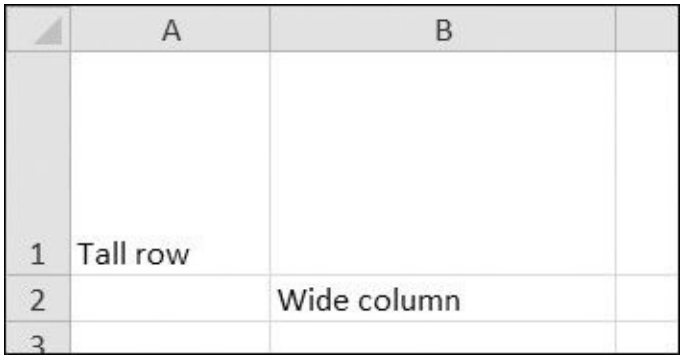


Figura 12-6. Fila 1 y columna B ajustadas a alturas y anchuras mayores

Una vez que tenga el objeto RowDimension, puede establecer su altura. Una vez que tenga el objeto ColumnDimension, puede establecer su anchura. La altura de la fila puede establecerse en un valor entero o flotante entre 0 y 409. Este valor representa la altura medida en puntos, donde un punto equivale a 1/72. Este valor representa la altura medida en *puntos*, donde un punto equivale a 1/72 de pulgada. El alto de fila por defecto es 12.75. La anchura de columna puede establecerse en un valor entero o flotante entre 0 y 255. Este valor representa el número de caracteres en la columna. Este valor representa el número de caracteres con el tamaño de fuente por defecto (11 puntos) que pueden mostrarse en la celda. El ancho de columna por defecto es 8.43

caracteres. Las columnas con anchura 0 o las filas con altura 0 se ocultan al usuario.

Fusión y separación de celdas

Un área rectangular de celdas puede fusionarse en una sola celda con la tecla método de hoja merge_cells(). Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
>>> wb= openpyxl.Workbook()
>>> hoja= wb.GET_ACTIVE_SHEET()
>>> hoja.COMBINAR_CELDAS('A1:D3')
>>> hoja['A1']= 'Doce CELDAS '.
>>> hoja.COMBINAR_CELDAS('C5:D5')
>>> sheet['C5']= 'Dos CELDAS combinadas.'
>>> wb.save('fusionado.XLSX')
```

El argumento de merge_cells() es una sola cadena de las celdas superior izquierda e inferior derecha del área rectangular a fusionar: 'A1:D3' fusiona 12 celdas en una sola celda. Para establecer el valor de estas celdas fusionadas, basta con establecer el valor de la celda superior izquierda del grupo fusionado.

Cuando ejecute este código, *merged.xlsx* tendrá el aspecto de la Figura 12-7.

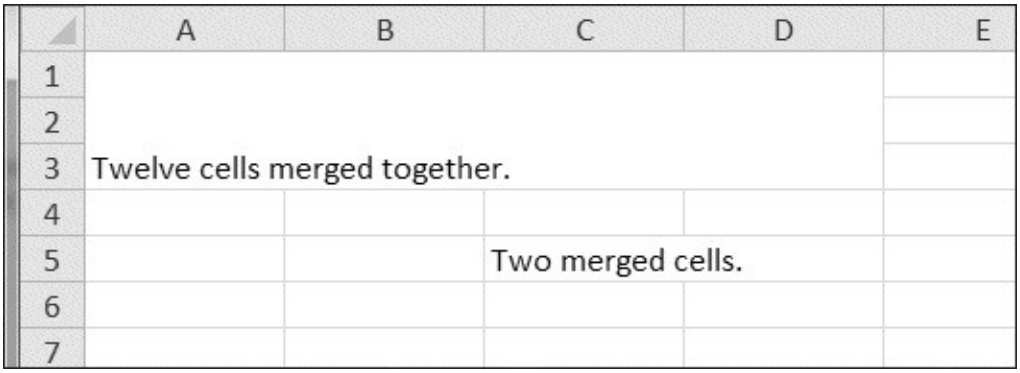


Figura 12-7. Celdas combinadas en una hoja de cálculo

Para descompilar celdas, llame al método de hoja unmerge_cells(). Introduzca esto en el intérprete de comandos interactivo.

```
>>> importar openpyxl
>>> wb= openpyxl.load_workbook('fusionado.XLSX')
>>> hoja= wb.GET_ACTIVE_SHEET()
>>> sheet.UNMERGE_CELLS('A1:D3')
>>> hoja.UNMERGE_CELLS('C5:D5')
>>> wb.save('fusionado.XLSX')
```

Si guardas los cambios y luego echas un vistazo a la hoja de cálculo, verás que las celdas combinadas han vuelto a ser celdas individuales.

Paneles de congelación

En el caso de las hojas de cálculo demasiado grandes para mostrarlas todas a la resulta útil "congelar" en pantalla algunas de las filas superiores o de las columnas situadas más a la izquierda. Las cabeceras de columna o congeladas, por ejemplo, son siempre visibles para el usuario aunque se desplace por la hoja de cálculo. Estos se conocen como *paneles congelados*. En OpenPyXL, cada objeto Hoja de Cálculo tiene un atributo freeze_panes que puede ser establecido a un objeto Celda o una cadena de coordenadas de una celda. Tenga en cuenta que todas las filas por encima y todas las columnas a la izquierda de esta celda se , pero la fila y la columna de la celda en sí no se congelarán.

Para descongelar todos los paneles, ajuste freeze_panes a None o 'A1'. La Tabla 12-3 muestra qué filas y columnas se congelarán para algunos ejemplos de configuración de freeze_panes.

Tabla 12-3. Ejemplos de paneles congelados

configuración de freeze_panes	**Filas y columnas congeladas**
hoja.congelar_paneles= 'A2'	Fila 1
hoja.congelar_paneles= 'B1'	Columna A
hoja.congelar_paneles= 'C1'	Columnas A y B
hoja.congelar_paneles= 'C2'	Fila 1 y columnas A y B
sheet.freeze_panes= 'A1' or sheet.freeze_panes= None	Sin cristales congelados

Asegúrese de que dispone de la hoja de cálculo de ventas de productos de http://nostarch.com/automatestuff/. A continuación, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar openpyxl
>>> wb= openpyxl.load_workbook('PRODUCIRVENTAS.XLSX')
>>> hoja= wb.GET_ACTIVE_SHEET()
>>> hoja.CONGELAR_PANELES= 'A2'
>>> wb.save('congelarEjemplo.XLSX')
```

Si establece el atributo freeze_panes en 'A2', la fila 1 siempre estará visible, independientemente de por dónde se desplace el usuario en la hoja de cálculo. Esto puede verse en Figura 12-8.

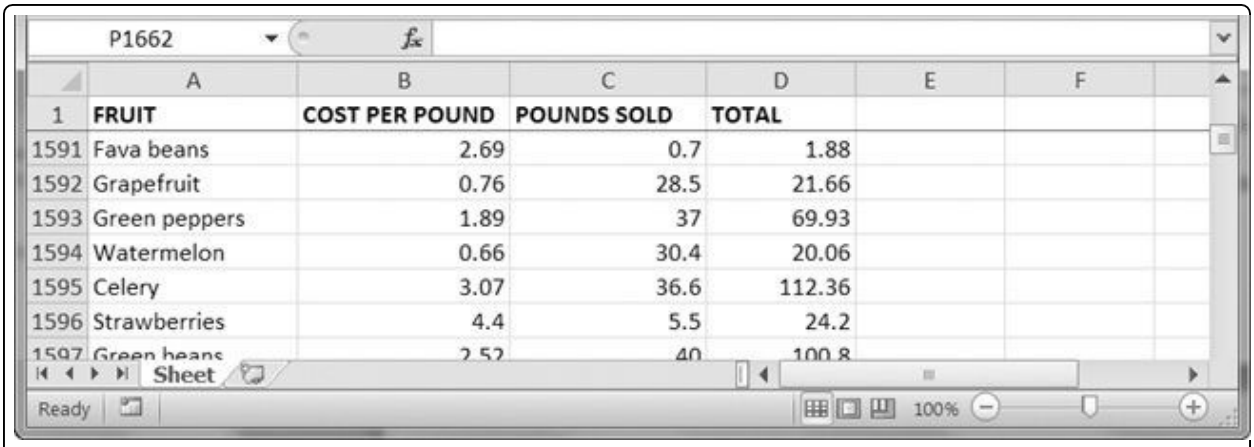


Figura 12-8. Con freeze_panes *establecido en* 'A2'*, la fila 1 siempre está visible incluso cuando el usuario se desplaza hacia abajo.*

Gráficos

OpenPyXL permite crear gráficos de barras, líneas, dispersión y circulares utilizando los datos de las celdas de una hoja. Para hacer un gráfico, tienes que hacer lo siguiente:

1. Crear un objeto Referencia a partir de una selección rectangular de celdas.
2. Crea un objeto Serie pasando el objeto Referencia.
3. Crear un objeto Gráfico.
4. Añade el objeto Serie al objeto Gráfico.
5. Opcionalmente, establezca los valores de dibujo.arriba, dibujo.izquierda, dibujo.anchura y
   variables drawing.height del objeto Chart.
6. Añada el objeto Gráfico al objeto Hoja de cálculo.

El objeto Reference requiere algunas explicaciones. Los objetos Reference se crean llamando a la función openpyxl.charts.Reference() y pasándole tres argumentos:

1. El objeto Hoja de cálculo que contiene los datos del gráfico.
2. Una tupla de dos enteros, que representa la celda superior izquierda de la selección rectangular de celdas que contienen los datos del gráfico: El primer entero de la tupla es la fila y el segundo es la columna. Tenga en cuenta que 1 es la primera fila, no 0.
3. Una tupla de dos enteros, que representa la celda inferior derecha de la selección rectangular de celdas que contienen los datos del gráfico: El primer entero de la tupla es la fila y el segundo es la columna.

La Figura 12-9 muestra algunos ejemplos de argumentos de coordenadas.

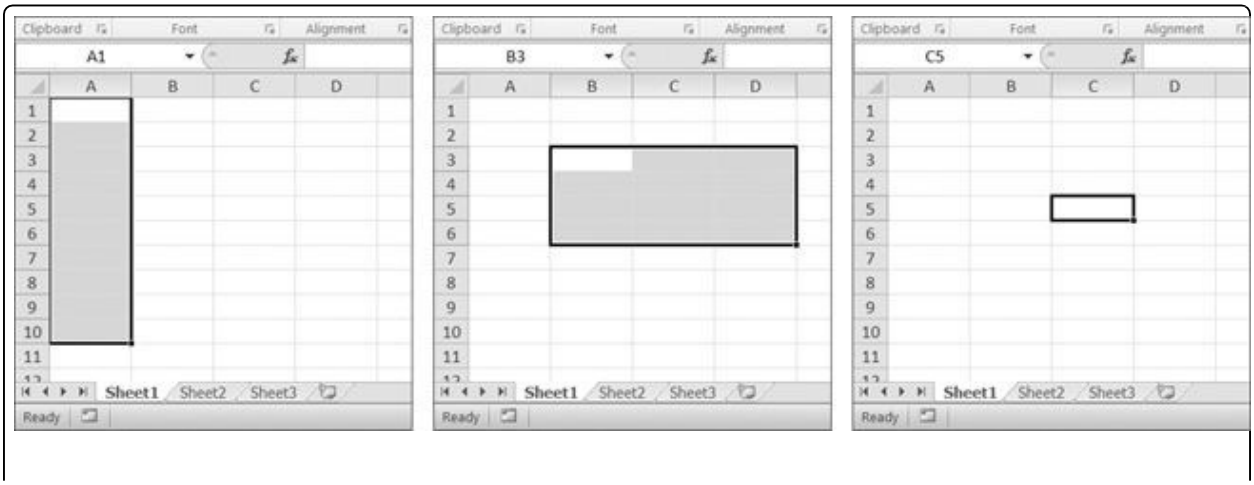


Figura 12-9. De izquierda a derecha: (1, 1), (10, 1)*;* (3, 2), (6, 4)*;* (5, 3), (5, 3)

Introduzca este ejemplo de shell interactivo para crear un gráfico de barras y añadirlo a la hoja de cálculo:

```
>>> importar openpyxl
>>> wb= openpyxl.Workbook()
>>> hoja= wb.GET_ACTIVE_SHEET()
>>> for i in range(1, 11):                # crear algunos datos en la
        columna A hoja['A'+ str(i)]= i

>>> refObj= openpyxl.CHARTS.Reference(hoja, (1, 1), (10, 1))

>>> seriesObj= openpyxl.CHARTS.SERIES(refObj, title='Primera SERIE')

>>> chartObj= openpyxl.CHARTS.BarChart()
>>> chartObj.append(SERIESOBJ)
>>> chartObj.drawing.top= 50              # fijar la posición
>>> chartObj.drawing.left= 100
>>> chartObj.drawing.width= 300           # establecer el tamaño
>>> chartObj.drawing.height= 200

>>> sheet.add_chart(chartObj)
>>> wb.save('ejemploGráfico.XLSX')
```

Esto produce una hoja de cálculo parecida a la Figura 12-10.

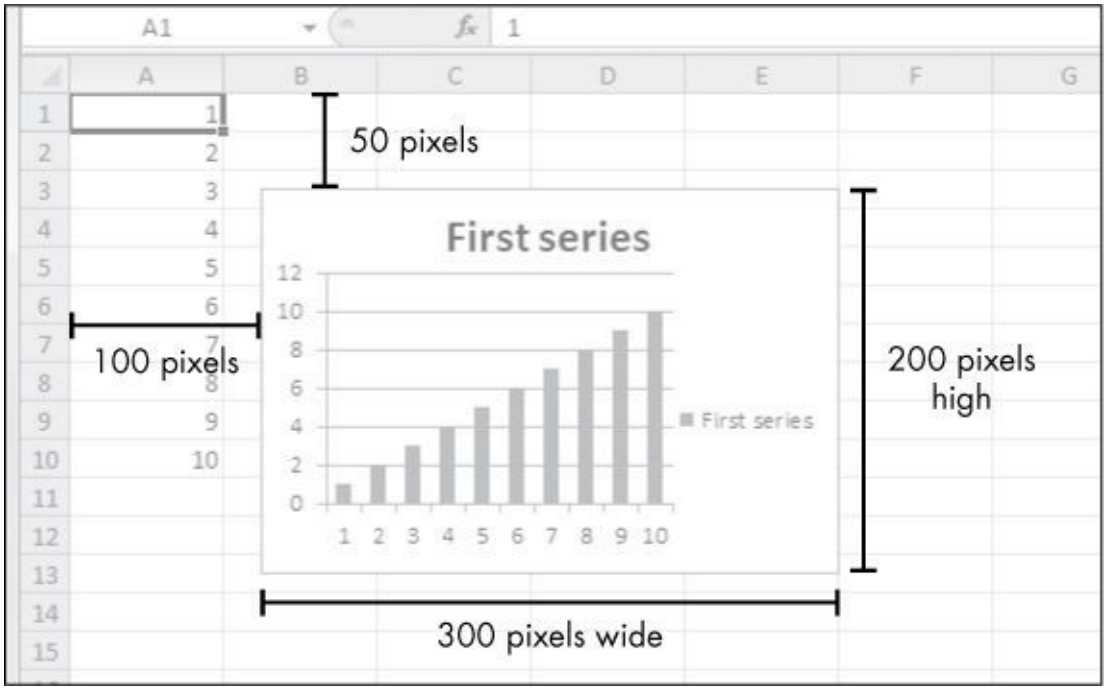


Figura 12-10. Una hoja de cálculo con un gráfico añadido

Hemos creado un gráfico de barras llamando a openpyxl.charts.BarChart(). También puede crear gráficos de líneas, de dispersión y de tarta llamando a openpyxl.charts.LineChart(), openpyxl.charts.ScatterChart() y openpyxl.charts.PieChart().

Desafortunadamente, en la versión actual de OpenPyXL (2.1.4), la función load_workbook() no carga gráficos en archivos Excel. Incluso si el archivo de Excel tiene gráficos, el objeto Workbook cargado no los incluirá. Si carga un objeto Workbook e inmediatamente lo guarda con el mismo nombre de archivo *.xlsx*, efectivamente eliminará los gráficos del mismo.

Resumen

A menudo, lo difícil de procesar información no es el procesamiento en sí, sino simplemente conseguir los datos en el formato adecuado para tu programa. Pero una vez que tienes tu hoja de cálculo cargada en Python, puedes extraer y manipular sus datos mucho más rápido de lo que podrías hacerlo a mano.

También puedes generar hojas de cálculo como salida de tus programas. Así, si los compañeros necesitan que su archivo de texto o PDF con miles de contactos de ventas se transfiera a un archivo de hoja de cálculono tendrán que copiarlo y pegarlo todo tediosamente en Excel.

Equipado con el módulo openpyxl y algunos conocimientos de programación, procesar incluso las hojas de cálculo más grandes será pan comido.

Preguntas prácticas

Para las siguientes preguntas, imagine que tiene un objeto Workbook en la variable wb, un objeto Worksheet en sheet, un objeto Cell en cell, un objeto Comment en comm y un objeto Image en img.

P: 1. ¿Qué devuelve la función openpyxl.load_workbook()?

P: 2. ¿Qué devuelve el método get_sheet_names() workbook?

P: 3. ¿Cómo recuperaría el objeto Hoja de cálculo para una hoja llamada 'Hoja1'?

P: 4. ¿Cómo recuperaría el objeto Worksheet para la hoja activa del libro de trabajo?

P: 5. ¿Cómo recuperarías el valor de la celda C5?

P: 6. ¿Cómo establecerías el valor de la celda C5 en "Hola"?

P: 7. ¿Cómo recuperarías la fila y la columna de la celda como números enteros?

P: 8. ¿Qué devuelven los métodos de hoja get_highest_column() y get_highest_row(), y cuál es el tipo de datos de estos valores de retorno?

P: 9. Si necesitara obtener el índice entero de la columna "M", ¿a qué función tendría que llamar?

P: 10. Si necesitara obtener el nombre de la cadena de la columna 14, ¿a qué función tendría que llamar?

P: 11. ¿Cómo se puede recuperar una tupla de todos los objetos Celda de A1 a F1?

P: 12. ¿Cómo guardaría el libro de trabajo con el nombre de archivo *ejemplo.xlsx*?

P: 13. ¿Cómo se establece una fórmula en una celda?

P: 14. Si quieres recuperar el resultado de la fórmula de una celda en lugar de la propia fórmula de la celda, ¿qué debes hacer primero?

P: 15. ¿Cómo fijaría la altura de la fila 5 en 100?

P: 16. ¿Cómo ocultarías la columna C?

P: 17. Nombre algunas características que OpenPyXL 2.1.4 no carga desde un archivo de hoja de cálculo.

Q: 18. ¿Qué es un panel de congelación?

P: 19. ¿Qué cinco funciones y métodos hay que llamar para crear un gráfico de barras?

Proyectos prácticos

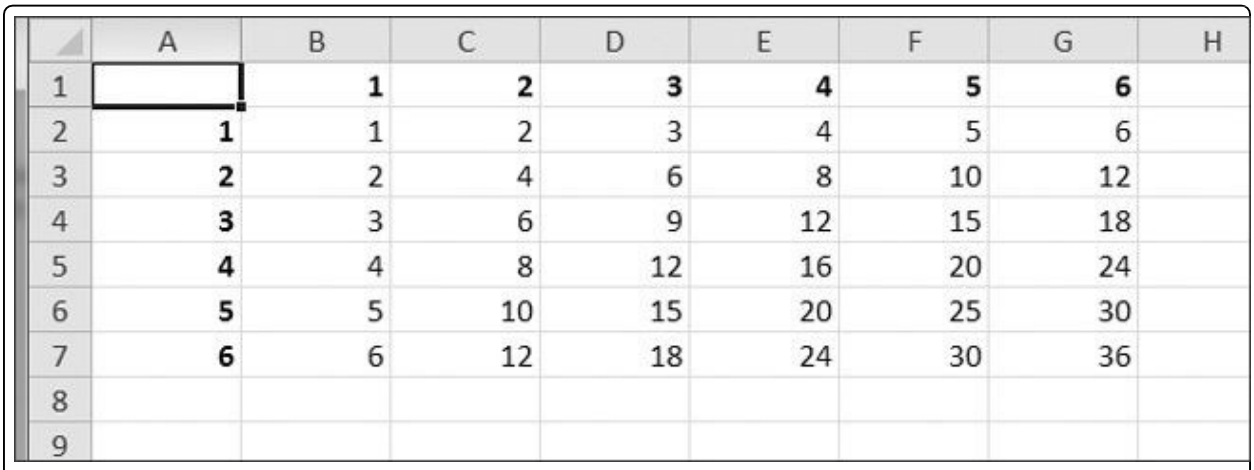
Para practicar, escribe programas que realicen las siguientes tareas.

Creador de tablas de multiplicar

Crear un programa *multiplicationTable.py* que toma un número *N* de la línea de comandos y crea una tabla de multiplicación *N*× N en una hoja de cálculo Excel. Por ejemplo, cuando el programa se ejecuta así:

```
py tablaMultiplicacion.py 6
```

... debería crear una hoja de cálculo parecida a la Figura 12-11.



	A	B	C	D	E	F	G	H
1		**1**	**2**	**3**	**4**	**5**	**6**	
2	**1**	1	2	3	4	5	6	
3	**2**	2	4	6	8	10	12	
4	**3**	3	6	9	12	15	18	
5	**4**	4	8	12	16	20	24	
6	**5**	5	10	15	20	25	30	
7	**6**	6	12	18	24	30	36	
8								
9								

Figura 12-11. Tabla de multiplicar generada en una hoja de cálculo

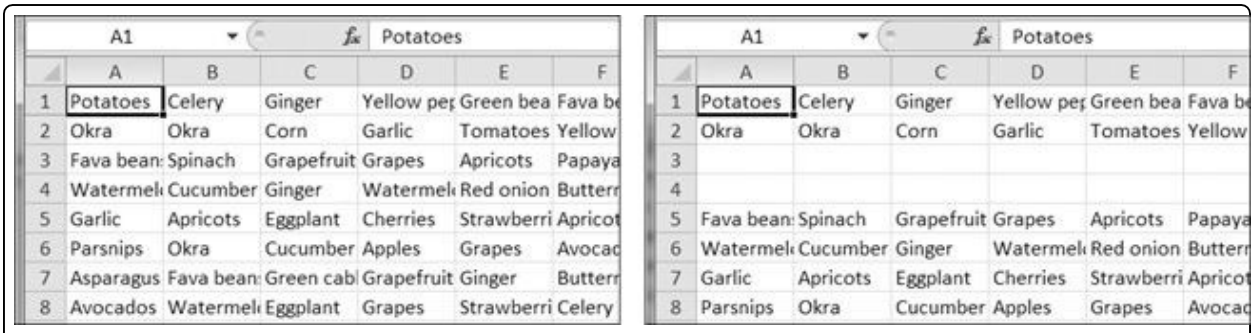
La fila 1 y la columna A deben utilizarse para las etiquetas y deben estar en negrita.

Insertador de filas en blanco

Crea un programa *blankRowInserter.py* que tome dos enteros y una cadena de nombre de archivo como argumentos en la línea de comandos. Llamemos al primer entero *N* y al segundo entero *M*. Empezando en la fila *N*, el programa debería insertar *M* filas en blanco en la hoja de cálculo. Por ejemplo, cuando el programa se ejecuta así:

```
python blankRowInserter.py 3 2 miProducir.xlsx
```

... las hojas de cálculo "antes" y "después" deberían parecerse a la Figura 12-12.



A1	Potatoes					
	A	B	C	D	E	F
1	Potatoes	Celery	Ginger	Yellow pep	Green bea	Fava be
2	Okra	Okra	Corn	Garlic	Tomatoes	Yellow
3	Fava bean	Spinach	Grapefruit	Grapes	Apricots	Papaya
4	Watermel	Cucumber	Ginger	Watermel	Red onion	Butterr
5	Garlic	Apricots	Eggplant	Cherries	Strawberri	Apricot
6	Parsnips	Okra	Cucumber	Apples	Grapes	Avocac
7	Asparagus	Fava bean	Green cab	Grapefruit	Ginger	Butterr
8	Avocados	Watermel	Eggplant	Grapes	Strawberri	Celery

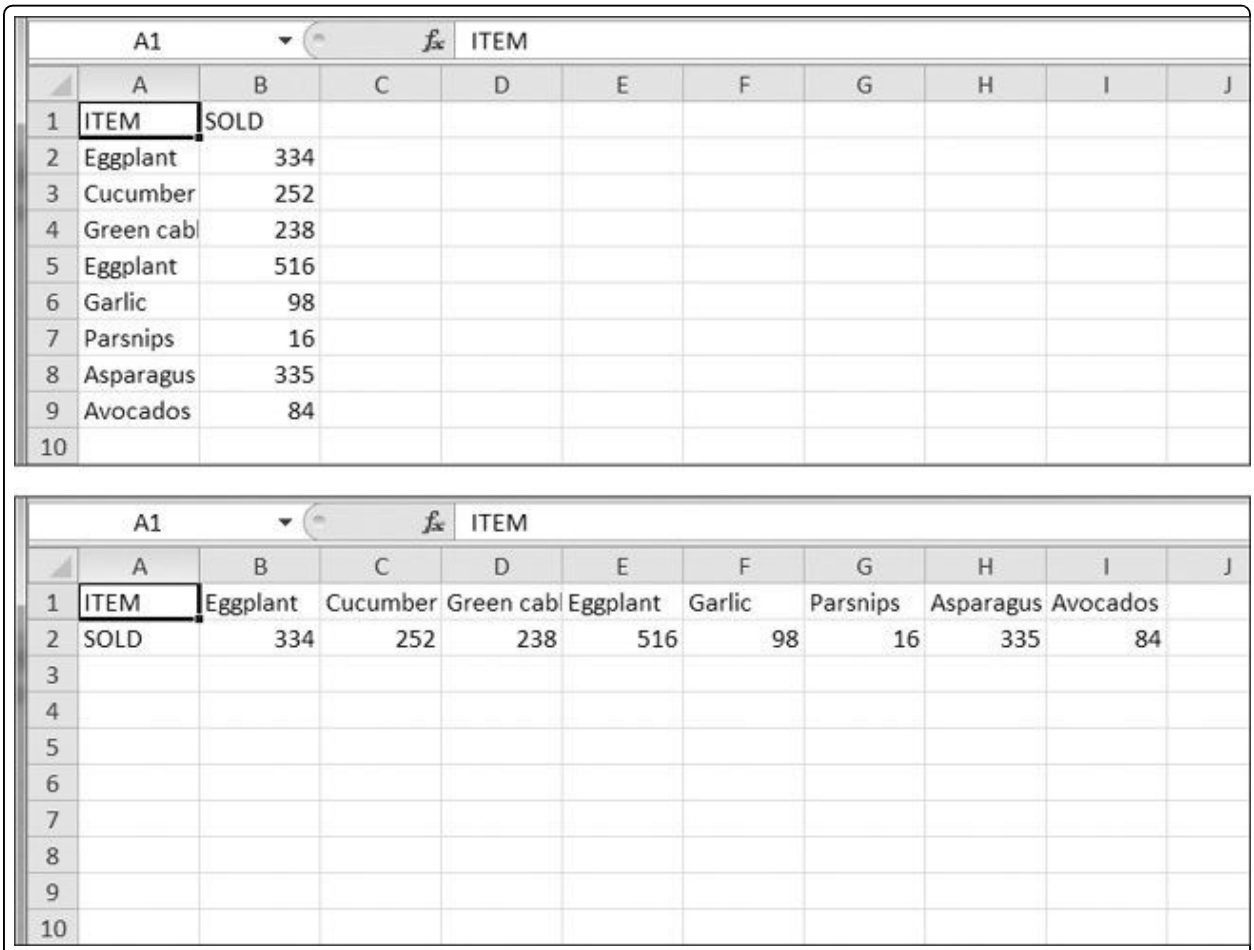
A1	Potatoes					
	A	B	C	D	E	F
1	Potatoes	Celery	Ginger	Yellow pep	Green bea	Fava be
2	Okra	Okra	Corn	Garlic	Tomatoes	Yellow
3						
4						
5	Fava bean	Spinach	Grapefruit	Grapes	Apricots	Papaya
6	Watermel	Cucumber	Ginger	Watermel	Red onion	Butterr
7	Garlic	Apricots	Eggplant	Cherries	Strawberri	Apricot
8	Parsnips	Okra	Cucumber	Apples	Grapes	Avocac

Figura 12-12. Antes (izquierda) y después (derecha) de insertar las dos filas en blanco en la fila 3

Puedes escribir este programa leyendo el contenido de la hoja de cálculo. Luego, al escribir la nueva hoja de cálculo, utiliza un bucle `for` para copiar las *N* primeras líneas. Para las líneas restantes, añade *M* al número de fila en la hoja de cálculo de salida.

Inversor de celdas de hoja de cálculo

Escribe un programa para invertir la fila y la columna de las celdas de la hoja de cálculo. Por ejemplo, el valor de la fila 5, columna 3 estará en la fila 3, columna 5 (y viceversa). Esto debe hacerse para todas las celdas de la hoja de cálculo. Por ejemplo, las hojas de cálculo "antes" y "después" tendrían un aspecto parecido al de la Figura 12-13.



A1 ITEM

	A	B	C	D	E	F	G	H	I	J
1	ITEM	SOLD								
2	Eggplant	334								
3	Cucumber	252								
4	Green cabl	238								
5	Eggplant	516								
6	Garlic	98								
7	Parsnips	16								
8	Asparagus	335								
9	Avocados	84								
10										

A1 ITEM

	A	B	C	D	E	F	G	H	I	J
1	ITEM	Eggplant	Cucumber	Green cabl	Eggplant	Garlic	Parsnips	Asparagus	Avocados	
2	SOLD	334	252	238	516	98	16	335	84	
3										
4										
5										
6										
7										
8										
9										
10										

Figura 12-13. La hoja de cálculo antes (arriba) y después (abajo) de la inversión

Puedes escribir este programa usando bucles `for` anidados para leer los datos de la hoja de cálculo en una estructura de datos de lista de listas. Esta estructura de datos podría tener `sheetData[x][y]` para la celda en la columna `x` y la fila `y`. Entonces, cuando se la nueva hoja de cálculo, se usa `sheetData[y][x]` para la celda en la columna `x` y la fila `y`.

Archivos de texto a hoja de cálculo

Escribe un programa que lea el contenido de varios archivos de texto (puedes crear los archivos tú mismo) y los inserte en una hoja de cálculo, con una línea de texto por fila. Las líneas del primer fichero de texto estarán en las celdas de la columna A, las líneas del segundo fichero de texto estarán en las celdas de la columna B, y así sucesivamente.

Utilice el método `readlines()` Objeto de archivo para devolver una lista de cadenas, una cadena por cada línea del archivo. Para el primer archivo, escriba la primera línea en la columna 1, fila 1. La segunda línea se escribirá en la columna 1, fila 2, y así sucesivamente. El siguiente fichero que se lea con `readlines()` se escribirá en la columna 2, el siguiente fichero en la columna 3, y así sucesivamente.

Hoja de cálculo a archivos de texto

Escribe un programa que realice las tareas del programa anterior en orden inverso: El programa debe abrir una hoja de cálculo y escribir las celdas de la columna A en un fichero de texto, las celdas de la columna B en otro fichero de texto, y así sucesivamente.

Capítulo 13. Trabajar con documentos PDF y Word

Los documentos PDF y Word son archivos binarios, lo que los hace mucho más complejos que los archivos de texto plano. Además de texto, almacenan mucha información sobre fuentes, colores y diseño. Si quiere que sus programas lean o escriban en documentos PDF o Wordtendrá que hacer algo más que simplemente pasar sus nombres de archivo a `open()`.

Afortunadamente, existen módulos de Python que facilitan la interacción con documentos PDF y Word. Este capítulo cubrirá dos de estos módulos: PyPDF2 y Python-Docx.

Documentos PDF

PDF son las siglas de *Portable Document Format (formato de documento portátil)* y utiliza la extensión de archivo *.pdf*. Aunque los PDF admiten muchas funciones, este capítulo se centrará en las dos cosas que hará más a menudo con ellos: leer el contenido de texto de los PDF y crear nuevos PDF a partir de documentos existentes.

El módulo que utilizarás para trabajar con PDFs es PyPDF2. Para instalarlo, ejecute `pip install PyPDF2` desde la línea de comandos. El nombre de este módulo distingue entre mayúsculas y minúsculas, así que asegúrate de que la *y* está en minúsculas y todo lo demás en mayúsculas. (Si el módulo se ha instalado correctamente, al ejecutar `import PyPDF2` en el intérprete de comandos interactivo no debería aparecer ningún error.

EL PROBLEMÁTICO FORMATO PDF

Aunque los archivos PDF son excelentes para presentar el texto de forma que sea fácil de imprimir y leer, no son fáciles de convertir en texto plano. Por ello, PyPDF2 puede cometer errores al extraer texto de un PDF e incluso puede ser incapaz de abrir algunos PDF. Desgraciadamente, no hay mucho que se pueda hacer al respecto. PyPDF2 puede simplemente ser incapaz de trabajar con algunos de sus archivos PDF en particular. Dicho esto, hasta ahora no he encontrado ningún archivo PDF que no se pueda abrir con PyPDF2.

Extraer texto de PDF

PyPDF2 no tiene una forma de extraer imágenes, gráficos u otros medios de documentos PDF, pero puede extraer texto y devolverlo como una cadena Python. Para empezar a aprender cómo funciona PyPDF2lo usaremos en el PDF de ejemplo que se muestra en Figura 13-1.

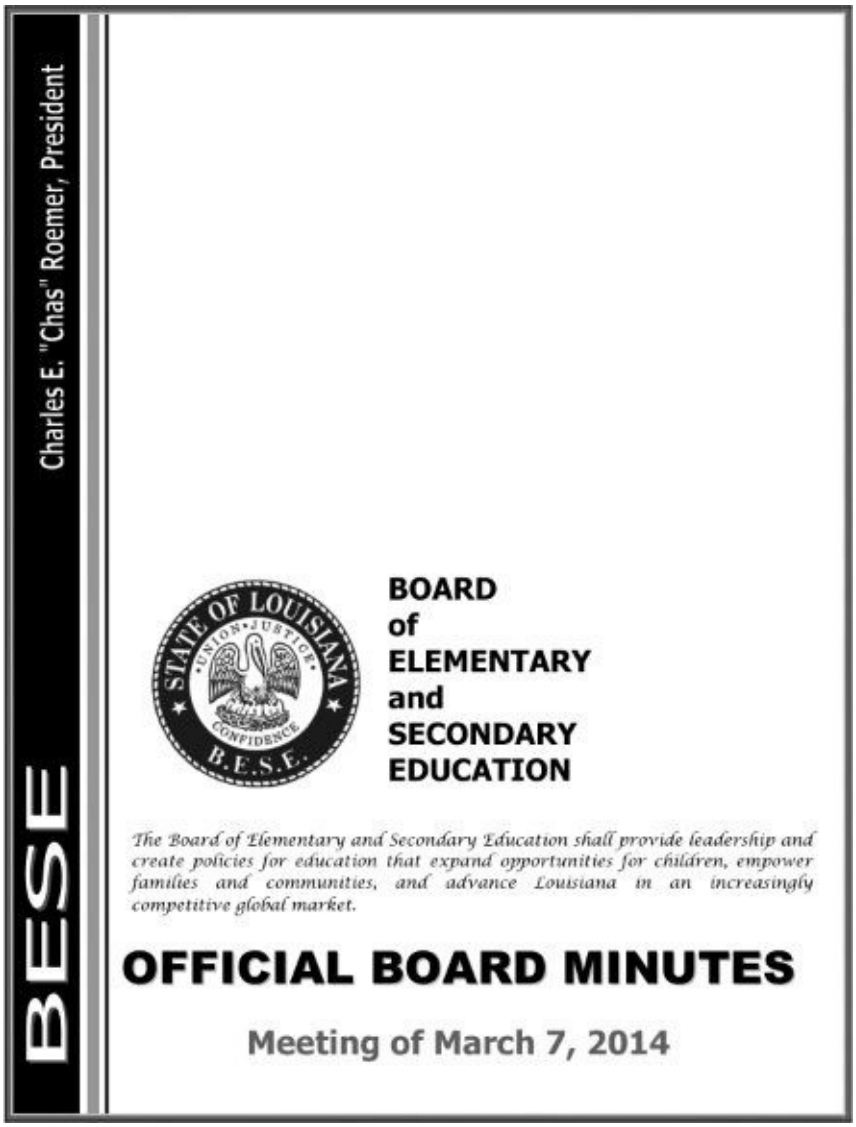


Figura 13-1. La página PDF de la que extraeremos texto

Descargue este PDF de http://nostarch.com/automatestuff/, e introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar PyPDF2
    >>> pdfFileObj= open('ACTASREUNION.pdf', 'rb')
```

```
>>> pdfReader= PyPDF2.PdfFileReader(pdfFileObj)
>>> ❶pdfReader.numPages 19
>>> ❷pageObj= pdfReader.getPage(0)
➲  >>> pageObj.extractText()
'OOFFFFIICCIIAALL BBOOAARRDD MMIINNUUTTEESS Reunión del 7 de marzo de 2015
\n        El Consejo de Educación Primaria y Secundaria proporcionará liderazgo y creará
políticas para la educación que amplíen las oportunidades para los niños, capaciten
a las familias y comunidades, y hagan avanzar a Luisiana en un mercado global cada vez
más competitivo. JUNTA DE EDUCACIÓN ELEMENTAL y SECUNDARIA '
```

Primero, importa el módulo PyPDF2. A continuación, *abre meetingminutes.pdf* en modo binario de lectura y guárdalo en pdfFileObj. Para obtener un objeto PdfFileReader que represente este PDF, llama a PyPDF2.PdfFileReader() y pásale pdfFileObj. Guarde este objeto PdfFileReader en pdfReader.

El número total de páginas del documento se almacena en el atributo numPages de un objeto PdfFileReader❶. El PDF de ejemplo tiene 19 páginas, pero vamos a extraer texto sólo de la primera página.

Para extraer texto de una página, necesita obtener un objeto Page, que representa una sola página de un PDF, de un objeto PdfFileReader. Puede obtener un objeto Page llamando al método getPage() ❷ de un objeto PdfFileReader y el número de página que le interesa, en nuestro caso, 0.

PyPDF2 utiliza un *índice basado en cero* para obtener las páginas: La primera página es la página 0, la segunda es Introducción, y así sucesivamente. Este es siempre el caso, incluso si las páginas están numeradas de forma diferente dentro del documento. Por ejemplo, digamos que su PDF es un extracto de tres páginas de un informe más largo, y sus páginas están numeradas 42, 43, y
44. Para obtener la primera página de este documento, debería llamar a pdfReader.getPage(0), no getPage(42) o getPage(1).

Una vez que tengas tu objeto Page, llama a su método extractText() para que te devuelva una cadena con el texto de la página ❸. La extracción de texto no es perfecta: El texto *Charles E. "Chas" Roemer, President* del PDF no aparece en la cadena devuelta por extractText(), y el espaciado a veces es incorrecto. Aun así, esta aproximación de el contenido de texto del PDF puede ser lo suficientemente bueno para su programa.

Descifrar PDF

Algunos documentos PDF tienen una función de cifrado que impide su lectura hasta que la persona que abre el documento introduce una contraseña. Introduzca lo siguiente en el intérprete de comandos interactivo con el PDF que ha descargado, que ha sido cifrado con la contraseña *rosebud*:

```
>>> importar PyPDF2
    >>> pdfReader= PyPDF2.PdfFileReader(open('encriptado.pdf', 'rb'))

>>> ❶pdfReader.isEncrypted
    Verdadero
    >>> pdfReader.getPage(0)
❷ Traceback (most recent call last):
      File "<pyshell#173>", line 1, in <module>
        pdfReader.getPage()
      --snip--
      File "C:\Python34\lib\site-packages\PyPDF2\pdf.py", line 1173, in getObject raise
        utils.PdfReadError("el fichero no ha sido desencriptado")
    PyPDF2.utils.PdfReadError: el archivo no ha sido descifrado
➲   >>> pdfReader.decrypt('rosebud')
    1
    >>> pageObj= pdfReader.getPage(0)
```

Todos los objetos PdfFileReader tienen un atributo isEncrypted que es True si el PDF está cifrado y False si no lo está❶. Cualquier intento de llamar a una función que lea el archivo antes de que haya sido descifrado con la contraseña correcta provocará un error ❷.

Para leer un PDF cifrado, llame a la decrypt() y pase la contraseña como una cadena ❸. Después de llamar a decrypt() con la contraseña correcta, verá que llamar a getPage() ya no provoca un error. Si se le da una contraseña incorrecta, la función decrypt() devolverá 0 y getPage() seguirá fallando. Nota que el método decrypt() sólo descifra el objeto PdfFileReader, no el archivo PDF real. Una vez finalizado el programa, el archivo permanece cifrado en el disco duro. Su programa tendrá que volver a llamar a decrypt() la próxima vez que se ejecute.

Creación de PDF

La contrapartida de PyPDF2 a los objetos PdfFileReader son los objetos PdfFileWriter, que pueden crear nuevos archivos PDF. Pero PyPDF2 no puede escribir texto arbitrario en un PDF como Python puede hacer con archivos de texto plano. En cambio, las capacidades de PyPDF2 para escribir PDFs se limitan a copiar páginas de otros PDFs, rotar páginas, superponer páginas y encriptar archivos.

PyPDF2 no permite editar directamente un PDF. En su lugar, tienes que crear un nuevo PDF y luego copiar el contenido de un documento existente. Los ejemplos de esta sección seguirán este enfoque general:

1. Abrir uno o varios PDF existentes (los PDF de origen) en PdfFileReader objetos.
2. Crea un nuevo objeto PdfFileWriter.
3. Copiar páginas de los objetos PdfFileReader en el PdfFileWriter objeto.
4. Por último, utilice el objeto PdfFileWriter para escribir el PDF de salida.

La creación de un objeto PdfFileWriter sólo crea un valor que representa un documento PDF en Python. No crea el archivo PDF real. Para ello, debe llamar al método write() del PdfFileWriter.

El método write() toma un objeto File normal que ha sido abierto en modo *binario de escritura*. Puede obtener un objeto File de este tipo llamando a la función open() de Python con dos argumentos: la cadena con el nombre del archivo PDF y 'wb' para indicar que el archivo debe abrirse en binario de escritura.

Si esto te parece un poco confuso, no te preocupes: verás cómo funciona en los siguientes ejemplos de código.

Copiar páginas

Puede utilizar PyPDF2 para copiar páginas de un documento PDF a otro. Esto le permite combinar varios archivos PDF, cortar páginas no deseadas o reordenar páginas.

Descargue *meetingminutes.pdf* y *meetingminutes2.pdf* de *http://nostarch.com/automatestuff/* y coloque los PDF en el directorio de trabajo actual. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar PyPDF2
    >>> pdf1File= open('ACTASREUNION.pdf', 'rb')
```

```
    >>> pdf2File= open('ACTASREUNION2.pdf', 'rb')
>>> ❶pdf1Reader= PyPDF2.PdfFileReader(pdf1File)
>>> ❷pdf2Reader= PyPDF2.PdfFileReader(pdf2File)
➲   >>> pdfWriter= PyPDF2.PdfFileWriter()

    >>> para pageNum en range(pdf1Reader.NUMPAGES):
❹           pageObj= pdf1Reader.getPage(pageNum)
❺           pdfWriter.addPage(pageObj)

    >>> para pageNum en range(pdf2Reader.NUMPAGES):
❻           pageObj= pdf2Reader.getPage(pageNum)
❼           pdfWriter.addPage(pageObj)

>>> ❽pdfOutputFile= open('COMBINEDMINUTES.pdf', 'wb')
    >>> pdfWriter.write(pdfArchivoSalida)
    >>> pdfOutputFile.close()
    >>> pdf1File.close()
    >>> pdf2File.close()
```

Abra ambos archivos PDF en modo binario de lectura y almacene los dos objetos File resultantes en pdf1File y pdf2File. Llama a PyPDF2.PdfFileReader() y pásale pdf1File para obtener un objeto PdfFileReader para *meetingminutes.pdf*❶. Llámelo de nuevo y pásele pdf2File para obtener un objeto PdfFileReader para *meetingminutes2.pdf* ❷. A continuación, cree un nuevo objeto PdfFileWriter, que representa un documento PDF en blanco ❸.

A continuación, copie todas las páginas de los dos PDF de origen y añádalas al objeto PdfFileWriter. Obtén el objeto Page llamando a getPage() en un objeto PdfFileReader❹. Luego pasa ese objeto Page al método addPage() de tu PdfFileWriter ❺. Estos pasos se hacen primero para pdf1Reader y luego de nuevo para pdf2Reader. Cuando haya terminado de copiar páginas, escriba un nuevo PDF llamado *combinedminutes.pdf* pasando un objeto File al método write() de PdfFileWriter. método❻.

NOTA

PyPDF2 no puede insertar páginas en medio de un objeto PdfFileWriter; el método addPage() sólo añadirá páginas al final.

Ahora ha creado un nuevo archivo PDF que combina las páginas *de meetingminutes.pdf* y *meetingminutes2.pdf* en un único documento. Recuerde que el objeto File pasado a PyPDF2.PdfFileReader() necesita ser abierto en

modo lectura-binario pasando 'rb' como segundo argumento a open(). Del mismo modo, el objeto File pasado a PyPDF2.PdfFileWriter() necesita ser abierto en modo escritura-binario con 'wb'.

Páginas giratorias

Las páginas de un PDF también se pueden girar en incrementos de 90 grados con los métodos rotateClockwise() y rotateCounterClockwise(). Pase uno de enteros 90, 180 o 270 a estos métodos. Introduzca lo siguiente en el intérprete de comandos interactivo, con el archivo *meetingminutes.pdf* en el directorio de trabajo actual:

```
>>> importar PyPDF2
    >>> minutesFile= open('ACTASREUNION.pdf', 'rb')
    >>> pdfReader= PyPDF2.PdfFileReader(MINUTESFILE)
>>> ❶página= pdfReader.getPage(0)
>>> ❷page.ROTATECLOCKWISE(90)
    {'/Contenidos': [IndirectObject(961, 0), IndirectObject(962, 0),
    --snip--
    }
    >>> pdfWriter= PyPDF2.PdfFileWriter()
    >>> pdfWriter.addPage(página)
➲   >>> resultPdfFile= open('rotatedPage.pdf', 'wb')
    >>> pdfWriter.write(resultadoArchivoPdf)
    >>> resultPdfFile.close()
    >>> MINUTESFILE.close()
```

Aquí usamos getPage(0) para seleccionar la primera página del PDF❶, y luego llamamos a rotateClockwise(90) en esa página ❷. Escribimos un nuevo PDF con la página rotada y lo guardamos como *rotatedPage.pdf* ❸.

El PDF resultante tendrá una página, girada 90 grados en el sentido de las agujas del reloj, como en la Figura 13-2. Los valores de retorno de rotateClockwise() y rotateCounterClockwise() contienen mucha información que puede ignorar.

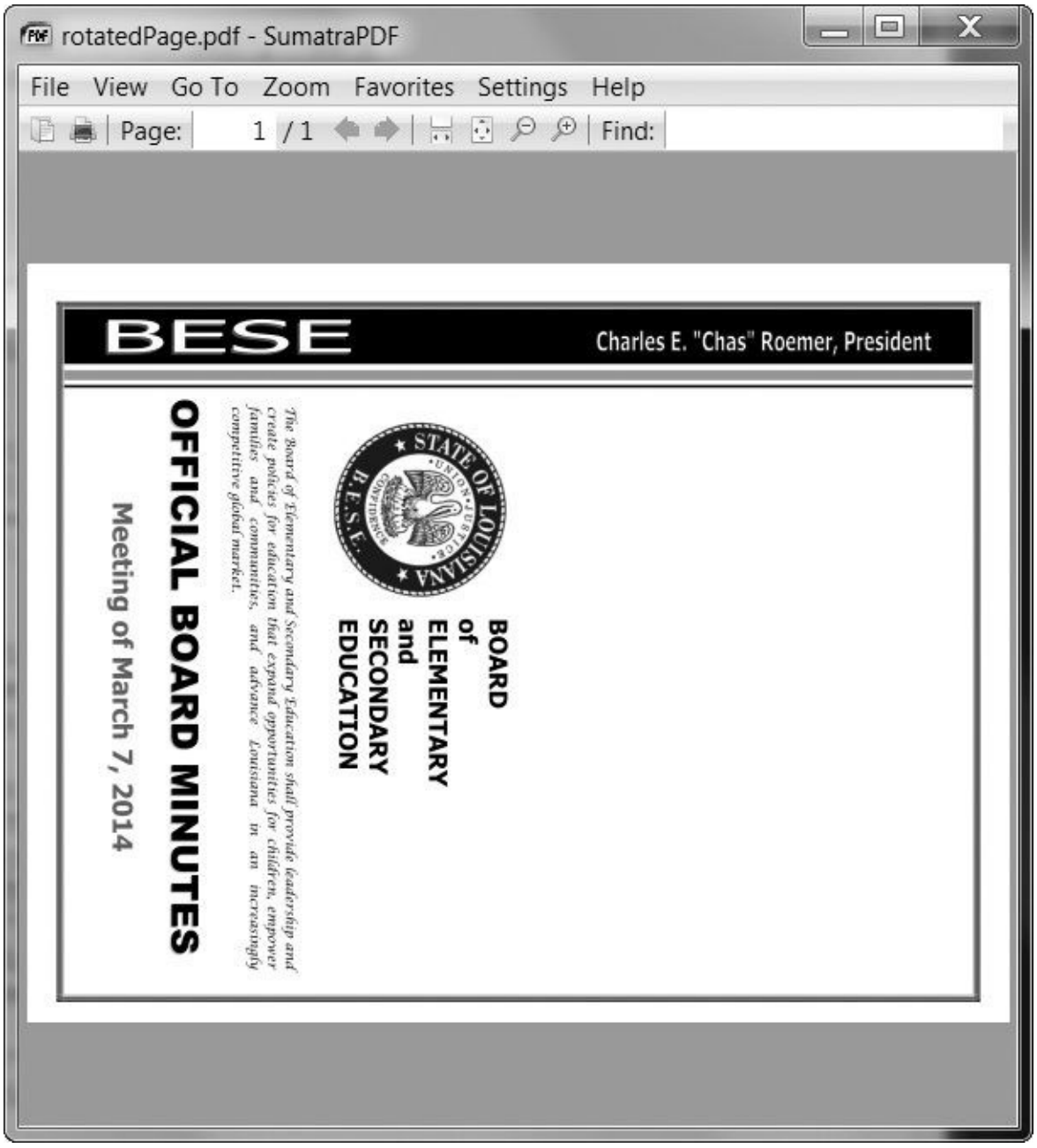


Figura 13-2. El archivo rotatedPage.pdf con la página girada 90 grados en el sentido de las agujas del reloj.

Superposición de páginas

PyPDF2 también puede superponer el contenido de una página sobre otra, lo que resulta útil para añadir un logotipo, una marca de tiempo o una marca de agua a una página. Con Python, es fácil añadir marcas de agua a varios archivos y sólo a las páginas que especifique el programa.

Descargue *watermark.pdf* de *http://nostarch.com/automatestuff/* y coloque el PDF en el directorio de trabajo actual junto con *meetingminutes.pdf*. A continuación, introduzca

lo siguiente en el intérprete de comandos interactivo:

```
>>> importar PyPDF2
    >>> minutesFile= open("actasreunion.pdf", 'rb')
>>> ❷pdfReader= PyPDF2.PdfFileReader(minutesFile)
>>> ❷minutesFirstPage= pdfReader.getPage(0)
➲   >>> pdfWatermarkReader= PyPDF2.PdfFileReader(open("filigrana.pdf", 'rb'))
    >>>     ❹minutesFirstPage.mergePage(pdfWatermarkReader.getPage(0))
>>> ❺pdfWriter= PyPDF2.PdfFileWriter()
    >>>     ❻ pdfWriter.addPage(minutesFirstPage)

>>> ❼for pageNum in range(1, pdfReader.numPages): pageObj=
                pdfReader.getPage(pageNum)
                pdfWriter.addPage(pageObj)
    >>> resultPdfFile= open("watermarkedCover.pdf", 'wb')
    >>>         pdfWriter.write(resultadoArchivoPdf)
    >>> minutesFile.close()
    >>> resultPdfFile.close()
```

Aquí hacemos un objeto PdfFileReader de *meetingminutes.pdf*❶. Llamamos a getPage(0) para obtener un objeto Page para la primera página y almacenamos este objeto en minutesFirstPage❷. A continuación, hacemos un PdfFileReader para *watermark.pdf* ❸ y llamamos a mergePage() en minutesFirstPage❹. El argumento que pasamos a mergePage() es un objeto Page para la primera página de *watermark.pdf*.

Ahora que hemos llamado a mergePage() en minutesFirstPage, minutesFirstPage representa la primera página con marca de agua. Hacemos un objeto PdfFileWriter ❺ y añadimos la primera página con marca de agua❻. Luego hacemos un bucle a través del resto de las páginas en *meetingminutes.pdf* y las añadimos al objeto PdfFileWriter❼. Por último, abrimos un nuevo PDF llamado *watermarkedCover.pdf* y escribimos el contenido del PdfFileWriter en el nuevo PDF.

La figura 13-3 muestra los resultados. Nuestro nuevo PDF, *watermarkedCover.pdf*, tiene todo el contenido de *meetingminutes.pdf*, y la primera página tiene marca de agua.



Figura 13-3. El PDF original (izquierda), el PDF con marca de agua (centro) y el PDF fusionado (derecha)

Cifrar PDF

Un objeto `PdfFileWriter` también puede añadir encriptación a un documento PDF. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar PyPDF2
    >>> pdfFile= open('ACTASREUNION.pdf', 'rb')
    >>> pdfReader= PyPDF2.PdfFileReader(pdfFile)
    >>> pdfWriter= PyPDF2.PdfFileWriter()
    >>> for pageNum in range(pdfReader.NUMPAGES): pdfWriter.addPage(pdfReader.getPage(pageNum))


   >>>    ❶ pdfWriter.encrypt("pez espada")
    >>> resultPdf= open('ACTASENCRIPTADAS.pdf', 'wb')
    >>> pdfWriter.write(resultadoPdf)
    >>> resultPdf.close()
```

Antes de llamar al método `write()` para guardar en un archivo, llame al método `encrypt()` y pásele una cadena de contraseña ❶. Los PDF pueden tener una *contraseña de usuario* (que le ver el PDF) y una *contraseña de propietario* (que le permite establecer permisos para imprimir, comentar, extraer texto y otras funciones). La contraseña de usuario
y owner password son el primer y segundo argumento de `encrypt()`, respectivamente. Si sólo se pasa un argumento de cadena a `encrypt()`, se utilizará para ambas contraseñas.

En este ejemplo, copiamos las páginas *de meetingminutes.pdf* a un objeto `PdfFileWriter`. Encriptamos el PdfFileWriter con la contraseña *swordfish*, abrimos un nuevo PDF llamado *encryptedminutes.pdf*, y escribimos el contenido del objeto

PdfFileWriter al nuevo PDF. Antes de que nadie pueda ver *el archivo meetingminutes.pdf encriptado*, tendrá que introducir esta contraseña. Si lo desea, puede eliminar el archivo *meetingminutes.pdf* original sin cifrar después de asegurarse de que su copia se ha cifrado correctamente.

Proyecto: Combinar páginas seleccionadas de varios PDF

Supongamos que tiene la aburrida tarea de fusionar varias docenas de documentos PDF en un único archivo PDF. Cada uno de ellos tiene una portada como primera página, pero no quiere que la portada se repita en el resultado final. Aunque hay muchos programas gratuitos para combinar PDF, muchos de ellos se limitan a fusionar archivos enteros. Vamos a escribir un programa en Python para personalizar qué páginas quieres en el PDF combinado.

A grandes , esto es lo que hará el programa:

- Buscar todos los archivos PDF en el directorio de trabajo actual.
- Ordena los nombres de archivo para que los PDF se añadan en orden.
- Escriba cada página, excepto la primera, de cada PDF en el archivo de salida. En términos de implementación, su código tendrá que hacer lo siguiente:
- Llame a `os.listdir()` para encontrar todos los archivos en el directorio de trabajo y eliminar cualquier archivo que no sea PDF.
- Llame al método de lista `sort()` de Python para ordenar alfabéticamente los nombres de archivo.
- Crea un objeto `PdfFileWriter` para el PDF de salida.
- Recorre cada archivo PDF, creando un objeto `PdfFileReader` para él.
- Recorra cada página (excepto la primera) de cada archivo PDF.
- Añada las páginas al PDF de salida.
- Escriba el PDF de salida en un archivo *llamado allminutes.pdf*.

Para este proyecto, abre una nueva ventana del editor de archivos y guárdalo como *combinePdfs.py*.

Paso 1: Buscar todos los archivos PDF

Primero, tu programa necesita obtener una lista de todos los archivos con la extensión *.pdf* en el directorio de trabajo actual y ordenarlos. Haz que tu código sea como el siguiente:

```
#! python3
    # combinePdfs.py - Combina todos los PDF del directorio de trabajo actual en # un
    único PDF.

❶ importar PyPDF2, os

    # Obtener todos los nombres de
    archivos PDF. pdfFiles= []
    for filename in os.listdir('.'): if
        filename.endswith(".pdf"):
❷           pdfFiles.append(nombrearchivo)
➲  pdfFiles.sort(clave= str.lower)

❹ pdfWriter= PyPDF2.PdfFileWriter()

    # TODO: Recorrer todos los archivos PDF.

    # TODO: Recorrer todas las páginas (excepto la primera) y .

    # TODO: Guardar el PDF resultante en un archivo.
```

Después de la línea shebang y el comentario descriptivo sobre lo que hace el programa, este código importa los módulos os y PyPDF2❶. La llamada a `os.listdir('.')` devolverá una lista de todos los archivos en el directorio de trabajo actual. El código hace un bucle sobre esta lista y añade sólo aquellos archivos con la extensión *.pdf* a pdfFiles

❷. Después, esta lista se ordena por orden alfabético con la clave= str.lower

argumento de palabra clave para sort()❸.

Se crea un objeto PdfFileWriter para guardar las páginas PDF combinadas ❹. Por último, algunos comentarios resumen el resto del programa.

Paso 2: Abrir cada PDF

Ahora el programa debe leer cada archivo PDF en pdfFiles. Añada lo siguiente a su programa:

```
#! python3
# combinePdfs.py - Combina todos los PDF del directorio de trabajo actual en # un
único PDF.

importar PyPDF2, os

# Obtener todos los nombres de
archivos PDF. pdfFiles= []
--snip--
# Recorrer todos los ARCHIVOS PDF.
for filename in PDFFILES:
    pdfFileObj= open(nombrearchivo, 'rb')
    pdfReader= PyPDF2.PdfFileReader(pdfFileObj)
    # TODO: Recorrer todas las páginas (excepto la primera) y .

# TODO: Guardar el PDF resultante en un archivo.
```

Para cada PDF, el bucle abre un nombre de archivo en modo lectura binaria llamando a open() con 'rb' como segundo argumento. La llamada a open() devuelve un objeto File, que se pasa a PyPDF2.PdfFileReader() para crear un objeto PdfFileReader para ese archivo PDF.

Paso 3: Añadir cada página

Para cada PDF, querrá hacer un bucle sobre cada página excepto la primera. Añada este código a su programa:

```
#! python3
    # combinePdfs.py - Combina todos los PDF del directorio de trabajo actual en # un
    único PDF.

    importar PyPDF2, os

    --snip--

    # Recorrer todos los archivos
    PDF. for filename in pdfFiles:
    --snip--
        # Recorre todas las PÁGINAS (excepto la primera) y .
❶       for pageNum in range(1, pdfReader.NUMPAGES):
            pageObj= pdfReader.getPage(pageNum)
            pdfWriter.addPage(pageObj)

    # TODO: Guardar el PDF resultante en un archivo.
```

El código dentro del bucle `for` copia cada objeto `Page` individualmente al objeto `PdfFileWriter`. Recuerda que quieres saltarte la primera página. Dado que PyPDF2 considera que `0` es la primera página, tu bucle debe comenzar en `1` ❶ y luego ir hasta, pero sin incluir, el entero en `pdfReader.numPages`.

Paso 4: Guardar los resultados

Después de que estos bucles `for` anidados hayan terminado de hacer bucles, la variable `pdfWriter` contendrá un objeto `PdfFileWriter` con las páginas de todos los PDF combinados. El último paso es escribir este contenido en un archivo del disco duro. Añade este código a tu programa:

```
#! python3
# combinePdfs.py - Combina todos los PDF del directorio de trabajo actual en # un
único PDF.
importar PyPDF2, os

--snip--

# Recorrer todos los archivos PDF.
for filename in pdfFiles:
--snip--
    # Recorre todas las páginas (excepto la primera) y . for pageNum in
    range(1, pdfReader.numPages):
    --snip--

# Guardar el PDF resultante en un archivo.
pdfOutput= open('ALLMINUTES.pdf', 'wb')
pdfWriter.write(pdfOutput)
pdfOutput.close()
```

Al pasar `'wb'` a `open()` se abre el archivo PDF de salida, *allminutes.pdf*, en modo binario de escritura. A continuación, pasando el objeto `File` resultante al método `write()` se crea el archivo PDF real. Una llamada al método `close()` finaliza el programa.

Ideas para programas similares

Ser capaz de crear PDF a partir de las páginas de otros PDF te permitirá crear programas que pueden hacer lo siguiente:

- Recorta páginas concretas de los PDF.
- Reordena las páginas de un PDF.
- Crea un PDF sólo a partir de aquellas páginas que tienen algún texto específico, identificado por `extractText()`.

Documentos Word

Python puede crear y modificar documentos de Word, que tienen la extensión de archivo *.docx*, con el python-docx. Puedes instalar el módulo ejecutando `pip install python-docx`. (El Apéndice A tiene todos los detalles sobre la instalación de módulos de terceros).

> **NOTA**
>
> *Cuando use pip para instalar Python-Docx por primera vez, asegúrese de instalar `python-docx`, no `docx`. El nombre de instalación `docx` es para un módulo diferente que este libro no cubre. Sin embargo, cuando vayas a importar el módulo `python-docx`, necesitarás ejecutar `import docx`, no `import python-docx`.*

Si no dispone de Word, LibreOffice Writer y OpenOffice Writer son dos aplicaciones alternativas gratuitas para Windows, OS X y Linux que pueden utilizarse para abrir *.docx*. Puede descargarlos en *https://www.libreoffice.org* y http://openoffice.org, respectivamente. La documentación completa de Python-Docx está disponible en https://python-docx.readthedocs.org/. Aunque existe una versión de Word para OS X, este capítulo se centrará en Word para Windows.

En comparación con el texto plano, los archivos *.docx* tienen mucha estructura. Esta estructura está representada por tres tipos de datos diferentes en Python-Docx. En el nivel más alto, un objeto Documento representa el documento completo. El objeto Document contiene una lista de objetos Paragraph para los párrafos del documento. (Un nuevo párrafo comienza cada vez que el usuario pulsa ENTER o RETURN mientras escribe en un documento Word). Cada uno de estos objetos Párrafo contiene una lista de uno o más objetos Ejecutar. El párrafo de una sola frase de la Figura 13-4 tiene cuatro ejecuciones.

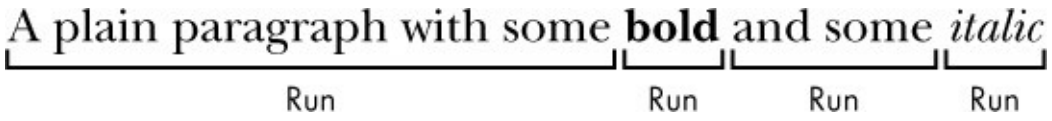


Figura 13-4. Los objetos de ejecución *identificados en un objeto* Párrafo

El texto de un documento Word es algo más que una cadena. Tiene asociada una fuente, un tamaño, un color y otra información de estilo. Un *estilo* en Word es una colección de estos atributos. Un objeto Ejecutar es una serie contigua de texto con el mismo estilo. Cada vez que cambia el estilo del texto, se necesita un nuevo objeto de ejecución.

Lectura de documentos Word

Experimentemos con el módulo python-docx. Descarga *demo.docx* de *http://nostarch.com/automatestuff/* y guarda el documento en el directorio de trabajo. A continuación, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar docx
>>> ❶doc= docx.Document('demo.docx')
>>> ❷len(doc.PÁRRAFOS)
7
➲  >>> doc.PÁRRAFOS[0].texto
Título del documento
>>> ❹doc.PARAGRAPHS[1].text
Un párrafo normal con negrita y cursiva".
>>> ❺len(doc.PÁRRAFOS[1].RECORRIDOS)
4
>>> ❻doc.PARAGRAPHS[1].RUNS[0].text
'Un párrafo sencillo con algunos '
>>> ❼doc.PÁRRAFOS[1].TIRADAS[1].texto
negrita
>>> ❽doc.PÁRRAFOS[1].TIRADAS[2].texto
' y algunos '
>>> ❾doc.PÁRRAFOS[1].TIRADAS[3].texto
cursiva
```

En❶, abrimos un archivo *.docx* en Python, llamamos a docx.Document(), y pasamos el nombre de archivo *demo.docx*. Esto devolverá un objeto Document, que tiene un atributo paragraphs que es una lista de objetos Paragraph. Cuando llamamos a len() en doc.paragraphs, devuelve 7, lo que nos dice que hay siete objetos Paragraph en este documento❷. Cada uno de estos objetos Paragraph tiene un texto que contiene una cadena con el texto de ese párrafo (sin el estilo información). Aquí, el primer atributo de texto contiene 'TítuloDocumento'❸, y el segundo contiene 'Un párrafo sin formato con algo de negrita y algo de cursiva'❹.

Cada objeto Párrafo tiene también un ejecuciones que es una lista de objetos Ejecuciones. Los objetos Run también tienen un atributo text, que contiene sólo el texto de ese run en particular. Veamos los atributos de texto en el segundo objeto Párrafo, 'Un párrafo plano con algo de negrita y algo de cursiva'. Si llamamos a len() en este objeto Párrafo, veremos que hay cuatro objetos

Ejecución❺. El primer objeto Run contiene 'A plain paragraph with some '❻. A continuación, el texto cambia a una negrita

por lo que 'negrita' inicia un nuevo objeto Run❼. El texto vuelve a un estilo sin negrita, lo que da lugar a un tercer objeto Run, ' y algunos '❽. Finalmente, el cuarto y último objeto de ejecución contiene 'cursiva' en un estilo cursiva❾.

Con Python-Docx, tus programas Python ahora podrán leer el texto de un archivo *.docx* y utilizarlo como cualquier otro valor de cadena.

Obtener el texto completo de un archivo .docx

Si sólo le interesa el texto, no la información de estilo, del documento Word, puede utilizar la función getText(). Acepta un nombre de archivo *.docx* y devuelve un único valor de cadena de su texto. Abre una nueva ventana del editor de archivos e introduce el siguiente código, guardándolo como *readDocx.py*:

```
#! python3

import docx

def getText(nombrearchivo):
    doc= docx.Document(nombrearchivo)
    fullText= []
    para para en doc.párrafos: fullText.append(para.texto)
    return '\n'.join(fullText)
```

La función getText() abre el documento Word, recorre todos los objetos Párrafo de la lista de párrafos y añade su texto a la lista en fullText. Después del bucle, las cadenas en fullText se unen con caracteres de nueva línea.

El programa *readDocx.py* se puede importar como cualquier otro módulo. Ahora, si sólo necesita el texto de un documento de Word, puede introducir lo siguiente:

```
>>> importar readDocx
>>> print(readDocx.getText('demo.docx'))
Título del documento
Un párrafo sin formato con negrita y cursiva Título,
nivel 1
Cita intensa
primer elemento de la
lista desordenada primer
elemento de la lista
ordenada
```

También puede ajustar getText() para modificar la cadena antes de devolverla. Por ejemplo, para sangrar cada párrafo, reemplace la llamada append() en *readDocx.py* con esto:

```
fullText.append(' '+ para.text)
```

Para añadir un doble espacio entre párrafos, cambie el código de la llamada a join() por el siguiente:

```
return '\n\n'.join(fullText)
```

Como puedes versólo hacen falta unas pocas líneas de código para escribir funciones que lean un archivo *.docx* y devuelvan una cadena con su contenido a tu gusto.

Estilización de objetos de párrafo y ejecución

En Word para Windows, puede ver los estilos pulsando CTRL-ALT-SHIFT-S para mostrar el panel Estilos, que tiene el aspecto de la Figura 13-5. En OS X, puede ver el panel Estilos haciendo clic en la opción de menú **Ver► Estilos**.

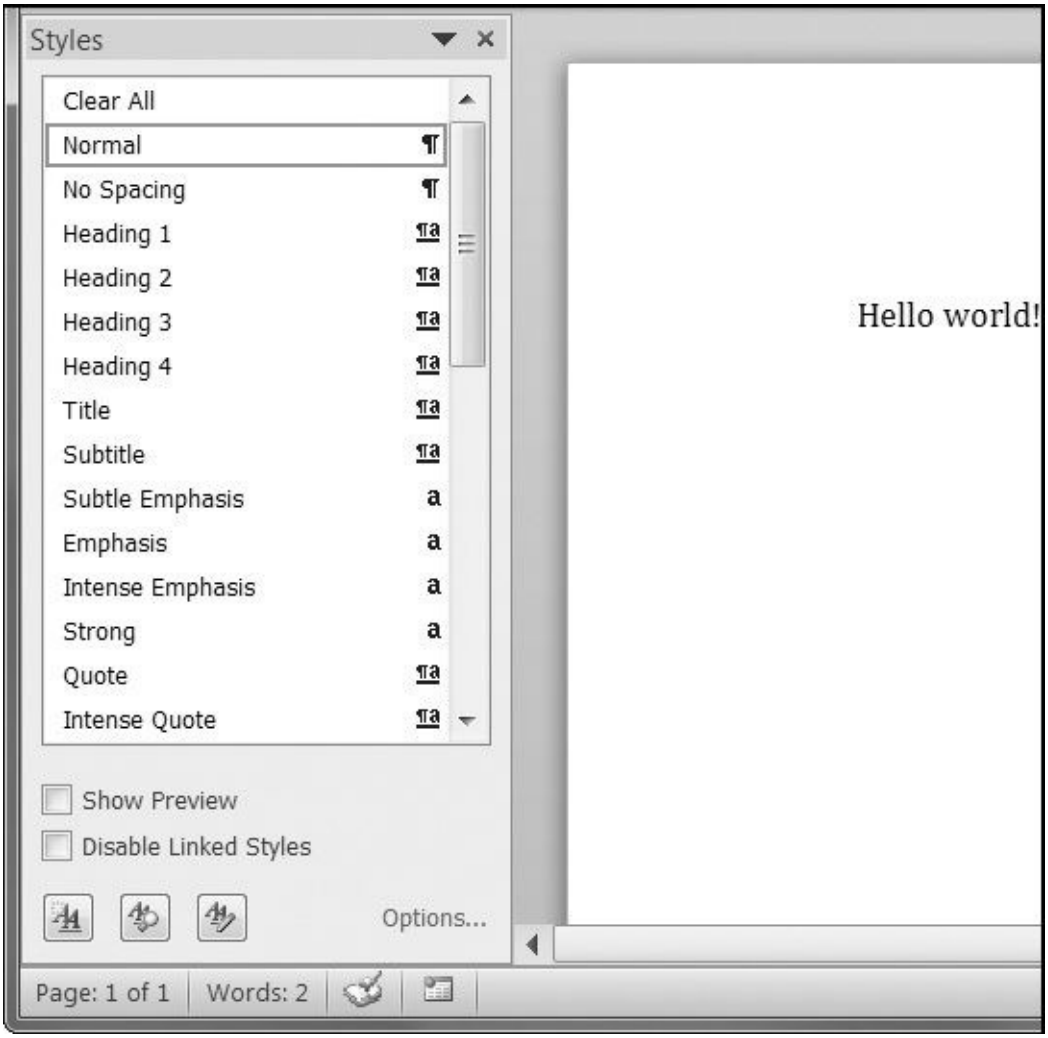


Figura 13-5. Visualice el panel Estilos pulsando CTRL-ALT-SHIFT-S en Windows.

Word y otros procesadores de texto utilizan estilos para que la presentación visual de tipos de texto similares sea coherente y fácil de cambiar. Por ejemplo, tal vez quieras que los párrafos del cuerpo tengan un texto de 11 puntos, Times New Roman, justificado a la izquierda y con la derecha irregular. Puedes crear un estilo con esta configuración y asignarlo a todos los párrafos del cuerpo del texto en . Luego, si más adelante quieres cambiar la presentación de todos los párrafos del cuerpo del documento, sólo tienes que cambiar el estilo y todos esos párrafos se actualizarán automáticamente.

Para los documentos Word, existen tres tipos de estilos: *Los estilos de párrafo* pueden aplicarse a los objetos de párrafo, *los estilos de carácter* pueden aplicarse a los objetos de ejecución y *los estilos enlazados* pueden aplicarse a ambos tipos de objetos. Puedes asignar estilos a los objetos Párrafo y Ejecutar estableciendo su atributo de estilo en una cadena. Esta cadena debe ser el nombre de un estilo. Si style se establece como None, entonces no habrá ningún estilo asociado con el objeto Paragraph o Run.

Los valores de cadena para los estilos predeterminados de Word son los siguientes:

Normal	Rúbrica5	ListBullet	ListaPárrafo
CuerpoTexto	Rúbrica6	ListBullet2	MacroTexto
CuerpoTexto2	Rúbrica7	ListBullet3	NoSpacing
CuerpoTexto3	Rúbrica8	ListaContinuar	Cita
Leyenda	Rúbrica9	ListaContinuar2	Subtítulo
Encabezamiento1	IntenseQuote	ListaContinuar3	TOCHeading
Rúbrica2	Lista	ListaNúmero	Título
Rúbrica3	Lista2	ListaNúmero2	
Rúbrica4	Lista3	ListaNúmero3	

Cuando defina el atributo de estilo, no utilice espacios en el nombre del estilo. Por ejemplo, mientras que el nombre del estilo puede ser Énfasis Sutil, debe establecer el atributo de estilo con el valor de cadena "ÉnfasisSutil" en lugar de 'ÉnfasisSutil'. Incluir espacios hará que Word interprete mal el nombre del estilo y no lo aplique.

Cuando utilices un estilo enlazado para un objeto Run, tendrás que añadir 'Char' al final de su nombre. Por ejemplo, para establecer el estilo vinculado Quote para un objeto Paragraph, utilizarías paragraphObj.style= "Quote", pero para un Run, utilizarías runObj.style= "QuoteChar".

En la versión actual de Python-Docx (0.7.4), los únicos estilos que se pueden usar son los estilos por defecto de Word y los estilos del *.docx* abierto. No se pueden crear nuevos estilos, aunque esto puede cambiar en futuras versiones de Python-Docx.

Creación de documentos Word con estilos no predeterminados

Si desea crear documentos Word que utilicen estilos más allá de los predeterminados, tendrá que abrir Word en un documento Word en blanco y crear los estilos usted mismo haciendo clic en el botón **Nuevo estilo** en la parte inferior del panel Estilos (la Figura 13-6 muestra esto en Windows).

Esto abrirá el diálogo Crear nuevo estilo desde Formato, donde puede introducir el nuevo estilo. Luego, vuelva al shell interactivo y abra este documento Word en blanco con `docx.Document()`, usándolo como base para su documento Word. El nombre que le diste a este estilo estará ahora disponible para usarlo con Python-Docx.

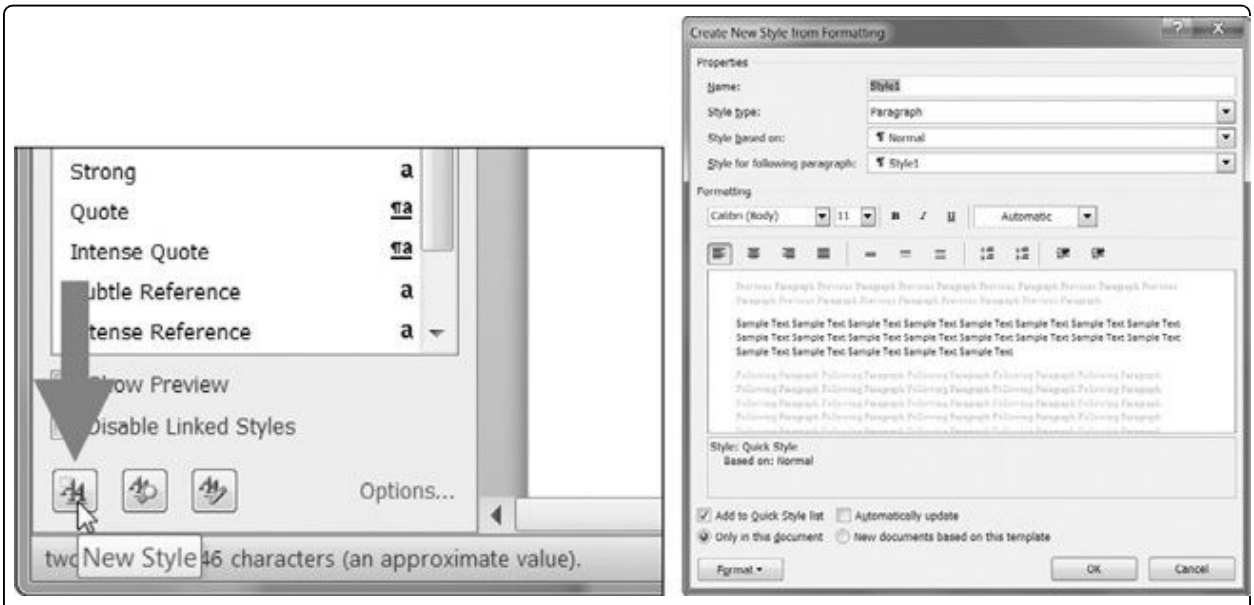


Figura 13-6. El botón Nuevo estilo (izquierda) y el diálogo Crear nuevo estilo a partir de formato (derecha)

Atributos de ejecución

Las tiradas pueden estilizarse aún más utilizando atributos de texto. Cada atributo puede tener uno de los tres valores siguientes: Verdadero (el atributo siempre está habilitado, independientemente de los demás estilos que se apliquen a la ejecución), Falso (el atributo siempre está deshabilitado) o Ninguno (por defecto, cualquiera que sea el estilo de la ejecución).

La Tabla 13-1 enumera los atributos de texto que pueden establecerse en los objetos Ejecutar.

Tabla 13-1. Atributos de texto *del objeto de* ejecución

Atributo	Descripción
negrita	El texto aparece en negrita.
cursiva	El texto aparece en cursiva.
subrayado	El texto está subrayado.
huelga	El texto aparece tachado.
doble_golpe	El texto aparece con doble tachado.
todas_las_tapas	El texto aparece en .
tapas_pequeñas	El texto aparece en , con las minúsculas dos puntos más pequeñas.
sombra	El texto aparece con una sombra.
esquema	El texto aparece contorneado en lugar de sólido.
rtl	El texto está escrito de derecha a izquierda.
imprimir	El texto aparece prensado en la página.
RELIEVE	El texto aparece levantado de la página en relieve.

Por ejemplo, para cambiar los estilos de *demo.docx*, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> doc= docx.Document('demo.docx')
>>> doc.PÁRRAFOS[0].texto
Título del documento
>>> doc.PÁRRAFOS[0].estilo
```

```
Título
>>> doc.paragraphs[0].style= 'Normal'
>>> doc.párrafos[1].texto
Un párrafo normal con negrita y cursiva".
>>> (doc.párrafos[1].tiradas[0].texto, doc.párrafos[1].tiradas[1].texto, doc.
párrafos[1].tiradas[2].texto, doc.párrafos[1].tiradas[3].texto)
('Un párrafo sencillo con algo de ', 'negrita', ' y algo de ', 'cursiva')
>>> doc.paragraphs[1].runs[0].style= 'QuoteChar'
>>> doc.paragraphs[1].runs[1].underline= True
>>> doc.paragraphs[1].runs[3].underline= True
>>> doc.save('reestilizado.docx')
```

Aquí, utilizamos los atributos de texto y estilo para ver fácilmente lo que hay en los párrafos de nuestro documento. Podemos ver que es sencillo dividir un párrafo en secciones y acceder a cada sección individualmente. Así que obtenemos la primera, segunda y cuarta tiradas del segundo párrafo, aplicamos estilo a cada tirada y guardamos los resultados en un nuevo documento.

Las palabras *Título del documento* en la parte superior de *restyled.docx* tendrán el estilo Normal en lugar del estilo Título, el objeto Ejecutar para el texto *Un párrafo sin formato con algunos* tendrá el estilo QuoteChar, y los dos objetos Ejecutar para las palabras *negrita* e *itálica* tendrán sus atributos de subrayado establecidos en Verdadero. La Figura 13-7 muestra el aspecto de los estilos de párrafos y ejecuciones en *restyled.docx*.

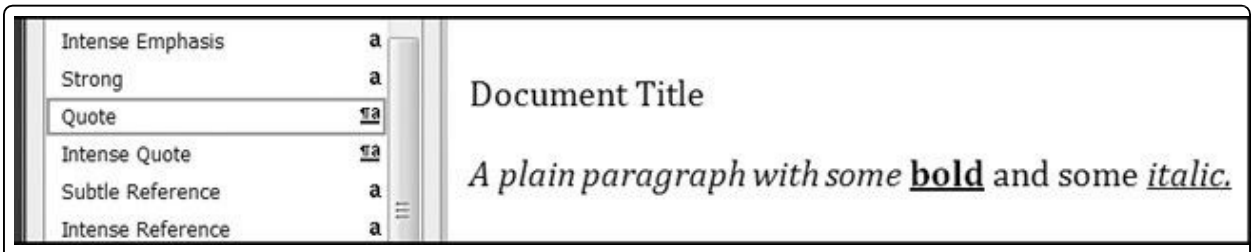


Figura 13-7. El archivo restyled.docx

Puede encontrar documentación más completa sobre el uso de estilos en Python-Docx en https://python-docx.readthedocs.org/en/latest/user/styles.html.

Redacción de documentos Word

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar docx
>>> doc= docx.Document()
>>> doc.add_paragraph('¡Hola mundo!')
<objeto docx.text.Paragraph en 0x0000000003B56F60>
>>> doc.save('helloworld.docx')
```

Para crear su propio archivo *.docx*, llame a docx.Document() para que devuelva un nuevo objeto Documento de Word en blanco. El método add_paragraph() añade un nuevo párrafo de texto al documento y devuelve una referencia al objeto Paragraph que se ha añadido. Cuando haya terminado de añadir texto, pase una cadena de nombre de archivo al método save() document para guardar el objeto Documento en un archivo.

Esto creará un archivo llamado *helloworld.docx* en el directorio de trabajo actual que, cuando se abra, tendrá el aspecto de la Figura 13-8.

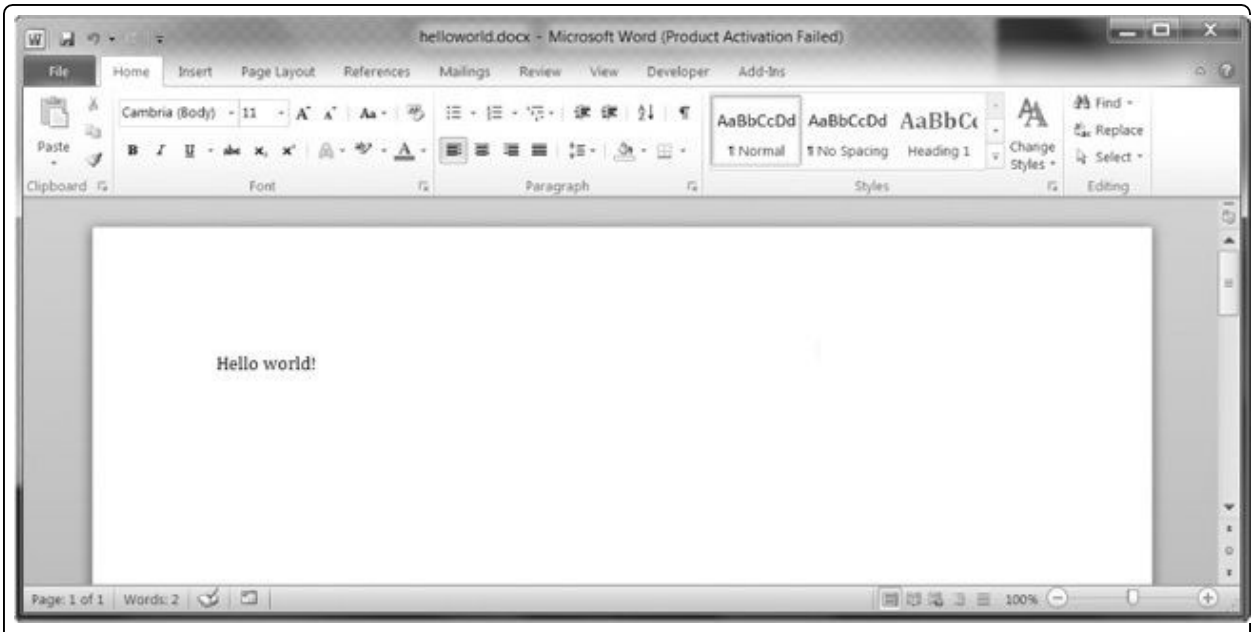


Figura 13-8. El documento Word creado usando add_paragraph('¡Hola mundo!')

Puede añadir párrafos llamando de nuevo al método add_paragraph() con el del nuevo párrafo. O para añadir texto al final de un párrafo existente, puedes llamar al método add_run() del y pasarle una cadena. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar docx
```

```
>>> doc= docx.Document()
>>> doc.add_paragraph('¡Hola mundo!')
<objeto docx.text.Paragraph en 0x000000000366AD30>
>>> paraObj1= doc.add_paragraph('ESTE ES un segundo párrafo.')
>>> paraObj2= doc.add_paragraph('ESTE es otro párrafo más.')
>>> paraObj1.add_run(' ESTE texto se añade al segundo párrafo.')
<objeto docx.text.Run en 0x0000000003A2C860>
>>> doc.save('MULTIPÁRRAFOS.docx')
```

El documento resultante tendrá el aspecto de la Figura 13-9. Observe que el texto *This text is being added to the second paragraph.* se añadió al objeto Paragraph en paraObj1, que era el segundo párrafo añadido a doc. Las funciones add_paragraph() y add_run() devuelven los objetos paragraph y Run, respectivamente, para ahorrarle la molestia de como un paso separado.

Ten en cuenta que a partir de la versión 0.5.3 de Python-Docx, los nuevos objetos Párrafo sólo pueden añadirse al final del documento, y los nuevos objetos Ejecutar sólo pueden añadirse al final de un objeto Párrafo.

El método save() puede ser llamado de nuevo para guardar los cambios adicionales que hayas hecho.

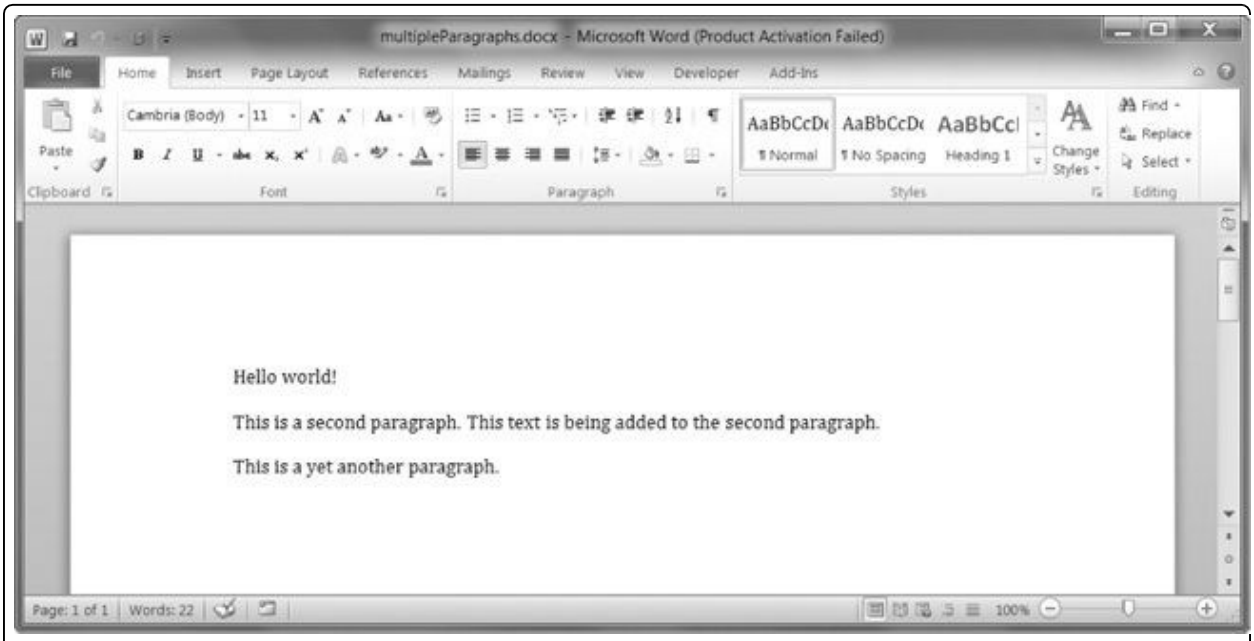


Figura 13-9. El documento con múltiples objetos Párrafo *y* Ejecutar *añadidos*

Tanto add_paragraph() como add_run() aceptan un segundo argumento opcional que es una cadena del estilo del objeto Paragraph o Run. Por ejemplo:

```
>>> doc.add_paragraph('¡Hola mundo!', 'Título')
```

Esta línea añade un párrafo con el texto *¡Hola mundo!* en el estilo Título.

Añadir títulos

Al llamar a add_heading() se añade un párrafo con uno de los estilos de encabezado. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> doc= docx.Document()
>>> doc.add_heading('Cabecera 0', 0)
<objeto docx.text.Paragraph en 0x00000000036CB3C8>
>>> doc.add_heading('Cabecera 1', 1)
<objeto docx.text.Paragraph en 0x00000000036CB630>
>>> doc.add_heading('Cabecera 2', 2)
<objeto docx.text.Paragraph en 0x00000000036CB828>
>>> doc.add_heading('Cabecera 3', 3)
<objeto docx.text.Paragraph en 0x00000000036CB2E8>
>>> doc.add_heading('Cabecera 4', 4)
<objeto docx.text.Paragraph en 0x00000000036CB3C8>
>>> doc.save('ENCABEZADOS.docx')
```

Los argumentos de add_heading() son una cadena con el texto del encabezamiento y un número entero de 0 a 4. El número entero 0 hace que el encabezamiento sea del estilo Título, que se utiliza para la parte superior del documento. Los enteros de 1 a 4 son para varios niveles de encabezado, siendo 1 el encabezado principal y 4 el subencabezado más bajo. La función add_heading() devuelve un objeto Párrafo para ahorrarle el paso de extraerlo del objeto Documento como un paso separado.

El archivo *headings.docx* resultante tendrá el aspecto de la Figura 13-10.



Figura 13-10. El documento headings.docx con los encabezamientos 0 a 4

Añadir saltos de línea y de página

Para añadir un salto de línea (en lugar de empezar un nuevo párrafo), puede llamar al método add_break() en el objeto Run tras el que desea que aparezca el salto. Si en cambio desea añadir un salto de página, debe pasar el valor docx.text.WD_BREAK.PAGE como argumento único a add_break(), como se hace en el centro del siguiente ejemplo:

```
>>> doc= docx.Document()
    >>> doc.add_paragraph('¡ESTO ESTÁ en la primera página!')
    <objeto docx.text.Paragraph en 0x0000000003785518>
>>> ❶doc.PARAGRAPHS[0].RUNS[0].add_break(docx.text.WD_BREAK.PAGE)
    >>> doc.add_paragraph('¡ESTO ESTÁ en la segunda página!')
    <objeto docx.text.Paragraph en 0x00000000037855F8>
    >>> doc.save('dosPáginas.docx')
```

Esto crea un documento Word de dos páginas con *¡Esto está en la primera página!* en la primera página y *¡Esto está en la segunda página!* en la segunda. Aunque todavía quedaba mucho espacio en la primera página después del texto *¡Esto está en la primera página!*, forzamos el siguiente párrafo a empezar en una página nueva insertando un salto de página después de la primera tirada del primer párrafo ❶.

Añadir imágenes

Los objetos Document tienen un método add_picture() que le permitirá añadir una imagen al final del documento. Digamos que tienes un archivo *zophie.png* en el directorio de trabajo actual. Puede añadir *zophie.png* al final de su documento con anchura de 1 pulgada y una altura de 4 centímetros (Word puede utilizar tanto unidades imperiales como métricas) introduciendo lo siguiente:

```
>>> doc.add_picture('zophie.png', width=docx.shared.INCHES(1),
height=docx.shared.Cm(4))
<objeto docx.shape.InlineShape en 0x00000000036C7D30>
```

El primer argumento es una cadena con el nombre de archivo de la imagen. Las palabras clave opcionales anchura y altura establecerán la anchura y la altura de la imagen en el documento. Si se omiten, la anchura y la altura se ajustarán por defecto al tamaño normal de la imagen.

Probablemente prefiera especificar la altura y la anchura de una imagen en unidades familiares como pulgadas y centímetros, por lo que puede utilizar las funciones docx.shared.Inches() y docx.shared.Cm() cuando especifique los argumentos de las palabras clave anchura y altura.

Resumen

La información de texto no es sólo para archivos de texto plano; de hecho, es bastante probable que trates con PDFs y documentos de Word mucho más a menudo. Puedes usar el módulo PyPDF2 para leer y escribir documentos PDF. Desafortunadamente, leer texto de documentos PDF puede no siempre resultar en una traducción perfecta a una cadena debido a
el complicado formato de archivo PDF, y algunos PDF podrían no ser legibles en absoluto. En estos casos, no tienes suerte a menos que futuras actualizaciones de PyPDF2 soporten características adicionales de PDF.

Los documentos de Word son más fiables, y puedes leerlos con el módulo python- docx. Puedes manipular texto en documentos Word mediante objetos Párrafo y Ejecutar. A estos objetos también se les pueden asignar estilos, aunque deben ser del conjunto de estilos por defecto o estilos ya existentes en el documento. Puedes añadir nuevos párrafos, encabezados, saltos e imágenes al documento, aunque sólo hasta el final.

Muchas de las limitaciones que conlleva trabajar con documentos PDF y Word se deben a que estos formatos están pensados para ser visualizados de forma agradable por los lectores humanos, en lugar de ser fáciles de analizar por el software. El siguiente capítulo examina otros dos formatos habituales para almacenar información: JSON y . Estos formatos están diseñados para ser utilizados por ordenadores, y verás que Python puede trabajar con estos formatos mucho más fácilmente.

Preguntas prácticas

P: 1. *No* se pasa un valor de cadena del nombre del fichero PDF a la función PyPDF2.PdfFileReader(). ¿Qué se pasa a la función en su lugar?

P: 2. ¿En qué modos deben abrirse los objetos Archivo para PdfFileReader() y PdfFileWriter()?

P: 3. ¿Cómo se adquiere un objeto Página para Acerca de este libro a partir de un objeto PdfFileReader?

P: 4. ¿Qué variable de PdfFileReader almacena el número de páginas del documento PDF?

P: 5. Si el PDF de un objeto PdfFileReader está cifrado con la contraseña swordfish, ¿qué debe hacer antes de poder obtener objetos Page de él?

P: 6. ¿Qué métodos se utilizan para rotar una página?

P: 7. ¿Qué método devuelve un objeto Documento para un archivo llamado *demo.docx*?

P: 8. ¿Cuál es la diferencia entre un objeto Párrafo y un objeto Ejecutar?

P: 9. ¿Cómo obtiene una lista de objetos Paragraph para un objeto Document que está almacenado en una variable ¿nombrado doc?

P: 10. ¿Qué tipo de objeto tiene las variables negrita, subrayado, cursiva, tachado y contorno?

P: 11. ¿Cuál es la diferencia entre establecer la variable negrita a Verdadero, Falso o Ninguno?

P: 12. ¿Cómo se crea un objeto Documento para un nuevo documento Word?

P: 13. ¿Cómo se añade un párrafo con el texto "!" a un objeto Documento almacenado en un objeto variable llamada doc?

P: 14. ¿Qué números enteros representan los niveles de encabezados disponibles en los documentos Word?

Proyectos prácticos

Para practicar, escribe programas que hagan lo siguiente.

PDF Paranoia

Utilizando la función `os.walk()` del capítulo 9, escriba una secuencia de comandos que recorra todos los PDF de una carpeta (y sus subcarpetas) y cifre los PDF utilizando una contraseña proporcionada en la línea de comandos. Guarde cada PDF cifrado con un *_encrypted.pdf* al nombre del archivo original. Antes de eliminar el archivo original, haga que el programa intente leer y descifrar el archivo para asegurarse de que se ha cifrado correctamente.

A continuación, escriba un programa que encuentre todos los PDF cifrados de una carpeta (y sus subcarpetas) y cree una copia descifrada del PDF utilizando una contraseña proporcionada. Si la contraseña es incorrecta, el programa debe imprimir un mensaje al usuario y continuar con el siguiente PDF.

Invitaciones personalizadas como documentos Word

Supongamos que tiene un archivo de texto con los nombres de los invitados. Este archivo *guests.txt* tiene un nombre por línea, como sigue:

```
Prof. Plum
Miss Scarlet
Col. Mustard Al
Sweigart
RoboCop
```

Escriba un programa que genere un documento Word con invitaciones personalizadas similares a las de la Figura 13-11.

Dado que Python-Docx sólo puede utilizar los estilos que ya existen en el documento de Word, tendrá que añadir primero estos estilos a un archivo de Word en blanco y luego abrir ese archivo con Python-Docx. Debería haber una invitación por página en el Word resultante, así que llame a `add_break()` para añadir un salto de página después del último de cada invitación. De este modo, sólo tendrá que abrir un documento de Word para imprimir todas las invitaciones a la vez.

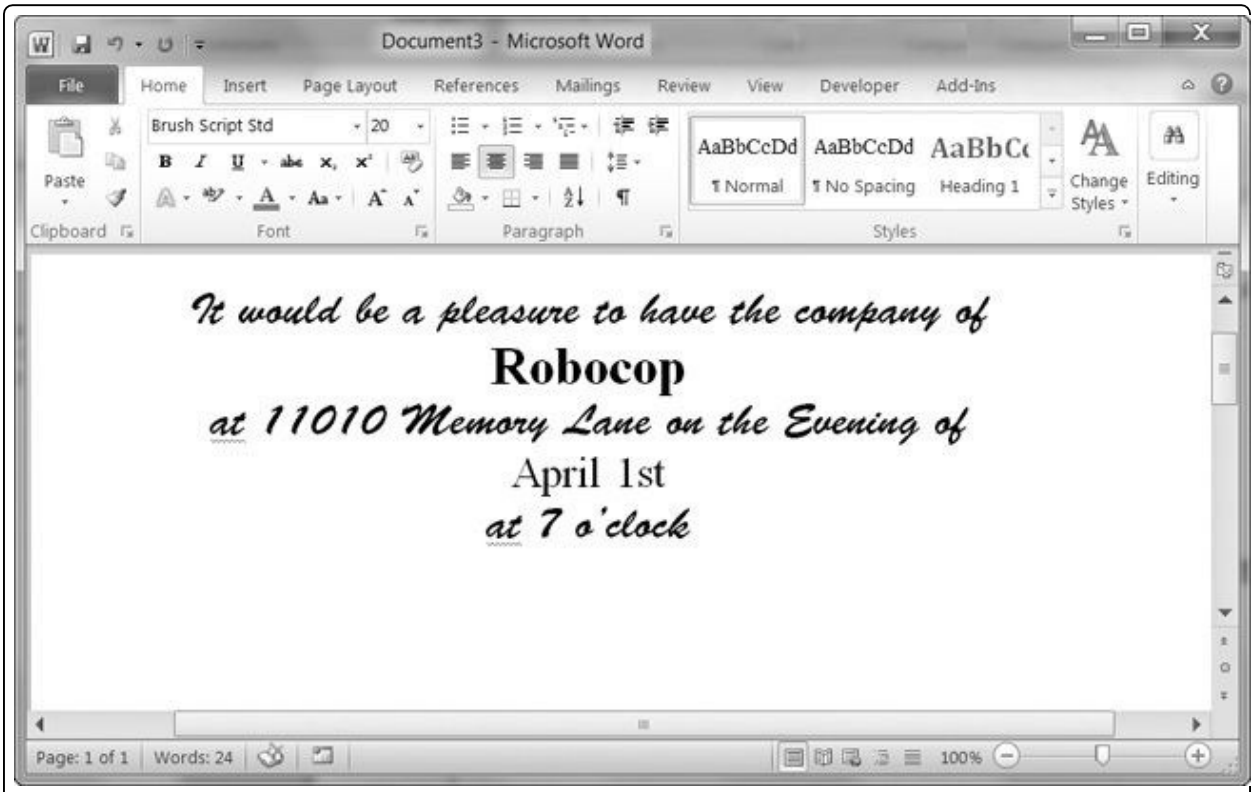


Figura 13-11. El documento Word generado por su script de invitación personalizado

Puede descargar un archivo *guests.txt* de muestra en http://nostarch.com/automatestuff/.

Brute-Force PDF Password Breaker

Supongamos que tienes un PDF cifrado cuya contraseña has olvidado, pero recuerdas que era una sola palabra en inglés. Intentar adivinar la contraseña olvidada es una tarea bastante aburrida. En su lugar, puede escribir un programa que descifre el PDF probando todas las palabras posibles en inglés hasta que encuentre una que funcione. Esto se llama un *ataque de fuerza bruta a la contraseña.* Descargue el archivo de texto *dictionary.txt* de http://nostarch.com/automatestuff/. Este *archivo de diccionario* contiene más de 44.000 palabras en inglés con una palabra por línea.

Usando las habilidades de lectura de archivos que aprendiste en el Capítulo 8, crea una lista de cadenas de palabras leyendo este archivo. Luego haga un bucle sobre cada palabra en esta , pasándola al método decrypt(). Si este método devuelve el entero 0, la contraseña era incorrecta y su programa debería continuar con la siguiente contraseña. Si decrypt() devuelve 1, entonces su programa debe salir del bucle e imprimir la hackeada. Deberías probar tanto la forma en mayúscula como en minúscula de cada palabra. (En mi portátil, repasar las 88.000 palabras en mayúsculas y minúsculas del archivo del diccionario lleva un par de minutos. Por eso no deberías usar una simple palabra en inglés para tus contraseñas).

Capítulo 14. Trabajar con archivos CSV y datos JSON Trabajar con archivos CSV y datos JSON

En el capítulo 13 aprendiste a extraer texto de documentos PDF y Word. Estos archivos estaban en formato binario, lo que requería módulos especiales de Python para acceder a sus datos. Los archivos CSV y JSON, por otro , son simplemente archivos de texto plano. Puedes verlos en un editor de texto, como el editor de archivos de IDLE. Pero Python también viene con los módulos especiales csv y json, cada uno proporcionando funciones para ayudar a
trabajar con estos formatos de archivo.

CSV significa "valores separados por comas", y los archivos CSV son hojas de cálculo simplificadas almacenadas como archivos de texto plano. El módulo csv de Python facilita el análisis de archivos CSV.

JSON (pronunciado "JAY-sawn" o "Jason", no importa cómo porque de cualquier forma la gente dirá que lo pronuncias mal) es un formato que almacena información como código fuente JavaScript en archivos de texto plano.

(JSON es la abreviatura de JavaScript Object Notation.) No es necesario conocer el lenguaje de programación JavaScript para utilizar archivos JSON, pero es útil conocer el formato JSON porque se utiliza en muchas aplicaciones web.

El módulo CSV

Cada línea de un archivo CSV representa una fila de la hoja de cálculo, y las comas separan las celdas de la fila. Por ejemplo, la hoja de cálculo *example.xlsx* de *http://nostarch.com/automatestuff/* tendría este aspecto en un archivo CSV:

```
4/5/2015 13:34,Manzanas,73
4/5/2015 3:41,Cerezas,85
4/6/2015 12:46,Peras,14
4/8/2015 8:59,Naranjas,52
4/10/2015 2:07,Manzanas,152
4/10/2015 18:10,Plátanos,23
4/10/2015 2:40,Fresas,98
```

Utilizaré este archivo para los ejemplos interactivos de shell de este capítulo. Puedes descargar *example.csv* de *http://nostarch.com/automatestuff/* o introducir el texto en un editor de texto y guardarlo como *example.csv*.

Los archivos CSV son sencillos y carecen de muchas de las características de una hoja de cálculo Excel. Por ejemplo, los archivos CSV

- No tienen tipos para sus valores-todo es una cadena
- No tiene ajustes de tamaño o color de fuente
- No tenga varias hojas de cálculo
- No se puede especificar la anchura y la altura de las celdas
- No puede haber celdas fusionadas
- No pueden tener imágenes ni gráficos incrustados

La ventaja de los archivos CSV es su simplicidad. Los archivos CSV son ampliamente compatibles con muchos tipos de programas, pueden visualizarse en editores de texto (incluido el editor de archivos de IDLE) y son una forma sencilla de representar datos de hojas de cálculo. El formato CSV es exactamente como se anuncia: Es sólo un archivo de texto de valores separados por comas.

Dado que los archivos CSV son sólo archivos de texto, podrías sentir la tentación de leerlos como una cadena y luego procesar esa cadena utilizando las técnicas que aprendiste en el Capítulo 8. Por ejemplo, dado que cada celda en un archivo CSV está separada por una coma, tal vez podría simplemente llamar al método `split()` en cada línea de texto para obtener los valores. Pero no todas las comas de un archivo CSV representan el límite entre dos celdas. CSV
también tienen su propio conjunto de caracteres de escape para permitir la inclusión de comas y otros caracteres *como parte de los valores*. El método `split()` no maneja estos caracteres de escape. Debido a estos peligros potenciales, debería

utilice siempre el módulo `csv` para leer y escribir archivos CSV.

Objetos de lectura

Para leer datos de un archivo CSV con el módulo csv, es necesario crear un objeto Lector. Un objeto Reader le permite iterar sobre las líneas del archivo CSV. Introduzca lo siguiente en el shell interactivo, con *ejemplo.csv* en el directorio de trabajo actual:

```
>>> ❶importar csv

>>> ❷ejemploArchivo= open('ejemplo.csv')

➲  >>> exampleReader= csv.reader(ejemploArchivo)

>>> ❹ejemploDatos= lista(ejemploLector)

>>> ❹ejemploDatos
    [['4/5/2015 13:34', 'Manzanas', '73'], ['4/5/2015 3:41', 'Cerezas', '85'],
    ['4/6/2015 12:46', 'Peras', '14'], ['4/8/2015 8:59', 'Naranjas', '52'],
    ['4/10/2015 2:07', 'Manzanas', '152'], ['4/10/2015 18:10', 'Plátanos', '23'],
    ['4/10/2015 2:40', 'Fresas', '98']]
```

El módulo csv viene con Python, así que podemos importarlo ❶ sin tener que instalarlo primero.

Para leer un fichero CSV con el módulo csv, ábralo primero mediante la función open() ❷, como lo harías con cualquier otro archivo de texto. Pero en lugar de llamar al método read() o readlines() en el objeto File que devuelve open(), pásalo a la función csv.reader() ❸. Esto devolverá un objeto Reader para que lo utilices. Tenga en cuenta que no se pasa una cadena de nombre de archivo directamente a la función csv.reader().

La forma más directa de acceder a los valores del objeto Reader es convertirlo en una lista simple de Python pasándolo a list()❹. Usando list() en este objeto Reader devuelve una lista de listas, que puedes almacenar en una variable como exampleData.

Al introducir exampleData en el intérprete de comandos se muestra la lista de listas❺.

Ahora que tiene el archivo CSV como una lista de listas, puede acceder al valor de una fila y columna concretas con la expresión exampleData[row][col], donde row es el índice de una de las listas de exampleData y col es el índice del elemento que desea de esa lista. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> exampleData[0][0]
'4/5/2015 13:34'
>>> ejemploDatos[0][1]
Manzanas
>>> ejemploDatos[0][2]
'73'
>>> ejemploDatos[1][1]
```

```
Cerezas
>>> ejemploDatos[6][1]
Fresas
```

ejemploDatos[0][0] va a la primera lista y nos da la primera cadena, ejemploDatos[0][2] va a la primera lista y nos da la tercera cadena, y así sucesivamente.

Lectura de datos de objetos de lectura en un bucle for

Para archivos CSV de gran tamaño, es conveniente utilizar el objeto Reader en un bucle for. Esto evita cargar todo el archivo en memoria de una sola vez. Por ejemplo, introduzca lo siguiente en el shell interactivo:

```
>>> importar csv
>>> ejemploArchivo= open('ejemplo.csv')
>>> exampleReader= csv.reader(ejemploArchivo)
>>> para fila en ejemploReader:
        print('Fila #'+ str(exampleReader.line_num)+ ' '+ str(fila))

Fila #1 ['4/5/2015 13:34', 'Manzanas', '73']
Fila #2 ['4/5/2015 3:41', 'Cerezas', '85']
Fila #3 ['4/6/2015 12:46', 'Peras', '14']
Fila #4 ['4/8/2015 8:59', "Naranjas", '52']
Fila #5 ['4/10/2015 2:07', 'Manzanas', '152']
Fila #6 ['4/10/2015 18:10', 'Bananas', '23']
Fila #7 ['4/10/2015 2:40', 'Fresas', '98']
```

Después de importar el módulo csv y crear un objeto Lector a partir del archivo CSV, puede recorrer las filas del objeto Lector. Cada fila es una lista de valores, donde cada valor representa una celda.

La llamada a la función print() imprime el número de la fila actual y el contenido de la fila. Para obtener el número de fila, utilice la variable line_num del objeto Reader, que contiene el número de la línea actual.

El objeto Reader sólo se puede volver a leer una vez. Para volver a leer el archivo CSV, debe llamar a csv.reader para crear un objeto Reader.

Objetos del escritor

Un objeto `Writer` permite escribir datos en un archivo CSV. Para crear un objeto `Writer`, utilice la función `csv.writer()`. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar csv
>>> ❶outputFile= open('output.csv', 'w', newline='')
>>> ❷outputWriter= csv.writer(outputFile)
>>> outputWriter.writerow(["spam", 'HUEVOS', 'bacon', 'jamón'])
21
>>> outputWriter.writerow(["¡Hola, mundo!', 'HUEVOS', 'bacon', 'jamón'])
32
>>> outputWriter.writerow([1, 2, 3.141592, 4])
16
>>> outputFile.close()
```

Primero, llama a `open()` y pásale `'w'` para abrir un archivo en modo escritura❶. Esto creará el objeto que luego puedes pasar a `csv.writer()` ❷ para crear un objeto `Writer`.

En Windows, también necesitará pasar una cadena en blanco para el argumento de la palabra clave `newline` de la función `open()`. Por razones técnicas que van más allá del alcance de este libro, si olvida establecer el argumento `de la nueva línea`, las filas del *archivo output.csv* aparecerán a doble espacio, como se muestra en la Figura 14-1.

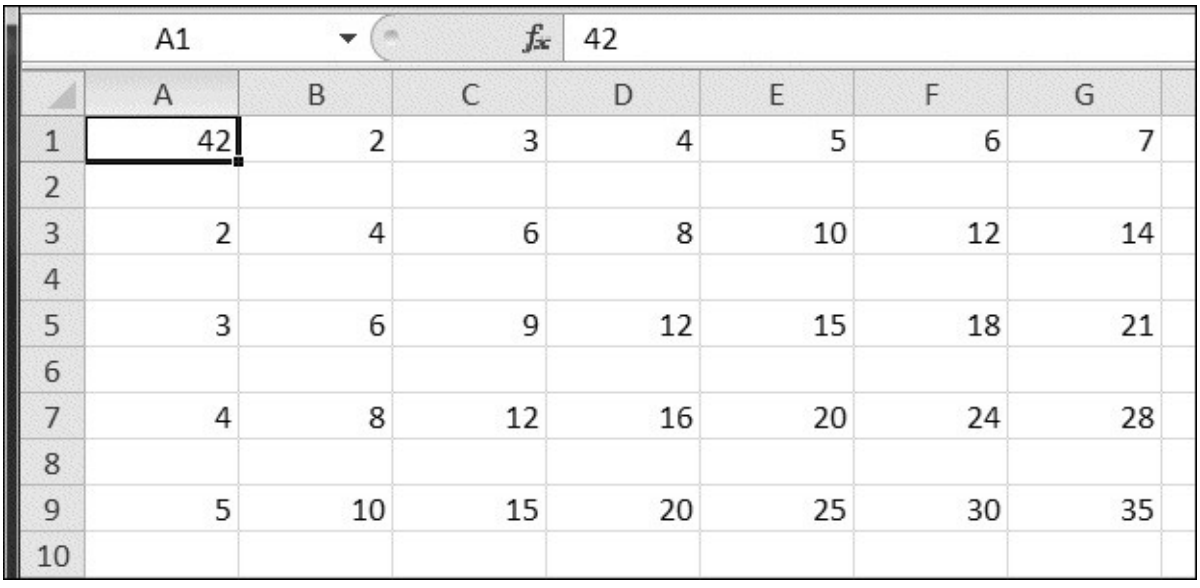


Figura 14-1. Si olvida el argumento de palabra clave `newline=''` *en* `open()`*, el archivo CSV tendrá doble espacio.*

El método `writerow()` para objetos `Writer` toma como argumento una lista. Cada valor de la lista se coloca en su propia celda en el archivo CSV de salida. El valor de retorno `de writerow()` es el número de caracteres escritos en el archivo para esa fila (incluidos los caracteres de nueva línea).

Este código produce un archivo *output.*csv con el siguiente aspecto:

```
spam, huevos, bacon, jamón
"¡Hola, mundo!",huevos,bacon,jamón
1,2,3.141592,4
```

Observe cómo el objeto `Writer` escapa automáticamente la coma en el valor `'¡Hola, mundo!'` con comillas dobles en el archivo CSV. El módulo `csv` te ahorra tener que manejar tú mismo estos casos especiales.

Argumentos de las palabras clave delimiter y lineterminator

Digamos que desea separar las celdas con un carácter de tabulación en lugar de una coma y desea que las filas estén a doble espacio. Podrías introducir algo como lo en el intérprete de comandos interactivo:

```
>>> importar csv
    >>> csvFile= open('ejemplo.TSV', 'w', newline='')
>>> ❶csvWriter= CSV.writer(CSVFILE, delimiter='\t', lineterminator='\n\n')
    >>>  csvWriter.writerow(["MANZANAS",  'NARANJAS',  'UVAS'])
    24
    >>> csvWriter.writerow(["HUEVOS", 'bacon', 'jamón'])
    17
    >>> csvWriter.writerow(["spam", 'spam', 'spam', 'spam', 'spam', 'spam'])
    32
    >>> CSVFILE.close()
```

Esto cambia los caracteres delimitador y terminador de línea de su fichero. El *delimitador* es el carácter que aparece entre las celdas de una fila. Por defecto, delimitador de un archivo CSV es una coma. El *terminador de línea* es el carácter que aparece al final de una fila. Por defecto, el terminador de línea es una nueva línea. Puede cambiar los caracteres a valores diferentes utilizando los argumentos de palabra clave `delimiter` y `lineterminator` con `csv.writer()`.

Pasando `delimitador= '\t'` y `lineterminador= '\n\n'` ❶ cambia el carácter entre celdas a un tabulador y el carácter entre filas a dos nuevas líneas. A continuación, llamamos a `writerow()` tres veces para obtener tres filas.

Esto produce un archivo llamado *ejemplo.tsv* con el siguiente contenido:

```
manza    naranjas  uvas
nas

huev     spam de   spam de
os       bacon     jamón     spam      spam      spam
spam
```

Ahora que nuestras celdas están separadas por tabuladores, estamos utilizando la extensión de archivo *.tsv*, para valores separados por tabuladores.

Proyecto: Eliminación de la cabecera de los archivos CSV

Supongamos que tiene la aburrida tarea de eliminar la primera línea de varios cientos de archivos CSV. Tal vez los vayas a introducir en un proceso automatizado que sólo necesita los datos y no los encabezados de la parte superior de las columnas. *Podrías* abrir cada archivo en Excel, eliminar la primera fila y volver a guardarlo, pero tardarías horas. un programa que lo haga.

El programa tendrá que abrir todos los archivos con la extensión *.csv* en el directorio de trabajo actual, leer el contenido del archivo CSV y reescribir el contenido sin la primera fila en un archivo del mismo nombre. Esto reemplazará el antiguo contenido del archivo CSV por el nuevo contenido sin la primera fila.

> **NOTA**
>
> *Como siempre, cuando escribas un programa que modifique archivos, asegúrate de hacer una copia de seguridad de los archivos, primero por si acaso tu programa no funciona como esperas. No querrás borrar accidentalmente tus archivos originales.*

A alto nivel, el programa debe hacer lo siguiente:

- Buscar todos los archivos CSV en el directorio de trabajo actual.
- Lee el contenido completo de cada archivo.
- Escriba el contenido, saltándose la primera línea, en un nuevo archivo CSV.

A nivel de código, esto significa que el programa tendrá que hacer lo siguiente:

- Recorre una lista de archivos desde `os.listdir()`, omitiendo los archivos que no son CSV.
- Crear un objeto CSV `Reader` y leer contenido del archivo, utilizando la función line_num para saber qué línea debe omitirse.
- Cree un objeto CSV `Writer` y los datos leídos en el nuevo archivo. Para este

proyecto, abra una nueva ventana de editor de archivos y guárdelo como *removeCsvHeader.py*.

Paso 1: Recorrer cada archivo CSV

Lo primero que debe hacer su programa es recorrer una lista de todos los nombres de archivos CSV del directorio de trabajo actual. Haz que tu *removeCsvHeader.py* tenga este aspecto:

```
#! python3
    # removeCsvHeader.py - Elimina la cabecera de todos los archivos CSV del #
    directorio de trabajo actual.

    import csv, os

    os.makedirs("headerRemoved", exist_ok=True)

    # Recorrer todos los archivos del directorio de trabajo actual.
    for csvFilename in os.listdir('.'):
            if not csvFilename.endswith(".csv"):
 ❶              continuar       # skip non-csv files

        print("Eliminando cabecera de '+ csvFilename+ '...') #

        TODO: Leer el fichero CSV (saltando la primera

        fila).

        # TODO: Escribir el archivo CSV.
```

La llamada a os.makedirs() creará una carpeta headerRemoved donde se escribirán todos los archivos CSV sin cabecera. Un bucle for en `os.listdir('.')` te lleva hasta allí, pero hará un bucle sobre *todos los* archivos en el directorio de trabajo, por lo tendrás que añadir algún código al principio del bucle que omita los nombres de archivo no terminen con .csv. La sentencia continue ❶ hace que el bucle for pase al siguiente nombre de fichero cuando se encuentre con un fichero que no sea CSV.

Sólo para que haya *alguna* salida mientras el programa se ejecuta, imprime un mensaje diciendo en qué archivo CSV está trabajando el programa. A continuación, añade algunos comentarios TODO para lo que el resto del programa debe hacer.

Paso 2: Leer el archivo CSV

El programa no elimina la primera línea del archivo CSV. En su lugar, crea una nueva copia del archivo CSV sin la primera línea. Como el nombre de archivo de la copia es el mismo que el del archivo original, la copia sobrescribirá el original.

El programa necesitará una forma de rastrear si está actualmente en bucle en la primera fila. Añade lo siguiente a *removeCsvHeader.py*.

```
#! python3
# removeCsvHeader.py - Elimina la cabecera de todos los archivos CSV en # el
directorio de trabajo actual.

--snip--
# Leer el archivo CSV (saltando la primera fila).
csvRows= []
csvFileObj= open(csvFilename)
readerObj= csv.reader(csvFileObj) for
row in readerObj:
    if readerObj.line_num== 1:
        continue    # saltar primera
        fila
    csvRows.append(fila)
csvFileObj.close()

# TODO: Escribir el archivo CSV.
```

El atributo line_num del objeto Reader puede utilizarse para determinar qué línea del archivo CSV está leyendo actualmente. Otro bucle for recorrerá las filas devueltas por el objeto Reader CSV, y todas las filas excepto la primera se añadirán a csvRows.

A medida que el bucle for itera sobre cada fila, el código comprueba si readerObj.line_num es igual a 1. Si es así, ejecuta un continue para pasar a la siguiente fila sin añadirla a csvRows. Para cada fila posterior, la condición será siempre False, y la fila se añadirá a csvRows.

Paso 3: Escribir el archivo CSV sin la primera fila

Ahora que csvRows contiene todas las filas menos la primera, la lista debe escribirse en un archivo CSV en la carpeta *headerRemoved*. Añade lo siguiente a *removeCsvHeader.py*:

```
#! python3
    # removeCsvHeader.py - Elimina la cabecera de todos los archivos CSV del #
    directorio de trabajo actual.
    --snip--

    # Recorre todos los archivos del directorio de trabajo actual.
❶ for csvFilename in os.listdir('.'):
        if not csvFilename.endswith(".csv"):
            continue      # omitir archivos no CSV

        --snip--

        # Escribe el archivo CSV.
        csvFileObj= open(os.path.join('headerRemoved', csvFilename), 'w', newline='')
        csvWriter= csv.writer(csvFileObj) for
        row in csvRows:
            csvWriter.writerow(fila) csvFileObj.close()
```

El objeto CSV Writer escribirá la lista en un archivo CSV en headerRemoved usando csvFilename (que también usamos en el lector CSV). Esto sobrescribirá el archivo original.

Una vez creado el objeto Writer, hacemos un bucle sobre las sublistas almacenadas en csvRows y escribir cada sublista en el archivo.

Una vez ejecutado el código, el bucle for externo ❶ pasará al siguiente nombre de fichero de os.listdir('.'). Cuando ese bucle termine, el programa estará completo.

Para probar su programa, descargue *removeCsvHeader.*zip de *http://nostarch.com/automatestuff/* y descomprímalo en una carpeta. Ejecute el programa *removeCsvHeader.py* en esa carpeta. El resultado será el siguiente:

```
Eliminar encabezado de NAICS_data_1048.csv... Eliminar
encabezado de NAICS_data_1218.csv...
```

```
--snip--
Eliminar encabezado de NAICS_data_9834.csv... Eliminar
encabezado de NAICS_data_9986.csv...
```

Este programa debe imprimir un nombre de archivo cada vez que elimine la primera línea de un archivo CSV.

Ideas para programas similares

Los programas que puedes escribir para archivos CSV son similares a los que puedes escribir para archivos Excel, ya que ambos son archivos de hojas de cálculo. Podrías escribir programas para hacer lo siguiente:

- Compare datos entre distintas filas de un archivo CSV o entre varios archivos CSV.
- Copie datos específicos de un archivo CSV a un archivo Excel, o viceversa.
- Compruebe si hay datos no válidos o errores de formato en los archivos CSV y avise al usuario de estos errores.
- Lea datos de un archivo CSV como entrada para sus programas Python.

JSON y API

JavaScript Object Notation es una forma popular de formatear datos como una única cadena legible por humanos. JSON es la forma nativa en que los programas JavaScript escriben sus estructuras de datos y suele parecerse a lo que produciría la función `pprint()` de Python. No es necesario conocer JavaScript para trabajar con datos formateados en JSON.

He aquí un ejemplo de datos formateados como JSON:

```
{"name": "Zophie", "isCat": true,
 "miceCaught": 0, "napsTaken": 37.5,
 "felineIQ": null}
```

Es útil conocer JSON, porque muchos sitios web ofrecen contenido JSON como forma de que los programas interactúen con el sitio web. Esto se conoce como proporcionar una *interfaz de programación de aplicaciones (API)*. Acceder a una API es lo mismo que acceder a cualquier otra página web a través de una URL. La diferencia es que los datos devueltos por una API están formateados (con JSON, por ejemplo) para máquinas; las API no son fáciles de leer para las personas.

Muchos sitios web ofrecen sus datos en formato JSON. Facebook, Twitter, Yahoo, Google, Tumblr, Wikipedia, Flickr, Data.gov, Reddit, IMDb, Rotten Tomatoes, LinkedIn y muchos otros sitios populares ofrecen API para que los programas las utilicen. Algunos de estos sitios requieren registro, que casi siempre es gratuito. Tendrás que encontrar documentación sobre las URL que tu programa debe solicitar para obtener los datos que quieres, así como el formato general de las estructuras de datos JSON que se devuelven. Esta documentación debe ser proporcionada por el sitio que ofrece la API; si tienen una página de "Desarrolladores", busque la documentación allí.

Utilizando APIs, puedes escribir programas que hagan lo siguiente:

- Extraer datos sin procesar de sitios web. (Acceder a las API suele ser más cómodo que descargar páginas web y analizar HTML con Beautiful Soup).
- Descarga automáticamente las nuevas publicaciones de una de tus cuentas de redes sociales y publícalas en otra cuenta. Por ejemplo, puedes tomar tus publicaciones de Tumblr y publicarlas en Facebook.
- Crea una "enciclopedia de películas" para tu colección personal de películas extrayendo datos de IMDb, Rotten Tomatoes y Wikipedia y poniéndolos en un único archivo de texto en tu ordenador.

Puede ver algunos ejemplos de API JSON en los recursos de http://nostarch.com/automatestuff/.

El módulo JSON

El módulo json de Python maneja todos los detalles de la traducción entre una cadena con datos JSON y valores Python para las funciones json.loads() y json.dumps(). JSON no puede almacenar *todo tipo de* valores Python. Sólo puede contener valores de los siguientes tipos de datos: cadenas, enteros, flotantes, booleanos, listas, diccionarios y NoneType. JSON no puede representar objetos específicos de Python, como objetos File, objetos CSV Reader o Writer, objetos Regex, u objetos Selenium WebElement.

Lectura de JSON con la función loads()

Para traducir una cadena que contenga datos JSON a un valor Python, pásela a la función json.loads(). (El nombre significa "cadena de carga", no "cargas".) Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> stringOfJsonData= '{"name": "Zophie", "isCat": true, "miceCaught": 0, "felineIQ": null}'
>>> importar json
>>> jsonDataAsPythonValue= json.loads(stringOfJsonData)
>>> jsonDataAsPythonValue
{'isCat': True, 'miceCaught': 0, 'name': 'Zophie', 'felineIQ': Ninguno}
```

Después de importar el módulo json, puedes llamar a loads() y pasarle una cadena de datos JSON. Ten en cuenta que las cadenas JSON siempre usan comillas dobles. Devolverá esos datos como un diccionario Python. Los diccionarios Python no están ordenados, por lo que los pares clave-valor pueden aparecer en un orden diferente cuando imprimas jsonDataAsPythonValue.

Escribir JSON con la función dumps()

función json.dumps() (que significa "volcar cadena", no "volcar") traducirá un valor Python en una cadena de datos con formato JSON. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> pythonValue= {'isCat': True, 'miceCaught': 0, 'name': 'Zophie', 'felineIQ':
None}
>>> importar json
>>> stringOfJsonData= json.dumps(pythonValue)
>>> stringOfJsonData
'{"isCat": true, "felineIQ": null, "miceCaught": 0, "name": "Zophie" }'
```

El valor sólo puede ser uno de los siguientes tipos de datos básicos de Python: diccionario, lista, entero, flotante, cadena, booleano o Ninguno.

Proyecto: Obtención de datos meteorológicos actuales

Consultar el tiempo parece bastante trivial: Abrir el navegador, hacer clic en la barra de direcciones, escribir la URL de un sitio web sobre el tiempo (o buscar uno y hacer clic en el enlace), esperar a que se cargue la página, pasar por alto todos los anuncios, etcétera.

En realidad, hay muchos pasos aburridos que podrías saltarte si tuvieras un programa que descargara la previsión meteorológica para los próximos días y la imprimiera como texto plano. Este programa utiliza el módulo de peticiones del Capítulo 11 para descargar datos de la Web.

En general, el programa hace lo siguiente

- Lee la ubicación solicitada desde la línea de comandos.
- Descarga datos meteorológicos JSON de OpenWeatherMap.org.
- Convierte la cadena de datos JSON en una estructura de datos Python.
- Imprime el tiempo para hoy y los próximos dos días. Así que el código tendrá que hacer lo siguiente:
- Une las cadenas en sys.argv para obtener la ubicación.
- Llama a requests.get() para descargar los datos meteorológicos.
- Llama a json.loads() para convertir los datos JSON en una estructura de datos Python.
- Imprime la previsión meteorológica.

Para este proyecto, abre una nueva ventana del editor de archivos y guárdalo como *quickWeather.py*.

Paso 1: Obtener la ubicación del argumento de la línea de comandos

La entrada para este programa vendrá de la línea de comandos. Hacer *quickWeather.py* tiene este aspecto:

```
#! python3
# quickWeather.py - Imprime el tiempo de un lugar desde la línea de comandos. import

json, requests, sys

# Calcula la ubicación a partir de los argumentos de la línea
de comandos. if len(sys.argv)< 2:
    print('Uso: quickWeather.py ubicación') sys.exit()
ubicación= ' '.join(sys.argv[1:])

# TODO: Descargar los datos JSON desde la API de OpenWeatherMap.org.

# TODO: Cargar los datos JSON en una variable Python.
```

En Python, los argumentos de la línea de comandos se almacenan en la lista sys.argv. Después de la línea #! shebang y las sentencias import, el programa comprobará si hay más de un argumento en la línea de comandos. (Recuerda que sys.argv siempre tendrá al menos un elemento, sys.argv[0], que contiene el nombre de archivo del script Python). Si sólo hay un elemento en la lista, entonces el usuario no proporcionó una ubicación en la línea de comandos, y un mensaje de "uso" será proporcionado al usuario antes de que el programa termine.

Los argumentos de la línea de comandos se dividen en espacios. El argumento de línea de comandos San Francisco, CA haría que sys.argv contuviera ['quickWeather.py', 'San', 'Francisco,', 'CA']. Por lo tanto, llama al método join() para unir todas las cadenas excepto la primera en sys.argv. Guarda esta cadena unida en una variable llamada location.

Paso 2: Descargar los datos JSON

OpenWeatherMap.org ofrece información meteorológica en tiempo real en formato JSON. Tu programa simplemente tiene que descargar la página en *http://api.openweathermap.org/data/2.5/forecast/daily?q=<Location>&cnt=3*, donde *<Location>* es el nombre de la ciudad cuyo tiempo quieres. Añade lo siguiente a *quickWeather.py*.

```
#! python3
# quickWeather.py - Imprime el tiempo de un lugar desde la línea de comandos.

--snip--

# Descarga los datos JSON de la API DE OpenWeatherMap.org.
url ='http://api.openweathermap.org/data/2.5/forecast/daily?q=%s&cnt=3' % (ubicación)
respuesta= REQUESTS.get(url)
response.RAISE_FOR_STATUS()

# TODO: Cargar datos JSON en una variable Python.
```

Tenemos la ubicación de nuestros argumentos de línea de comandos. Para crear la URL a la queremos acceder, utilizamos el marcador de posición %s e insertamos la cadena almacenada en location en ese lugar de la cadena URL. Almacenamos el resultado en url y pasamos url a requests.get(). La llamada a requests.get() devuelve un objeto Response, que puede comprobar si hay errores llamando a raise_for_status(). Si no se lanza ninguna excepción, el texto descargado estará en response.text.

Paso 3: Cargar datos JSON e imprimir el tiempo

La variable miembro response.text contiene una gran cadena datos con formato JSON. Para convertir esto en un valor Python, llama a la función json.loads(). Los datos JSON tendrán este aspecto:

```
{'city': {'coord': {'lat': 37.7771, 'lon': -122.42},
          'country': 'Estados Unidos de América', 'id':
          '5391959',
          nombre": "San Francisco",
          "población": 0},
'cnt': 3,
'cod': '200',
'lista': [{'nubes': 0,
          'deg': 233,
          'dt': 1402344000,
          humedad":   58,
          'presión': 1012,23,
          velocidad: 1,96,
             'temp': {'día': 302.29,
                    'eve': 296.46,
                     "max": 302.29,
                      "min": 289.77,
                     "morn":   294.59,
                     'noche': 289,77},
          'tiempo': [{'descripción': 'el cielo está despejado',
                      'icon': '01d',
--snip--
```

Puede ver estos datos pasando weatherData a pprint.pprint(). Puede consultar *http://openweathermap.org/* para obtener más documentación sobre el significado de estos campos. Por ejemplo, la documentación en línea le dirá que el campo 302,29 después de "día" es la temperatura diurna en grados Kelvin, no Celsius ni Fahrenheit.

Las descripciones del tiempo que quieres están después de "main" y 'description'. Para ordenadamente, añade lo siguiente a *quickWeather.py*.

```
python3
  # quickWeather.py - Imprime el tiempo de un lugar desde la línea de comandos.

  --snip--

  # Cargar datos JSON en una variable Python.
  weatherData= JSON.LOADS(response.text)
```

```
    # Imprimir DESCRIPCIONES meteorológicas.
❶ w= weatherData['list']
    print('Tiempo actual en %s:' % (ubicación)) print(w[0]['tiempo'][0]['principal'], '-',
    w[0]['tiempo'][0]['descripción']) print()
    print('Mañana:')
    print(w[1]['tiempo'][0]['principal'], '-', w[1]['tiempo'][0]['descripción']) print()
    print('Pasado mañana:')
    print(w[2]['weather'][0]['main'], '-', w[2]['weather'][0]['description'])
```

Observa cómo el código almacena `weatherData['list']` en la variable `w` para ahorrarte algo de tecleo❶. Utiliza `w[0]`, `w[1]` y `w[2]` para recuperar los diccionarios del tiempo de hoy, mañana y pasado mañana, respectivamente. Cada uno de estos diccionarios tiene una clave `'tiempo'`, que contiene un valor de lista. Te interesa el primer elemento de la lista, un diccionario anidado con varias claves más, en el índice 0. Aquí, imprimimos los valores almacenados en las claves `'main'` y `'description'`, separados por un guión.

Cuando este programa se ejecuta con el argumento de línea de comandos `quickWeather.py San Francisco, CA`, la salida tiene este aspecto:

```
Tiempo actual en San Francisco, CA:
Clear - cielo despejado

Mañana:
Nubes - pocas nubes

Pasado mañana:
Clear - cielo despejado
```

(El clima es una de las razones por las que me gusta vivir en San Francisco).

Ideas para programas similares

El acceso a los datos meteorológicos puede constituir la base de muchos tipos de programas. Puede crear programas similares para hacer lo siguiente:

- Recopila las previsiones meteorológicas de varios campings o rutas de senderismo para ver cuál tendrá el mejor tiempo.
- Programe un programa para que compruebe regularmente el tiempo y le envíe una alerta de heladas si necesita trasladar sus plantas al interior. (El capítulo 15 trata de la programación y el capítulo 16 explica cómo enviar correos electrónicos).
- Extraiga datos meteorológicos de varios sitios para mostrarlos todos a la vez, o calcule y muestre la media de las múltiples predicciones meteorológicas.

Resumen

CSV y JSON son formatos comunes de texto plano para almacenar datos. Son fáciles de analizar para los programas y legibles para las personas, por lo que se suelen utilizar para hojas de cálculo sencillas o datos de aplicaciones web. Los módulos `csv` y `json` simplifican enormemente el proceso de lectura y escritura en archivos CSV y JSON.

Los últimos capítulos te han enseñado cómo usar Python para analizar información de una amplia variedad de formatos de archivo. Una tarea común es tomar datos de una variedad de formatos y analizarlos para obtener la información particular que necesitas. Estas tareas son a menudo tan específicas que el software comercial no es de gran ayuda. Escribiendo tus propios scripts, puedes hacer que el ordenador maneje grandes cantidades de datos presentados en estos formatos.

En el capítulo 15, romperás con los formatos de datos y aprenderás a hacer que tus programas se comuniquen contigo enviándote correos electrónicos y mensajes de texto.

Preguntas prácticas

P: 1. ¿Qué características tienen las hojas de cálculo Excel que no tienen las hojas de cálculo CSV?

P: 2. ¿Qué le pasas a `csv.reader()` y `csv.writer()` para crear los objetos `Reader` y `Writer`?

P: 3. ¿En qué modos deben abrirse los objetos `Archivo` de los objetos Lector y `Escritor`?

P: 4. ¿Qué método toma como argumento una lista y la escribe en un archivo CSV?

P: 5. ¿Qué hacen los argumentos de palabra clave `delimiter` y `lineterminator`?

P: 6. ¿Qué función toma una cadena de datos JSON y devuelve una estructura de datos Python?

P: 7. ¿Qué función toma una estructura de datos Python y devuelve una cadena de datos JSON?

Proyecto práctico

Para practicar, escribe un programa que haga lo siguiente.

Conversor de Excel a CSV

Excel puede guardar una hoja de cálculo en un archivo CSV con unos pocos clics del ratón, pero si tuvieras que convertir cientos de archivos Excel en CSV, te llevaría horas de clics. Utilizando el módulo openpyxl del Capítulo 12, escriba un programa que lea todos los archivos Excel en directorio de trabajo actual y los convierta en archivos CSV.

Un único archivo Excel puede contener varias hojas; tendrá que crear un archivo CSV por *hoja*. Los nombres de archivo de los archivos CSV deben ser *<nombre_archivo_excel>_<título_hoja>.csv*, donde *<nombre_archivo_excel>* es el nombre del archivo Excel sin la extensión (por ejemplo, "datos_spam", no "datos_spam.xlsx") y *<título_hoja>* es la cadena de la variable de título del objeto Hoja de cálculo.

Este programa incluirá muchos bucles for anidados. El esqueleto del programa será algo como esto:

```
para excelArchivo en os.listdir('.'):
    # Omitir archivos que no sean xlsx, cargar el objeto
    libro de trabajo. for sheetName in
    wb.get_sheet_names():
        # Recorrer todas las hojas del libro. hoja=
        wb.get_sheet_by_name(sheetName)

        # Crear el nombre del fichero CSV a partir del nombre del fichero Excel y
        el título de la hoja. # Crear el objeto csv.writer para este archivo
        CSV.

        # Recorre todas las filas de la hoja.
        for rowNum in range(1, sheet.get_highest_row()+ 1):
            rowData= []         # Añadir cada celda a esta lista
            # Recorrer cada celda de la fila.
            for colNum in range(1, sheet.get_highest_column()+ 1): # Añade
                los datos de cada celda a rowData.

            # Escribir la lista rowData en el archivo CSV.

        csvFile.close()
```

Descargue el archivo ZIP *excelSpreadsheets.zip* de http://nostarch.com/automatestuff/, y descomprima las hojas de cálculo en el mismo directorio que su programa. Puede utilizar estos archivos para el programa.

Capítulo 15. Llevar la cuenta del tiempo, programar tareas e iniciar programas

Ejecutar programas mientras estás sentado frente al ordenador está bien, pero también es útil que los programas se ejecuten sin tu supervisión directa. El reloj de tu ordenador puede programar programas para que ejecuten código a una hora y fecha determinadas o a intervalos regulares. Por ejemplo, tu programa podría rastrear un sitio web cada hora para comprobar si hay cambios o realizar una tarea que consuma mucha CPU a las 4 de la mañana mientras duermes.

Los módulos time y datetime de Python proporcionan estas funciones.

También puede escribir programas que lancen otros programas según un calendario utilizando los módulos de subprocesos e hilos. A menudo, la forma más rápida de programar es aprovechar las aplicaciones que ya han escrito otras personas.

El módulo de tiempo

El reloj del sistema de tu ordenador está ajustado a una fecha, hora y zona horaria específicas. El módulo de tiempo incorporado permite a tus programas Python leer el reloj del sistema para conocer la hora actual. Las funciones time.time() y time.sleep() son las más útiles del módulo tiempo.

La función time.time()

La época Unix es una referencia temporal utilizada habitualmente en programación: las 12 AM del 1 de enero de 1970, Tiempo Universal Coordinado (UTC). La función `time.time()` devuelve el número de segundos transcurridos desde ese momento como un valor float. (Recuerde que un float es simplemente un número con un punto decimal.) Este número se denomina *epoch timestamp*. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> tiempo de importación
>>> time.time()
1425063955.068649
```

Aquí estoy llamando a `time.time()` el 27 de febrero de 2015, a las 11:05 hora estándar del Pacífico, o 7:05 PM UTC. El valor de retorno es cuántos segundos han pasado entre la época Unix y el momento en que se llamó a `time.time()`.

> **NOTA**
>
> *Los ejemplos del shell interactivo darán fechas y horas de cuando escribí este capítulo en febrero de 2015. A menos que seas un viajero en el tiempo, tus fechas y horas serán diferentes.*

Las marcas de tiempo de época se pueden utilizar para *perfilar* código, es decir, medir cuánto tiempo tarda en ejecutarse un fragmento de código. Si llama a `time.time()` al principio del bloque de código que desea medir y de nuevo al final, puede restar la primera marca de tiempo de la segunda para hallar el tiempo transcurrido entre esas dos llamadas. Por ejemplo, abra una nueva ventana del editor de archivos e introduzca el siguiente programa:

```
tiempo de importación
❶ def calcProd():
        # Calcula el producto de los 100.000 primeros números.
        product= 1
        for i in range(1, 100000):
            product= product * i
        devolver producto

❷ startTime= time.time() prod=
    calcProd()
➲   endTime= time.time()
❹ print('El resultado tiene %s dígitos.' % (len(str(prod))))
❺ print('Tardó %s segundos en calcularse.' % (endTime - startTime))
```

En definimos una función `calcProd()` para recorrer los enteros de 1 a 99.999 y devolver su producto. ❷, llamamos a `time.time()` y lo almacenamos en

startTime. Justo después de llamar a calcProd(), volvemos a llamar a time.time() y lo almacenamos en endTime❸. Terminamos imprimiendo la longitud del producto devuelto por calcProd() ❹ y cuánto tiempo tardó en ejecutarse calcProd()❺.

Guarde este programa como *calcProd.py* y ejecútelo. El resultado será algo parecido a esto:

```
El resultado  456569 dígitos.
Tardó 2,844162940979004 segundos en calcularse.
```

NOTA

Otra forma de perfilar su código es utilizar la `cProfile.run()`*, que proporciona un nivel de detalle mucho más informativo que la simple técnica* `time.time()`*. La función* `cProfile.run()` *se explica en* https://docs.python.org/3/library/profile.html.

La función time.sleep()

Si necesitas pausar tu programa por un , llama a la función time.sleep() y pásale el número de segundos que quieres que tu programa permanezca pausado. Introduce lo siguiente en el intérprete de comandos interactivo:

```
>>> tiempo de importación
    >>> para i en rango(3):
❶           print('Tic')
❷           time.sleep(1)
➲           print('Tock')
❹           time.sleep(1)
    Tick
    Tock
    Tick
    Tock
    Tick
    Tock
>>> ❺time.sleep(5)
```

El bucle for imprimirá Tick❶, hará una pausa de un segundo❷, imprimirá Tock❸, hará una pausa de un segundo❹, imprimirá Tick, hará una pausa, y así sucesivamente hasta que Tick y Tock se hayan impreso tres veces cada uno.

La función time.sleep() *se* decir, no volverá y liberará tu programa para ejecutar otro código-hasta que haya transcurrido el número de segundos que le pasaste a time.sleep(). Por ejemplo, si introduces time.sleep(5)

❺, verás que la siguiente indicación (>>>) no aparece hasta que han pasado cinco segundos.

Ten en cuenta que pulsar CTRL-C no interrumpirá las llamadas a time.sleep() en IDLE. IDLE espera hasta que la pausa completa haya terminado antes de lanzar la excepción KeyboardInterrupt. Para solucionar este problema, en lugar de hacer una sola llamada a time.sleep(30) para hacer una pausa de 30 segundos, utilice un bucle for para hacer 30 llamadas a time.sleep(1).

```
>>> for i in range(30):
        time.sleep(1)
```

Si pulsa CTRL-C en algún momento durante estos 30 segundos, debería ver el icono Lanzamiento de la excepción KeyboardInterrupt.

Redondear números

Cuando trabajes con tiempos, a menudo encontrarás valores flotantes con muchos dígitos después del decimal. Para que sea más fácil trabajar con estos valores, puedes acortarlos con la función integrada round() de Python, que redondea un flotante a la precisión que especifiques. Sólo tiene que introducir el número que desea redondear, más un segundo argumento opcional que representa cuántos dígitos después del punto decimal desea redondear. Si omite el segundo argumento, round() redondea el número al entero más próximo. Introduzca lo siguiente en el intérprete de órdenes interactivo:

```
>>> tiempo de importación
>>> now= time.time()
>>> ahora
1425064108.017826
>>> round(ahora, 2)
1425064108.02
>>> round(ahora, 4)
1425064108.0178
>>> redondo(ahora)
1425064108
```

Después de importar time y almacenar time.time() en now, llamamos a round(now, 2) para redondear now a dos dígitos después del decimal, round(now, 4) para redondear a cuatro dígitos después del decimal, y round(now) para redondear al entero más cercano.

Proyecto Super Cronómetro

Supongamos que quieres controlar cuánto tiempo dedicas a tareas aburridas que aún no has automatizado. No tienes un cronómetro físico, y es sorprendentemente difícil encontrar una aplicación de cronómetro gratuita para tu portátil o smartphone que no esté cubierta de anuncios y no envíe una copia de tu historial de navegación a los vendedores. (Dice que puede hacer esto en el acuerdo de licencia que aceptaste. Leíste el acuerdo de licencia, ¿verdad?) Tú mismo puedes escribir un sencillo programa de cronómetro en Python.

A grandes , esto es lo que hará tu programa:

- Registra el tiempo transcurrido entre las pulsaciones de la tecla ENTER, y cada pulsación inicia una nueva "vuelta" en el temporizador.
- Imprime el número de vuelta, el tiempo total y el tiempo de vuelta.

Esto significa que su código tendrá que hacer lo siguiente:

- Encuentra la hora actual llamando a `time.time()` y guárdala como una marca de tiempo al inicio del programa, así como al inicio de cada vuelta.
- Mantenga un contador de vueltas e increméntelo cada vez que el usuario pulse ENTER.
- Calcula el tiempo transcurrido restando las marcas de tiempo.
- Maneja la excepción `KeyboardInterrupt` para que el usuario pueda presionar CTRL-C para salir.

Abre una nueva ventana del editor de archivos y guárdalo como *cronómetro.py*.

Paso 1: Configurar el programa para controlar los tiempos

El programa cronómetro necesitará usar la hora actual, por lo que importar el módulo `time`. Su programa también debería imprimir algunas instrucciones breves al usuario antes de llamar a `input()`, para que el cronómetro pueda comenzar después de que el usuario presione ENTER. Entonces el código empezará a registrar los tiempos de vuelta.

Introduzca el siguiente código en el editor de archivos, escribiendo un comentario `TODO` como marcador de posición para el resto del código:

```
#! python3
# cronómetro.py - Un programa simple de cronómetro.

import tiempo

# Mostrar las instrucciones del programa.
print('Pulse ENTER para comenzar. Después, pulse ENTER para "pulsar" el cronómetro.
Pulse Ctrl-C para salir.')
input()                         # pulse Enter para
comenzar print('Iniciado.')
startTime= time.time()      # obtener la hora de inicio de
la primera vuelta lastTime= startTime
lapNum= 1

# TODO: Iniciar el seguimiento de los tiempos de vuelta.
```

Ahora que hemos escrito el código para mostrar las instrucciones, comienza la primera vuelta, anota el tiempo y establece nuestro contador de vueltas en 1.

Paso 2: Controlar e imprimir los tiempos de vuelta

Ahora vamos a escribir el código para iniciar cada nueva vuelta, calcular cuánto tiempo duró la vuelta anterior, y calcular el tiempo total transcurrido desde el inicio del cronómetro. Mostraremos el tiempo de la vuelta y el tiempo total e incrementaremos el contador de vueltas para cada nueva vuelta. Añade el siguiente código a tu programa:

```
#! python3
    # cronómetro.py - Un programa simple de cronómetro.

    import tiempo

    --snip--

    # Comienza el seguimiento de los TIEMPOS DE VUELTA.
❶ Inténtalo:
❷       while True:
                entrad
                a()
➲               lapTime= round(time.time() - lastTime, 2)
❹               totalTime= round(time.time() - startTime, 2)
❺               print('Vuelta #%s: %s (%s)' % (lapNum, totalTime, lapTime), end='')
                lapNum+= 1
                lastTime= time.time() # restablecer el tiempo de la última vuelta
❻ excepto KeyboardInterrupt:
        # Maneja la excepción Ctrl-C para evitar que se muestre SU mensaje de error. print('\nHecho.')
```

Si el usuario pulsa CTRL-C para detener el cronómetro, se lanzará la excepción KeyboardInterrupt, y el programa se bloqueará si su ejecución no es una sentencia try. Para , envolvemos esta parte del programa en una sentencia try

sentencia ❶. Manejaremos la excepción en la cláusula except ❻, de forma que cuando se pulse CTRL- C y se produzca la excepción, la ejecución del programa se mueva a la cláusula except para imprimir Hecho, en lugar del mensaje de error KeyboardInterrupt. Hasta que esto ocurra, la ejecución está dentro de un bucle infinito ❷ que llama a input() y espera hasta que el usuario pulsa ENTER para terminar una vuelta. Cuando termina una vuelta calculamos cuánto tiempo ha durado la vuelta restando la hora de inicio de la vuelta, lastTime, de la hora actual, time.time()❸. Calculamos el tiempo total transcurrido

restando la hora de inicio global del cronómetro, startTime, de la hora actual❹.

Dado que los resultados de estos cálculos de tiempo tendrán muchos dígitos después del punto decimal (como 4.766272783279419), utilizamos la función round() para

redondea el valor float a dos dígitos en ❸ y❹.

En❺, imprimimos el número de vuelta, el tiempo total transcurrido y el tiempo de vuelta.

Como el usuario que pulsa ENTER para la llamada `input()` imprimirá una nueva línea en la pantalla, pasa `end=''` a la función `print()` para evitar el doble espacio en la salida. Después de imprimir la información de la vuelta, nos preparamos para la siguiente vuelta añadiendo 1 a la cuenta `lapNum` y ajustando `lastTime` a la hora actual, que es la hora de inicio de la siguiente vuelta.

Ideas para programas similares

El seguimiento del tiempo abre varias posibilidades para tus programas. Aunque puedes descargarte aplicaciones para hacer algunas de estas cosas, la ventaja de escribir programas tú mismo es que serán gratuitos y no estarán hinchados de anuncios y funciones inútiles. Podrías escribir programas similares para hacer lo siguiente:

- Cree una sencilla aplicación de hoja de horas que registre el nombre de una persona y utilice la hora actual para ficharla a la entrada o a la salida.
- Añada una función a su programa para mostrar el tiempo transcurrido desde que se inició un proceso, como una descarga que utiliza el módulo de peticiones. (Véase el Capítulo 11.)
- Compruebe de forma intermitente cuánto tiempo lleva ejecutándose un programa y ofrezca al usuario posibilidad de cancelar las tareas que estén tardando demasiado.

El módulo datetime

El módulo time es útil para obtener una marca de tiempo de época Unix con la que trabajar. Pero si desea mostrar una fecha en un formato más conveniente, o hacer aritmética con fechas (por ejemplo, averiguar qué fecha fue hace 205 días o qué fecha es dentro de 123 días), debe utilizar el datetime.

El módulo datetime tiene su propio tipo de datos datetime. Los valores datetime representan un momento específico en el tiempo. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar datetime
>>> ❶datetime.datetime.now()
❷ datetime.datetime(2015, 2, 27, 11, 10, 49, 55, 53)
➲  >>> dt= datetime.datetime(2015, 10, 21, 16, 29, 0)
>>> ❹dt.año, dt.mes, dt.día
    (2015, 10, 21)
>>> ❺dt.hora, dt.minuto, dt.segundo
    (16, 29, 0)
```

Llamar a datetime.datetime.now() ❶ devuelve un objeto datetime ❷ para la fecha y hora actuales, según el reloj de tu ordenador. Este objeto incluye el año, mes, día, hora, minuto, segundo y microsegundo del momento actual. También puedes recuperar un objeto datetime para un momento específico utilizando la función datetime.datetime()❸, pasándole enteros que representen el año, mes, día, hora y segundo del momento que desees. Estos enteros se almacenarán en los atributos año, mes, día ❹, hora, minuto y segundo ❺ del objeto datetime.

Una marca de tiempo de época Unix puede convertirse en un objeto datetime con la función datetime.datetime.fromtimestamp(). La fecha y la hora del objeto datetime se convertirán para la zona horaria local. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> datetime.datetime.FROMTIMESTAMP(1000000)
datetime.datetime(1970, 1, 12, 5, 46, 40)
>>> datetime.datetime.FROMTIMESTAMP(time.time())
datetime.datetime(2015, 2, 27, 11, 13, 0, 604980)
```

Llamar a datetime.datetime.fromtimestamp() y pasarle 1000000 devuelve un objeto datetime para el momento 1.000.000 de segundos después de la época Unix. Pasando time.time(), la marca de tiempo de época Unix para el momento actual,

devuelve un objeto datetime para el momento actual. Así, las expresiones datetime.datetime.now() y datetime.datetime.fromtimestamp(time.time()) hacen lo mismo; ambas devuelven un objeto datetime para el momento actual.

> **NOTA**
>
> *Estos ejemplos se introdujeron en un ordenador configurado con la hora estándar del Pacífico. Si se encuentra en otra zona horaria, los resultados serán diferentes.*

Los objetos datetime pueden compararse entre sí utilizando operadores de comparación para averiguar cuál precede al otro. El objeto datetime posterior es el valor "mayor". Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> ❶halloween2015= datetime.datetime(2015, 10, 31, 0, 0, 0)

>>> ❷newyears2016= datetime.datetime(2016, 1, 1, 0, 0, 0)
    >>> oct31_2015= datetime.datetime(2015, 10, 31, 0, 0, 0)
➲   >>> halloween2015== oct31_2015
    Verdadero

>>> ❹halloween2015> año nuevo2016
    Falso

>>> ❺año nuevo2016> halloween2015
    Verdadero
    >>> newyears2016 != oct31_2015
    Verdadero
```

Crea un objeto datetime para el primer momento (medianoche) del 31 de octubre de 2015 y guárdalo en halloween2015 ❶. Haz un objeto datetime para el primer momento del 1 de enero de 2016 y guárdalo en newyears2016 ❷. Luego haz otro objeto para la medianoche del 31 de octubre de 2015 y almacénalo en oct31_2015. Comparar halloween2015 y oct31_2015 muestra que son iguales ❸. Comparando newyears2016 y halloween2015 muestra que newyears2016 es mayor (posterior) que halloween2015 ❹❺.

Tipo de datos timedelta

El módulo datetime también proporciona un tipo de datos timedelta, que representa una *duración* de tiempo en lugar de un *momento* en el tiempo. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> ❶delta= datetime.timedelta(DAYS=11, HOURS=10, MINUTES=9, SECONDS=8)

>>> ❷delta.DÍAS, delta.SEGUNDOS, delta.MICROSEGUNDOS
    (11, 36548, 0)
    >>> delta.TOTAL_SEGUNDOS()
    986948.0
    >>> str(delta)
    '11 días, 10:09:08'
```

Para crear un objeto timedelta, utilice la función datetime.timedelta(). La función datetime.timedelta() toma como argumentos palabras clave semanas, días, horas, minutos, segundos, milisegundos y microsegundos. No hay argumento de palabra clave mes o año porque "un mes" o "un año" es una variable.
cantidad de tiempo en función del mes o año concreto. Un objeto timedelta tiene la duración total representada en días, segundos y microsegundos. Estos números se almacenan en los atributos días, segundos y microsegundos, respectivamente. El método total_seconds() devolverá la duración sólo en número de segundos. Si se pasa un objeto timedelta a str(), se obtendrá una representación del objeto en forma de cadena legible por el ser humano.

En este , pasamos argumentos de palabra clave a datetime.delta() para especificar una duración de 11 días, 10 horas, 9 minutos y 8 segundos, y almacenamos el objeto timedelta devuelto en delta❶. El atributo días de este objeto timedelta almacena 11, y su atributo segundos almacena 36548 (10 horas, 9 minutos y 8 segundos, expresados en segundos)❷. Llamando a total_seconds() nos dice que 11 días, 10 horas, 9 minutos y 8 segundos son 986.948 segundos. Por último, al pasar el objeto timedelta a str() se obtiene una cadena que explica claramente la duración.

Los operadores aritméticos pueden utilizarse para realizar *operaciones aritméticas* con valores fecha-hora. Por ejemplo, para calcular la fecha dentro de 1.000 días, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> dt= datetime.datetime.now()
>>> dt
datetime.datetime(2015, 2, 27, 18, 38, 50, 636181)
>>> THOUSANDDAYS= datetime.timedelta(DAYS=1000)
```

```
>>> dt+ milDías
datetime.datetime(2017, 11, 23, 18, 38, 50, 636181)
```

Primero, crea un objeto datetime para el momento actual y guárdalo en dt. A continuación, crea un objeto timedelta para una duración de 1.000 días y guárdalo en milDías. Suma dt y milDías para obtener un objeto datetime para la fecha de dentro de 1.000 . Python hará la aritmética de fechas para calcular que 1.000 días después del 27 de febrero de 2015 será el 23 de noviembre de 2017. Esto es útil porque cuando calculas 1.000 días a partir de una fecha dada, tienes que recordar cuántos días hay en cada mes y tener en cuenta los años bisiestos y otros detalles complicados. El módulo datetime se encarga de todo esto.

Los objetos timedelta pueden sumarse o restarse con objetos datetime u otros objetos timedelta mediante los operadores+ y -. Un objeto timedelta multiplicarse o dividirse por valores enteros o flotantes con los * y /. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> ❶oct21st= datetime.datetime(2015, 10, 21, 16, 29, 0)

>>> ❷aboutThirtyYears= datetime.timedelta(days=365 * 30)
    >>> 21 de octubre
    datetime.datetime(2015, 10, 21, 16, 29)
    >>> oct21st - aboutThirtyYears
    datetime.datetime(1985, 10, 28, 16, 29)
    >>> oct21st - (2 * aboutThirtyYears)
    datetime.datetime(1955, 11, 5, 16, 29)
```

Aquí hacemos un objeto datetime para el 21 de octubre de 2015 ❶ y un objeto timedelta para una duración de unos 30 años (estamos asumiendo 365 días para cada uno de esos años)❷. Restando aboutThirtyYears de oct21st nos da un objeto datetime para la fecha 30 años antes del 21 de octubre de 2015. Restando 2 * aboutThirtyYears from oct21st devuelve un objeto datetime para la fecha 60 años antes del 21 de octubre de 2015.

Pausa hasta una fecha concreta

El método time.sleep() te permite pausar un programa durante un cierto número de segundos. Usando un bucle while, puedes pausar tus programas hasta una fecha específica. Por ejemplo, el siguiente código continuará el bucle hasta Halloween de 2016:

```
import datetime
import time
halloween2016= datetime.datetime(2016, 10, 31, 0, 0, 0) while
datetime.datetime.now()< halloween2016:
    time.sleep(1)
```

La llamada time.sleep(1) pausará tu programa Python para que el ordenador no malgaste ciclos de procesamiento de la CPU comprobando la hora una y otra vez. En su lugar, el bucle while sólo comprobará la condición una vez por segundo y continuará con el resto del programa después de Halloween 2016 (o cuando programes que se detenga).

Conversión de objetos datetime en cadenas

Las marcas de tiempo y los objetos datetime no son muy amigables para el ojo humano. Utilice el método strftime() para mostrar un objeto datetime como una cadena. (La *f* en el nombre de la función strftime() significa *formato*).

El método strftime() utiliza directivas similares al formateo de cadenas de Python. La Tabla 15-1 tiene una lista completa de las directivas de strftime().

Tabla 15-1. Directivas strftime()

directiva strftime	Significado
%Y	Año con siglo, como en "2014
%y	Año sin siglo, de "00" a "99" (de 1970 a 2069)
%m	Mes como número decimal, de "01" a "12
%B	Nombre completo del mes, como ""noviembre""
%b	Nombre abreviado del mes, como "Nov".
%d	Día del mes, del "01" al "31".
%j	Día del año, del "001" al "366".
%w	Día de la , de "0" (domingo) a "6" (sábado)
%A	Nombre completo del día de la semana, como ""lunes""
%a	Nombre abreviado del día de la semana, como ""Lun""
%H	Hora (reloj de 24 horas), de "00" a "23".
%I	Hora (reloj de 12 horas), de "01" a "12
%M	Minuto, de '00' a '59'
%S	Segundo, de '00' a '59
%p	AM o PM
%%	Carácter literal ""%

Pase a strrftime() una cadena de formato personalizada que contenga directivas de formato (junto con las barras oblicuas, dos puntos, etc. que desee), y strftime() devolverá la información del objeto datetime como una cadena formateada. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> oct21st= datetime.datetime(2015, 10, 21, 16, 29, 0)
>>> oct21st.strftime('%Y/%m/%d %H:%M:%S')
'2015/10/21 16:29:00'
>>> oct21st.strftime('%I:%M %p')
'04:29 PM'
>>> oct21st.strftime("%B de '%y")
"Octubre 15"
```

Aquí tenemos un objeto datetime para el 21 de octubre de 2015 a las 16:29, almacenado en oct21st. Pasando a strftime() la cadena de formato personalizado '%Y/%m/%d %H:%M:%S' devuelve una cadena que contiene 2015, 10, y 21 separados por slahes y 16, 29, y 00 separados por dos puntos. Pasando '%I:%M%p' devuelve '04:29 PM', y pasando "%B of '%y" devuelve "October of '15". Tenga en cuenta que strftime() no comienza
con datetime.datetime.

Conversión de cadenas en objetos datetime

Si tiene una cadena de información de fecha, como '2015/10/21 16:29:00' o '21 de octubre de 2015', y necesita convertirla en un objeto datetime, utilice la función datetime.datetime.strptime(). La función strptime() es la inversa del método strftime(). Se debe pasar una cadena de formato personalizada que utilice las mismas directivas que strftime() para que strptime() sepa cómo analizar y comprender la cadena. (La *p* en el nombre de la función strptime() significa *parse*).

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> ❶datetime.datetime.strptime('21 de octubre de 2015', '%B %d, %Y')
    datetime.datetime(2015, 10, 21, 0, 0)
    >>> datetime.datetime.strptime('2015/10/21 16:29:00', '%Y/%m/%d %H:%M:%S')
    datetime.datetime(2015, 10, 21, 16, 29)
    >>> datetime.datetime.strptime("Octubre del '15", "%B de '%y")
    datetime.datetime(2015, 10, 1, 0, 0)
    >>> datetime.datetime.strptime("Noviembre del 63", "%B de '%y")
    datetime.datetime(2063, 11, 1, 0, 0)
```

Para obtener un objeto datetime a partir de la cadena '21 de octubre de 2015', pase '21 de octubre de 2015' como primer argumento a strptime() y la cadena de formato personalizado que corresponde a '21 de octubre de 2015' como segundo argumento❶. La cadena con la información de la fecha debe coincidir exactamente con la cadena de formato personalizado, o Python lanzará una excepción ValueError.

Revisión de las funciones temporales de Python

Las fechas y horas en Python pueden implicar bastantes tipos de datos y funciones diferentes. He aquí un repaso de los tres tipos diferentes de valores utilizados para representar el tiempo:

- Una marca de tiempo de época Unix (utilizada por el módulo de tiempo) es un valor flotante o entero del número de segundos transcurridos desde las 12 AM del 1 de enero de 1970, UTC.
- Un objeto datetime (del módulo datetime) tiene enteros almacenados en los atributos año, mes, día, hora, minuto y segundo.
- Un objeto timedelta (del módulo datetime) representa una duración de tiempo, en lugar de un momento concreto.

A continuación repasamos las funciones temporales y sus parámetros y valores de retorno:

- La función time.time() devuelve un valor float epoch timestamp del momento actual.
- La función time.sleep(*seconds*) detiene el programa durante la cantidad de segundos especificada por el argumento *seconds*.
- La función datetime.datetime(*año*, *mes*, *día*, *hora*, *minuto*, *segundo*) devuelve un objeto datetime del momento especificado por los argumentos. Si no se proporcionan los argumentos de *hora*, *minuto* o *segundo*, por defecto serán 0.
- La función datetime.datetime.now() devuelve un objeto datetime del momento actual.
- La función datetime.datetime.fromtimestamp(*epoch*) devuelve un objeto datetime del momento representado por el argumento *epoch* timestamp.
- La función datetime.timedelta(*semanas*, *días*, *horas*, *minutos*, *segundos*, *milisegundos*, *microsegundos*) devuelve un objeto timedelta que representa una duración de tiempo. Los argumentos de la función son todos opcionales y no incluyen el *mes* ni el *año*.
- El método total_seconds() para objetos timedelta devuelve el número de segundos que representa el objeto timedelta.

- El método strftime(*format*) devuelve una cadena con la hora representada por el objeto datetime en un formato personalizado basado en la cadena *format*. Consulte la Tabla 15-1 para conocer los detalles del formato.
- La función datetime.datetime.strptime(*time_string*, *format*) devuelve un objeto datetime del momento especificado por *time_string*, analizado utilizando el argumento *format* string. Consulte los detalles del formato en la Tabla 15-1.

Multihilo

Para introducir el concepto de multihilo, veamos una situación de ejemplo. Digamos que quieres programar algún código para que se ejecute después de un retraso o a una hora específica. Podrías añadir código como el siguiente al principio de tu programa:

```
import time, datetime

startTime= datetime.datetime(2029, 10, 31, 0, 0, 0) while
datetime.datetime.now()< startTime:
    time.sleep(1)

print('Programa ahora a partir de Halloween 2029')
--snip--
```

Este código designa una hora de inicio el 31 de octubre de 2029, y sigue llamando a `time.sleep(1)` hasta que llega la hora de inicio. Tu programa no puede hacer nada mientras espera a que termine el bucle de llamadas a `time.sleep()`; simplemente se queda hasta Halloween de 2029. Esto se debe a que los programas Python tienen por defecto un único *hilo* de ejecución.

Para entender lo que es un hilo de ejecución, recuerde la discusión del Capítulo 2 sobre el control de flujo, cuando imaginó la ejecución de un programa como si colocara su dedo en una línea de código de su programa y se moviera a la siguiente línea o a donde fuera enviado por una sentencia de control de flujo. Un programa *de un solo hilo* sólo tiene un dedo. Pero un *programa multihilo* tiene múltiples dedos. Cada dedo todavía se mueve a la siguiente línea de código como se define por las sentencias de control de flujo, pero los dedos pueden estar en diferentes lugares del programa, ejecutando diferentes líneas de código al mismo . (Todos los programas de este libro hasta ahora han sido un solo hilo).

En lugar de tener todo tu código esperando hasta que la función `time.sleep()` termine, puedes ejecutar el código retrasado o programado en un hilo separado usando el módulo de hilos de Python. El hilo separado hará una pausa para las llamadas a `time.sleep`. Mientras tanto, tu programa puede hacer otro trabajo en el hilo original.

Para crear un hilo independiente, primero necesitas crear un objeto `Thread` llamando a la función `threading.Thread()`. Introduce el siguiente código en un nuevo archivo y guárdalo como *threadDemo.py*:

```
import threading, time
    print('Inicio del programa.")
```

```
❶ def takeANap():
        time.sleep(5)
        print('!')

❷ threadObj= threading.Thread(target=takeANap)
➲  threadObj.start()

    print('Fin del programa.')
```

En❶, definimos una función que queremos utilizar en un nuevo hilo. Para crear un objeto Thread, llamamos a threading.Thread() y le pasamos el argumento de palabra clave target=takeANap❷. Esto significa que la función que queremos llamar en el nuevo hilo es takeANap(). Observe que el argumento clave es target=takeANap, no target=takeANap(). Esto es porque quieres pasar la propia función takeANap() como argumento, no llamar a takeANap() y pasar su valor de retorno.

Después de almacenar el objeto Thread creado por threading.Thread() en threadObj, llamamos a threadObj.start() ❸ para crear el nuevo hilo y comenzar a ejecutar la función objetivo en el nuevo hilo. Cuando este programa se ejecuta, la salida se verá así:

```
Inicio del programa.
Fin del programa.
!
```

Esto puede ser un poco confuso. Si print('Fin del programa.') es la última línea del programa, podrías pensar que debería ser la última cosa impresa. La razón por la que Wake up! viene después es que cuando se llama a threadObj.start(), la función de destino para threadObj se ejecuta en un nuevo hilo de ejecución. Piensa en ello como un segundo dedo que aparece al comienzo de la takeANap(). El hilo principal continúa imprimiendo('Fin del programa.'). Mientras tanto, el nuevo hilo que ha estado ejecutando la llamada time.sleep(5), hace una pausa de 5 segundos. Después de despertarse de su siesta de 5 segundos, imprime '!' y luego vuelve de la función takeANap(). Cronológicamente, ''!" es lo último que el programa.

Normalmente un programa termina cuando se ha ejecutado la última línea de código del archivo (o se llama a la función sys.exit()). *Pero threadDemo.py* tiene dos hilos. El primero es el hilo original que comenzó al principio del programa y termina después de print('Fin del programa.'). El segundo hilo se crea cuando se llama a threadObj.start(), comienza al inicio de la takeANap(),

y finaliza cuando vuelve takeANap().

Un programa Python no terminará hasta que todos sus hilos hayan terminado. Cuando ejecutó *threadDemo.py*, aunque el hilo original había terminado, el segundo hilo seguía ejecutando la llamada time.sleep(5).

Pasar argumentos a la función de destino del subproceso

Si la función de destino que desea ejecutar en el nuevo hilo toma argumentos, puede pasar los argumentos de la función de destino a threading.Thread(). Por ejemplo, digamos que desea ejecutar esta llamada print() en su propio hilo:

```
>>> print('Gatos', 'Perros', 'Ranas', sep=' & ')
Gatos, perros y ranas
```

Esta llamada a print() tiene tres argumentos regulares, 'Gatos', 'Perros' y 'Ranas', y un argumento de palabra clave, sep=' & '. Los argumentos regulares se pueden pasar como una lista al argumento de palabra clave args en threading.Thread(). El argumento de palabra clave puede especificarse como un diccionario al argumento de palabra clave kwargs en threading.Thread().

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar roscado
>>> threadObj= threading.Thread(target=print, args=['Gatos', 'Perros', 'Ranas'], kwargs={'sep': ' & '})
>>> threadObj.start()
Gatos, perros y ranas
```

Para asegurarnos de que los argumentos 'Gatos', Perros' y 'Ranas' se pasan a print() en el nuevo hilo, pasamos args=['Gatos', 'Perros', 'Ranas'] a threading.Thread(). Para asegurarse de que el argumento de palabra clave sep=' & ' se pasa a print() en el nuevo hilo, pasamos kwargs={'sep': '& '} a threading.Thread().

La llamada a threadObj.start() creará un nuevo hilo para llamar a la función print(), y pasará 'Gatos', 'Perros', y 'Ranas' como argumentos y ' & ' para el argumento de la palabra clave sep.

Esta es una forma incorrecta de crear el nuevo hilo que llama a print():

```
threadObj= threading.Thread(target=print('Gatos', 'Perros', 'Ranas', sep=' & '))
```

Lo que esto acaba haciendo es llamar a la función print() y pasar su valor de retorno (el valor de retorno de print() es siempre Ninguno) como argumento de la palabra clave de destino. *No* pasa la propia función print(). Cuando pases argumentos a una función en un nuevo hilo, usa los argumentos de la función threading.Thread() y

kwargs argumentos de palabra clave.

Problemas de concurrencia

Puedes crear fácilmente varios hilos nuevos y tenerlos todos funcionando al mismo tiempo. Pero múltiples hilos también pueden causar problemas llamados *problemas de concurrencia*.
Estos problemas ocurren cuando los hilos leen y escriben variables al mismo tiempo, haciendo que los hilos se tropiecen entre sí. Los problemas de concurrencia pueden ser difíciles de reproducir de forma consistente, lo que los hace difíciles de depurar.

La programación multihilo es un tema muy amplio y va más allá del alcance de este libro. Lo que debes tener en cuenta es lo siguiente: Para evitar problemas de concurrencia, nunca dejes que varios hilos lean o escriban las mismas variables. Cuando crees un nuevo objeto `Thread`, asegúrate de que su función de destino utiliza sólo variables locales en esa función. Esto evitará problemas de concurrencia difíciles de depurar en tus programas.

> **NOTA**
>
> *En* http://nostarch.com/automatestuff/ *encontrará un tutorial para principiantes sobre programación multihilo.*

Proyecto: Descargador XKCD multihilo

En el Capítulo 11, escribiste un programa que descargaba todas las tiras cómicas de XKCD de la página web de XKCD. Este era un programa de un solo hilo: Descargaba un cómic a la vez. Gran parte del tiempo de ejecución del programa se dedicaba a establecer la conexión de red para iniciar la descarga y a escribir las imágenes descargadas en el disco duro. Si tienes una conexión a Internet de banda ancha, tu programa monohilo no estaba utilizando plenamente el ancho de banda disponible.

Un programa multihilo que tiene algunos hilos descargando cómics mientras otros están estableciendo conexiones y escribiendo los archivos de imagen de los cómics en el disco utiliza tu conexión a Internet de forma más eficiente y descarga la colección de cómics más rápidamente. Abre una nueva ventana del editor de archivos y guárdalo como *multidownloadXkcd.py*. Modificarás este programa para añadir multithreading. El código fuente completamente modificado está disponible para su descarga en http://nostarch.com/automatestuff/.

Paso 1: Modificar el programa para utilizar una función

Este programa será en su mayor parte el mismo código de descarga del Capítulo 11, así me saltaré la explicación del código de Requests y BeautifulSoup. Los principales cambios que necesitas hacer son importar el módulo threading y hacer una función downloadXkcd(), que toma como parámetros los números de inicio y final del cómic.

Por ejemplo, si se llama a downloadXkcd(140, 280) se realizará un bucle sobre el código de descarga para descargar los cómics en http://xkcd.com/140, http://xkcd.com/141, http://xkcd.com/142, y así sucesivamente, hasta http://xkcd.com/279. Cada hilo que crees llamará a downloadXkcd() y pasará un rango diferente de cómics a descargar.

Añade el siguiente código a tu programa *multidownloadXkcd.py*:

```
#! python3
    # multidownloadXkcd.py - Descarga cómics XKCD usando múltiples hilos.

    importar peticiones, os, bs4, threading
s.makedirs("xkcd", exist_ok=True) # guarda los cómics en ./xkcd

❷ def descargarXkcd(inicioCómic, finCómic):
➲       for urlNumber in range(startComic, endComic): #
            Descarga la página.
            print('Descargando página http://xkcd.com/%s...' % (urlNúmero))
❹           res=
            requests.get('http://xkcd.com/%s'
            % (urlNumber)) res.raise_for_status()

❺           soup= bs4.BeautifulSoup(res.text) #

            Encuentra la URL de la imagen del

            comic.
❻           comicElem= soup.select("#comic img")
            si comicElem== []:
                print('No se ha encontrado la imagen del
            cómic.") else:
❼               comicUrl= comicElem[0].get("src") #
                Descarga la imagen.
                print('Descargando imagen %s...' % (comicUrl))
❽               res= requests.get(comicUrl)
                res.raise_for_status()

                # Guarda la imagen en ./xkcd.
                imageFile= open(os.path.join('xkcd', os.path.basename(comicUrl)),
wb")
```

```
            for chunk in res.iter_content(100000):
                imageFile.write(chunk)
            imageFile.close()

    # TODO: Crear e iniciar los objetos Thread. #
    TODO: Esperar a que todos los hilos
    terminen.
```

Después de importar los módulos que necesitamos, creamos un directorio para almacenar los cómics ❶ y empezamos a definir downloadxkcd() ❷. Hacemos un bucle a través de todos los números en el rango especificado ❸ y descargamos cada página ❹. Usamos Beautiful Soup para buscar en el HTML de cada página ❺ y encontrar la imagen del cómic ❻. Si no se encuentra ninguna imagen de cómic en una , imprimimos un mensaje. En caso contrario, obtenemos la URL de la
imagen ❼ y descargamos la imagen ❽. Por último, guardamos la imagen en el directorio que hemos creado.

Paso 2: Crear e iniciar hilos

Ahora que hemos definido downloadXkcd(), vamos a crear los múltiples hilos que cada uno llama a downloadXkcd() para descargar diferentes rangos de cómics desde el sitio web XKCD. Añade el siguiente código a *multidownloadXkcd.py* después de la definición de la función downloadXkcd():

```
#! python3
# multidownloadXkcd.py - Descarga cómics XKCD usando múltiples hilos.

--snip--

# Crear e iniciar los OBJETOS Thread.
DESCARGARHILOS= []                      # una lista de todos los OBJETOS
Thread for i in range(0, 1400, 100):  # BUCLES 14 VECES, CREA 14 HILOS
    downloadThread= threading.Thread(target=downloadXkcd, ARGS=(i, i+ 99))
    DOWNLOADTHREADS.append(downloadThread)
    downloadThread.start()
```

Primero hacemos una lista vacía downloadThreads; la lista nos ayudará a mantener un registro de los muchos objetos Thread que crearemos. A continuación, iniciamos nuestro bucle for. Cada vez que pasamos por el bucle, creamos un objeto Thread con threading.Thread(), añadimos el objeto Thread a la lista, y llamamos a start() para empezar a ejecutar downloadXkcd() en el nuevo hilo. Dado que el bucle for establece la variable i de 0 a 1400 en pasos de 100i se establecerá en 0 en la primera iteración, 100 en la segunda iteración, 200 en la tercera, y así sucesivamente. Como pasamos args=(i, i+ 99) a threading.Thread(), los dos argumentos pasados a downloadXkcd() serán 0 y 99 en la primera iteración, 100 y 199 en la segunda iteración, 200 y 299 en la tercera, etc.

Cuando el método start() del objeto Thread es llamado y el nuevo hilo comienza a ejecutar el código dentro de downloadXkcd(), el hilo principal continuará hasta la siguiente iteración del bucle for y creará el siguiente hilo.

Paso 3: Esperar a que finalicen todos los hilos

El hilo principal sigue su curso normal mientras los otros hilos que creamos descargan cómics. Pero digamos que hay algún código que no quieres ejecutar en el hilo principal hasta que todos los hilos hayan . Llamar al método `join()` de un objeto `Thread` bloqueará hasta que ese thread haya terminado. Usando un bucle `for` para iterar sobre todos los objetos `Thread` de la lista `downloadThreads`, el thread principal puede llamar al método
`join()` en cada uno de los otros hilos. Añade lo siguiente al final de tu programa:

```
#! python3
# multidownloadXkcd.py - Descarga cómics XKCD usando múltiples hilos.

--snip--

# Espera a que terminen todos los HILOS.
para downloadThread en DOWNLOADTHREADS:
    downloadThread.join()
print('Hecho.')
```

La cadena `"Listo"` no se imprimirá hasta que todas las llamadas a `join()` hayan retornado. Si un objeto `Thread` ya ha finalizado cuando su método `join()` es llamado, entonces el método simplemente retornará inmediatamente. Si quisieras extender este programa con código que se ejecute sólo después de que todos los cómics se hayan descargado, podrías reemplazar la línea `print('Done.')` con tu nuevo código.

Ejecutar otros programas desde Python

Tu programa Python puede iniciar otros programas en tu ordenador con la función Popen() del módulo incorporado de subprocesos. (La *P* en el nombre de la función Popen() significa *proceso*). Si tienes múltiples instancias de una aplicación abiertas, cada una de esas instancias es un proceso separado del mismo programa. Por ejemplo, si abre varias ventanas de su navegador web al mismo tiempo, cada una de esas ventanas es un proceso diferente del programa navegador web. Véase en la Figura 15-1 un ejemplo de varios procesos de calculadora abiertos a la vez.

Cada proceso puede tener varios hilos. A diferencia de los hilos, un proceso no puede leer y escribir directamente las variables de otro proceso. Si piensas en un programa multihilo como tener varios dedos siguiendo el código fuente, entonces tener varios procesos del mismo programa abiertos es como tener un amigo con una copia separada del código fuente del programa. Ambos están ejecutando independientemente el mismo programa.

Si quieres iniciar un programa externo desde tu script de Python, pasa el nombre de archivo del programa a subprocess.Popen(). (En Windows, haz clic con el botón derecho en el elemento del menú **Inicio** de la aplicación y selecciona **Propiedades** para ver el nombre de archivo de la aplicación. En OS X, haga CTRL-clic en la aplicación y seleccione **Mostrar contenido del paquete** para encontrar la ruta al archivo ejecutable). La función Popen() volverá inmediatamente. Tenga en cuenta que el programa lanzado no se ejecuta en el mismo hilo que su programa Python.

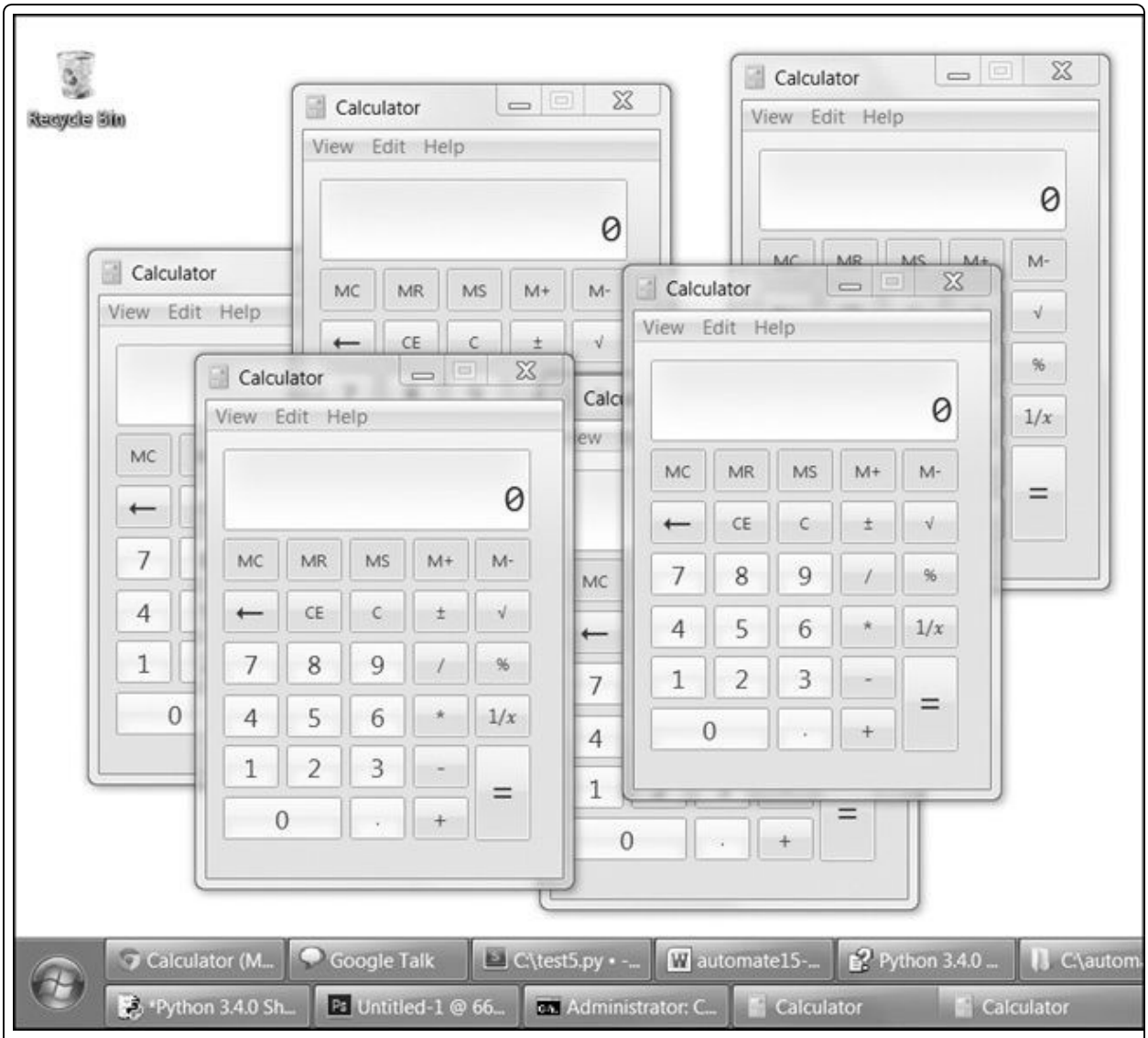


Figura 15-1. Seis procesos en ejecución del mismo programa de calculadora

En un ordenador Windows, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar SUBPROCESO
>>>              SUBPROCESS.Popen("C:\Windows\System32\\calc.exe")
<objeto Subprocess.Popen en 0x0000000003055A58>
```

En Ubuntu Linux, introducirías lo siguiente:

```
>>> importar SUBPROCESO
>>> SUBPROCESO.Popen('USRBIN/GNOME-CALCULATOR')
<objeto subprocess.Popen en 0x7f2bcf93b20>
```

En OS X, el proceso es ligeramente diferente. Consulte Abrir archivos con aplicaciones predeterminadas.

El valor de retorno es un objeto Popen, que tiene dos métodos útiles: poll() y wait().

Puedes pensar en el poll() como si le preguntaras a tu amiga si ha terminado de ejecutar el código que le diste. El poll() devolverá None si el proceso todavía se está ejecutando en el momento de llamar a poll(). Si el programa ha terminado, devolverá el código entero *de salida del* proceso. Un código de salida se usa para indicar si el proceso terminó sin errores (un código de salida de 0) o si un error causó que el proceso terminara (un código de salida distinto de cero-generalmente 1, pero puede variar dependiendo del programa).

El wait() es como esperar a que tu amiga termine de trabajar en su código antes de seguir trabajando en el tuyo. El método wait() bloqueará hasta que el proceso lanzado haya terminado. Esto es útil si quieres que tu programa haga una pausa hasta que el usuario termine con el otro programa. El valor de retorno de wait() es el código entero de salida del proceso.

En Windows, introduzca lo siguiente en el intérprete de comandos interactivo. Tenga en cuenta que wait()
se bloqueará hasta que salga del programa de calculadora iniciado.

```
>>> ❶ calcProc = subprocess.Popen("c:\\Windows\System32\calc.exe")
>>> ❷calcProc.poll()== None
Verdadero
➲ >>> calcProc.wait()
0
>>> calcProc.poll()
0
```

Aquí abrimos un proceso de calculadora❶. Mientras sigue en ejecución, comprobamos si poll() devuelve None❷. Debería, ya que el proceso sigue en ejecución. Entonces cerramos el programa de calculadora y llamamos a wait() sobre el proceso terminado❸. wait() y poll() devuelven ahora 0, indicando que el proceso terminó sin errores.

Pasar argumentos de línea de comandos a Popen()

Puede pasar argumentos de línea de órdenes a los procesos que cree con Popen(). Para ello, pase una lista como único argumento a Popen(). La primera cadena de esta lista será el nombre de archivo ejecutable del programa que desea lanzar; todas las cadenas siguientes serán los argumentos de línea de órdenes que se pasarán al programa cuando se inicie. En efecto, esta lista será el valor de sys.argv para el lanzado.

La mayoría de las aplicaciones con una interfaz gráfica de usuario (GUI) no utilizan argumentos de línea de comandos tan extensamente como los programas basados en línea de comandos o en terminal. Pero la mayoría de las aplicaciones GUI aceptarán un único argumento para un archivo que las aplicaciones abrirán inmediatamente cuando se inicien. Por ejemplo, si utiliza Windows, cree un archivo de texto simple llamado *C:\hello.txt* y, a continuación, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> subprocess.Popen(["C:\\notepad.exe", 'C:\\hello.txt'])
<objeto Subprocess.Popen en 0x00000000032DCEB8>
```

Esto no sólo iniciará la aplicación Notepad, sino que también hará que abra inmediatamente el archivo *C:\hello.txt*.

Programador de tareas, launchd y cron

Si eres un experto en informática, es posible que conozcas el Programador de tareas en Windows, launchd en OS X o el programador cron en Linux. Estas herramientas fiables y bien documentadas de te permiten programar el inicio de aplicaciones a horas concretas. Si quieres saber más sobre ellas, puedes encontrar enlaces a tutoriales en http://nostarch.com/automatestuff/.

Utilizar el programador integrado de su sistema operativo le ahorra tener que escribir su propio código de comprobación del reloj para programar sus programas. Sin embargo, utilice la función `time.sleep()` si sólo necesita que su programa se detenga brevemente. O, en lugar de utilizar el programador del sistema operativo, tu código puede hacer un bucle hasta una fecha y hora determinadas, llamando a `time.sleep(1)` cada vez que pase por el bucle.

Abrir sitios web con Python

La función webbrowser.open() puede lanzar un navegador web desde tu programa a un sitio web específico, en lugar de abrir la aplicación del navegador con subprocess.Popen(). Ver Proyecto: mapit.py con el Módulo webbrowser para más detalles.

Ejecución de otros scripts de Python

Puedes lanzar un script de Python desde Python como cualquier otra aplicación. Sólo tienes que pasar el ejecutable *python.exe* a Popen() y el nombre de fichero del *.py* que desea ejecutar como argumento. Por ejemplo, se ejecutaría el script *hello.py* del Capítulo 1:

```
>>> SUBPROCESO.Popen(['C:\\python34\python.exe', 'hola.py'])
<objeto Subprocess.Popen en 0x000000000331CF28>
```

Pasa a Popen() una lista que contiene una cadena con la ruta del ejecutable de Python y una cadena con el nombre de fichero del script. Si el script que está lanzando necesita argumentos de línea de comandos, añádalos a la lista después del nombre de archivo del script. La ubicación del ejecutable de Python en Windows es *C:\python34\python.exe*. En OS X, es *LibraryFrameworks/Python.framework/Versions/3.3/bin/python3*. En Linux*, es usrbin/python3*.

A diferencia de importar el programa Python como un módulo, cuando tu programa Python lanza otro programa Python, los dos se ejecutan en procesos separados y no podrán compartir las variables del otro.

Abrir archivos con aplicaciones predeterminadas

Al hacer doble clic en un archivo .txt de su ordenador, se iniciará automáticamente la aplicación asociada a la extensión de archivo *.txt*. Su ordenador ya tendrá configuradas varias de estas asociaciones de extensiones de archivo. Python también puede abrir archivos de esta forma con Popen().

Cada sistema operativo tiene un programa que realiza el equivalente a hacer doble clic en un archivo de documento para . En Windows, es el programa de inicio. En OS X, es el programa Abrir. En Ubuntu Linux, es el ver. Introduzca lo siguiente en el shell interactivo, pasando 'start', 'open', o 'see' a Popen() dependiendo de su sistema:

```
>>> fileObj= open('hola.txt', 'w')
>>> fileObj.write('¡Hola mundo!')
12
>>> fileObj.close()
>>> importar SUBPROCESO
>>> SUBPROCESO.Popen(['inicio', 'hola.txt'], shell=True)
```

Aquí escribimos ¡Hola mundo! en un nuevo archivo *hello.txt*. Luego llamamos a Popen(), pasándole una lista que contiene el nombre del programa (en este , 'start' para Windows) y el nombre del archivo. También le pasamos la palabra shell=True, que sólo es necesaria en Windows. El sistema operativo conoce todas las asociaciones de archivos y puede averiguar que debe lanzar, digamos, *Notepad.exe* para manejar el archivo *hola.txt*.

En OS X, el programa Abrir se utiliza para abrir tanto archivos de documentos como programas. Introduzca lo siguiente en el intérprete de comandos interactivo si tiene un Mac:

```
>>> SUBPROCESO.Popen(['abrir', 'APLICACIONESCALCULADORA.app/'])
<objeto subprocess.Popen en 0x10202ff98>
```

Debería abrirse la aplicación Calculadora.

LA FILOSOFÍA UNIX

Los programas bien diseñados para ser lanzados por otros programas se vuelven más potentes que su código por sí solo. *La filosofía Unix* es un conjunto de principios de diseño de software establecidos por los programadores del sistema operativo Unix (sobre el que se construyen los modernos Linux y OS X). Dice que es mejor escribir programas pequeños, de propósito limitado y que puedan interoperar, que aplicaciones grandes y repletas de funciones. Los programas más pequeños son más fáciles de entender y, al ser interoperables, pueden ser los componentes básicos de aplicaciones mucho más potentes.

Las aplicaciones para teléfonos inteligentes también siguen este planteamiento. Si su aplicación de restaurante necesita mostrar indicaciones para llegar a una cafetería, los desarrolladores no han reinventado la rueda escribiendo su propio código de mapas. La aplicación de restaurante simplemente

lanza una aplicación de mapas y le pasa la dirección de la cafetería, igual que tu código Python llamaría a una función y le pasaría argumentos.

Los programas Python que has estado escribiendo en este libro se ajustan en su mayoría a la filosofía Unix, especialmente en un aspecto importante: Usan argumentos de línea de comandos en lugar de llamadas a la función input(). Si toda información que tu programa necesita puede ser suministrada por adelantado, es preferible pasar esta información como argumentos de línea de comandos en lugar de esperar a que el usuario la escriba. De esta forma, los argumentos de la línea de comandos pueden ser introducidos por un usuario humano o suministrados por otro programa. Este enfoque interoperable hará que sus programas sean reutilizables como parte de otro programa.

La única excepción es que usted no quiere que las contraseñas se pasen como argumentos de línea de comandos, ya que la línea de comandos puede registrarlas como parte de su función de historial de comandos. En su lugar, tu programa debería llamar a la función input() cuando necesite que introduzcas una contraseña.

Puedes leer más sobre la filosofía Unix en https://en.wikipedia.org/wiki/Unix_philosophy/.

Proyecto: Programa simple de cuenta atrás

Así como es difícil encontrar una aplicación simple de cronómetro, puede ser difícil encontrar una aplicación simple de cuenta regresiva. Escribamos un programa de cuenta atrás que reproduzca una alarma al final de la cuenta atrás.

A grandes , esto es lo que hará tu programa:

- Cuenta atrás desde 60.
- Reproducir un archivo de sonido (*alarm.wav*) cuando la cuenta atrás

llegue a cero. Esto significa que tu código tendrá que hacer lo siguiente:

- Haz una pausa de un segundo entre la visualización de cada número de la cuenta atrás llamando a `time.sleep()`.
- Llama a `subprocess.Popen()` para abrir el archivo de sonido con la aplicación por defecto.

Abre una nueva ventana del editor de archivos y guárdalo como *countdown.py*.

Paso 1: Cuenta atrás

Este programa requerirá el módulo time para la función time.sleep() y el módulo subprocess para la función subprocess.Popen(). Introduzca el siguiente código y guarde el archivo como *countdown.py*:

```
#! python3
    # countdown.py - Un simple script de cuenta atrás.

    import time, subprocess

❶ timeLeft= 60
    while timeLeft> 0:
❷       print(tiempoIzquierdo, end='')
➲       time.sleep(1)
❹       tiempoIzquierdo= tiempoIzquierdo - 1

    # TODO: Al final de la cuenta atrás, reproducir un archivo de sonido.
```

Después de importar el tiempo y el subproceso, crea una variable llamada timeLeft para mantener el número de segundos que quedan en la cuenta atrás ❶. Puede comenzar en 60-o puede cambiar el valor aquí a lo que usted necesita o incluso hacer que se establece a partir de un argumento de línea de comandos.

En un bucle while, se muestra la cuenta restante ❷, se hace una pausa de un segundo ❸, y luego se decrementa la variable timeLeft ❹ antes de que el bucle comience de nuevo. El bucle seguirá en bucle mientras timeLeft sea mayor que 0. Después de eso, la cuenta atrás habrá terminado.

Paso 2: Reproducir el archivo de sonido

Aunque existen módulos de terceros para reproducir archivos de sonido de distintos formatos, la forma más rápida y sencilla consiste en ejecutar cualquier aplicación que el usuario ya utilice para reproducir archivos de sonido. El sistema operativo averiguará, a partir de la extensión del archivo .wav, qué aplicación debe ejecutar para reproducir el archivo. Este *archivo .wav* podría ser fácilmente otro formato de archivo de sonido, como *.mp3* u *.ogg*.

Puedes utilizar cualquier archivo de sonido que tengas en tu ordenador para que suene al final de la cuenta atrás, o puedes descargar *alarm.wav* de http://nostarch.com/automatestuff/.

Añade lo siguiente a tu código:

```
#! python3
# countdown.py - Un simple script de cuenta atrás.

import time, subprocess

--snip--

# Al final de la cuenta atrás, reproduce un archivo de sonido.
subprocess.Popen(['start', 'alarm.wav'], shell=True)
```

Después de que el bucle while termine, *alarm.wav* (o el archivo de sonido que elijas) se reproducirá para notificar al usuario que la cuenta atrás ha terminado. En Windows, asegúrate de incluir 'start' en la lista que pasas a Popen() y pasa la palabra clave shell=True. En OS , pase 'open' en lugar de 'start' y elimine shell=True.

En lugar de reproducir un archivo de sonido, podría guardar un archivo de texto en algún lugar con un mensaje como *¡Se acabó el tiempo de pausa!* y utilizar Popen() para abrirlo al final de la cuenta atrás. Esto creará efectivamente una ventana emergente con un mensaje. O puede utilizar la función webbrowser.open() para abrir un sitio web específico al final de la cuenta atrás. A diferencia de alguna aplicación gratuita de cuenta regresiva que encontraría en línea, ¡la alarma de su propio programa de cuenta regresiva puede ser lo que usted quiera!

Ideas para programas similares

Una cuenta atrás es un simple retardo antes de continuar la ejecución del programa. También se puede utilizar para otras aplicaciones y funciones, como las siguientes:

- Utilice `time.sleep()` para dar al usuario la oportunidad de pulsar CTRL-C para cancelar una acción, como borrar archivos. Su programa puede imprimir un mensaje "Presione CTRL-C para cancelar" y luego manejar cualquier excepción `KeyboardInterrupt` con sentencias `try` y `except`.
- Para una cuenta atrás a largo plazo, puedes utilizar objetos `timedelta` para medir el número de días, horas, minutos y segundos que faltan para algún momento (¿un cumpleaños? ¿un aniversario?) en el futuro.

Resumen

La época Unix (1 de enero de 1970, a medianoche, UTC) es un tiempo de referencia estándar para muchos lenguajes de programación, incluido Python. Mientras que el módulo de funciones `time.time()` devuelve una marca de tiempo de época (es decir, un valor flotante del número de segundos desde la época de Unix), el módulo `datetime` es mejor para realizar aritmética de fechas y formatear o analizar cadenas con información de fechas.

La función `time.sleep()` se bloqueará (es decir, no volverá) durante un cierto número de segundos. Puede usarse para añadir pausas a tu programa. Pero si desea programar sus programas para que se inicien a una hora determinada, las instrucciones de *http://nostarch.com/automatestuff/* pueden indicarle cómo utilizar el programador que ya proporciona su sistema operativo.

El módulo `de hilos` se utiliza para crear múltiples hilos, lo que es útil cuando se necesita descargar varios archivos o realizar otras tareas simultáneamente. Pero asegúrate de que el hilo lee y escribe sólo variables locales, o podrías tener problemas de concurrencia.

Finalmente, tus programas Python pueden lanzar otras aplicaciones con la función `subprocess.Popen()`. Se pueden pasar argumentos de línea de comandos a la llamada Popen() para abrir documentos específicos con la aplicación. Alternativamente, puedes usar el programa `start`, `open`, o `see` con `Popen()` para usar las asociaciones de archivos de tu ordenador para averiguar automáticamente qué aplicación usar para abrir un documento. Usando las otras aplicaciones de tu ordenador, tus programas Python pueden aprovechar sus capacidades para tus necesidades de automatización.

Preguntas prácticas

P: 1. ¿Qué es la época Unix?

P: 2. ¿Qué función devuelve el número de segundos desde la época Unix?

P: 3. ¿Cómo puede pausar su programa exactamente 5 segundos?

P: 4. ¿Qué devuelve la función round()?

P: 5. ¿Cuál es la diferencia entre un objeto datetime y un objeto timedelta?

P: 6. Digamos que tienes una función llamada spam(). ¿Cómo puedes llamar a esta función y ejecutar el código dentro de ella en un hilo separado?

P: 7. ¿Qué debe hacer para evitar problemas de concurrencia con múltiples hilos?

P: 8. ¿Cómo puede hacer que su programa Python ejecute el programa *calc.exe* ubicado en la carpeta ¿Carpeta *C:\Windows\System32*?

Proyectos prácticos

Para practicar, escribe programas que hagan lo siguiente.

Cronómetro embellecido

Amplíe el proyecto cronómetro de este capítulo para que utilice los métodos de cadena `rjust()` y `ljust()` para "embellecer" la salida. (Estos métodos se trataron en el Capítulo 6.) En lugar de una salida como esta:

```
Vuelt #1:  3.56 (3.56)
  a
Vuelt #2:  8.63 (5.07)
  a
Vuelt #3:  17.68 (9.05)
  a
Vuelt #4:  19.11 (1.43)
  a
```

... la salida tendrá este aspecto:

```
Vuelta # 1:  3.56 (     3.56)
Vuelta # 2:  8.63 (     5.07)
Vuelta # 3: 17.68 (    9.05)
Vuelta # 4: 19.11 (    1.43)
```

Tenga en cuenta necesitará las versiones de cadena de `lapNum`, `lapTime` y `totalTime` variables integer y float para llamar a los métodos string sobre ellas.

A continuación, utilice el módulo `pyperclip` introducido en el capítulo 6 para copiar la salida de texto en el portapapeles, de modo que el usuario pueda pegar rápidamente la salida en un archivo de texto o en un correo electrónico.

Descargador programado de Webcómics

Escribe un programa que compruebe los sitios web de varios cómics web y descargue automáticamente las imágenes si el cómic se ha actualizado desde la última visita del programa. El programador de tu sistema operativo (Scheduled Tasks en Windows, launchd en OS X y cron en Linux) puede ejecutar tu programa Python una vez al día. El propio programa Python puede descargar el cómic y luego copiarlo a tu escritorio para que sea fácil de encontrar. Esto te evitará tener que consultar tú mismo el sitio web para ver si se ha actualizado. (Puedes consultar una lista de cómics en Internet en http://nostarch.com/automatestuff/.)

Capítulo 16. Envío de correo electrónico y mensajes de texto

Revisar y responder el correo electrónico es una enorme pérdida de tiempo. Por supuesto, no puedes escribir un programa que gestione todo tu correo electrónico por ti, ya que cada mensaje requiere su propia respuesta. Pero puedes automatizar un montón de tareas relacionadas con el correo electrónico una vez que sepas cómo escribir programas que puedan enviar y recibir correo electrónico.

Por ejemplo, puede que tengas una hoja de cálculo llena de registros de clientes y quieras enviar a cada cliente un formulario de carta diferente en función de su edad y datos de localización. Es posible que un software comercial no pueda hacer esto por ti; afortunadamente, puedes escribir tu propio programa para enviar estos correos electrónicos, ahorrándote mucho tiempo copiando y pegando correos electrónicos de formulario.

También puedes escribir programas que envíen mensajes de correo electrónico y SMS para avisarte de cosas incluso cuando no estás delante del ordenador. Si estás automatizando una tarea que lleva un par de horas, no querrás volver al ordenador cada pocos minutos para comprobar el estado del programa. En su lugar, programa puede enviar un mensaje de texto a tu teléfono cuando haya terminado, lo que te permitirá centrarte en cosas más importantes mientras estás lejos del ordenador.

SMTP

Al igual que HTTP es el protocolo utilizado por los ordenadores para enviar páginas web a través de Internet, *el Protocolo Simple de Transferencia de Correo (SMTP*) es el protocolo utilizado para enviar correo electrónico. SMTP dicta cómo deben formatearse, encriptarse y retransmitirse los mensajes de correo electrónico entre servidores de correo, y todos los demás detalles que tu ordenador gestiona después de hacer clic en Enviar. No necesitas conocer estos detalles técnicos,
porque el módulo smtplib de Python las simplifica en unas pocas funciones.

SMTP sólo se ocupa de enviar correos electrónicos a otras personas. Un protocolo diferente, llamado IMAP, se ocupa de recuperar los correos electrónicos que te han enviado y se describe en IMAP.

Envío de correo electrónico

Puede que estés familiarizado con el envío de correos electrónicos desde Outlook o Thunderbird o a través de un sitio web como Gmail o Yahoo! Mail. Desafortunadamente, Python no te ofrece una interfaz gráfica de usuario agradable como esos servicios. En su lugar, tienes que llamar a funciones para realizar cada paso importante de SMTP, como se muestra en el siguiente ejemplo de shell interactivo.

NOTA

No introduzcas este ejemplo en IDLE; no funcionará porque smtp.ejemplo.com,bob@example.com , MY_SECRET_PASSWORD, *y*alice@example.com *son sólo marcadores de posición. Este código es sólo una visión general del proceso de envío de correo electrónico con Python.*

```
>>> importar smtplib
>>> smtpObj= smtplib.SMTP('smtp.ejemplo.com', 587)
>>> smtpObj.ehlo()
(250, b"mx.ejemplo.com a su servicio, [216.172.148.131]\nSIZE 35882577\
n8BITMIME\nSTARTTLS\nENHANCEDSTATUSCODES\nCHUNKING')
>>> smtpObj.STARTTLS()
(220, b'2.0.0 Listo para iniciar TLS')
>>>  smtpObj.login('bob@example.com', '      MY_SECRET_PASSWORD')
(235, b'2.7.0 Aceptado')
>>> smtpObj.sendmail('bob@example.com',
'alice@example.com', 'Asunto: Adiós.\nQuerida Alice, adiós y GRACIAS
por todo el pescado. Atentamente, Bob')
{}
>>> smtpObj.quit()
(221, b'2.0.0 cerrando conexión ko10sm23097611pbd.52 - gsmtp")
```

En las siguientes secciones, recorreremos cada paso, reemplazando los marcadores de posición con su información para conectarse e iniciar sesión en un servidor SMTP, enviar un correo electrónico y desconectarse del servidor.

Conexión a un servidor SMTP

Si alguna vez ha configurado Thunderbird, Outlook u otro programa para conectarse a su cuenta de correo electrónico, es posible que esté familiarizado con la configuración del servidor SMTP y el puerto. Estos ajustes serán diferentes para cada proveedor de correo electrónico, pero una búsqueda en la web de *<su proveedor> smtp settings* debería mostrar el servidor y el puerto a utilizar.

El nombre de dominio del servidor SMTP suele ser el nombre de dominio de tu proveedor de correo electrónico, con *smtp.* delante. Por ejemplo, el servidor SMTP de Gmail está en *smtp.gmail.com*. La Tabla 16-1 enumera algunos proveedores de correo electrónico comunes y sus servidores SMTP. (El puerto es un valor entero y casi siempre será 587, que es el utilizado por el estándar de encriptación de comandos, TLS).

Tabla 16-1. Proveedores de correo electrónico y sus servidores SMTP

Proveedor	**Nombre de dominio del servidor SMTP**
Gmail	*smtp.gmail.com*
Outlook.com/Hotmail.com	*smtp-mail.outlook.com*
Correo Yahoo	*smtp.mail.yahoo.com*
AT&T	*smpt.mail.att.net* (puerto 465)
Comcast	*smtp.comcast.net*
Verizon	*smtp.verizon.net* (puerto 465)

Una vez que tenga el nombre de dominio y la información del puerto de su proveedor de correo electrónico, cree un objeto SMTP llamando a smptlib.SMTP(), pasando el nombre de dominio como argumento de cadena, y pasando el puerto como argumento entero. El objeto SMTP representa una conexión a un servidor de correo SMTP y tiene métodos para enviar correos electrónicos. Por ejemplo, la siguiente llamada crea un objeto SMTP para conectarse a Gmail:

```
>>> smtpObj= smtplib.SMTP('smtp.gmail.com', 587)
>>> type(smtpObj)
<clase 'smtplib.SMTP'>
```

Introducir type(smtpObj) le muestra que hay un objeto SMTP almacenado en smtpObj. Necesitará este objeto SMTP para poder llamar a los métodos que le registran y envían correos electrónicos. Si la llamada a smptlib.SMTP() no tiene éxito, su SMTP

puede no soportar TLS en el puerto 587. En este , tendrá que crear un SMTP usando smtplib.SMTP_SSL() y el puerto 465 en su lugar.

```
>>> smtpObj= smtplib.SMTP_SSL('smtp.gmail.com', 465)
```

NOTA

Si no está conectado a Internet, Python lanzará un socket.gaierror: [Errno 11004] getaddrinfo *failed o una excepción similar.*

Para sus programas, las diferencias entre TLS y SSL no son importantes. Sólo necesitas saber qué estándar de encriptación utiliza tu servidor SMTP para saber cómo conectarte a él. En todos los ejemplos de shell interactivo que siguen, la variable smtpObj contendrá un objeto SMTP devuelto por la función smtplib.SMTP() o smtplib.SMTP_SSL().

Envío del mensaje SMTP "Hola

Una vez que tengas el objeto SMTP, llama a su extrañamente llamado método ehlo() para "saludar" al servidor de correo SMTP. Este saludo es el primer paso en SMTP y es importante para establecer una conexión con el servidor. No necesitas conocer los detalles de estos protocolos. Sólo asegúrate de llamar al método ehlo() lo primero después de obtener el objeto SMTP o de lo contrario las llamadas a métodos posteriores darán lugar a errores.

A continuación se muestra un ejemplo de llamada a ehlo() y su valor de retorno:

```
>>> smtpObj.ehlo()
(250, b"mx.google.com a su servicio, [216.172.148.131]\nSIZE 35882577\
n8BITMIME\nSTARTTLS\nENHANCEDSTATUSCODES\nCHUNKING')
```

Si el primer elemento de la tupla devuelta es el número entero 250 (el código de "éxito" en SMTP), el saludo se ha realizado correctamente.

Iniciar el cifrado TLS

Si te estás conectando al puerto 587 del servidor SMTP (es decir, estás usando encriptación TLS), necesitarás llamar al método `starttls()` a continuación. Este paso obligatorio habilita el cifrado para su conexión. Si te estás conectando al puerto 465 (usando SSL), entonces la encriptación ya está configurada, y deberías saltarte este paso.

He aquí un ejemplo de la llamada al método `starttls()`:

```
>>> smtpObj.STARTTLS()
(220, b'2.0.0 Listo para iniciar TLS')
```

`starttls()` pone su conexión SMTP en modo TLS. El `220` en el valor de retorno le indica que el servidor está listo.

Inicio de sesión en el servidor SMTP

Una vez que su conexión encriptada al servidor SMTP está configurada, puede iniciar sesión con su nombre de usuario (normalmente su dirección de correo electrónico) y contraseña de correo electrónico llamando al método `login()`.

```
>>> smtpObj.login('my_email_address@gmail.com', 'MY_SECRET_PASSWORD')
(235, b'2.7.0 Aceptado')
```

CONTRASEÑAS ESPECÍFICAS DE LAS APLICACIONES DE GMAIL

Gmail dispone de una función de seguridad adicional para las cuentas de Google denominada *contraseñas específicas de la aplicación*. Si recibes un mensaje de error de `contraseña específica de la aplicación` cuando tu programa intenta acceder, tendrás que configurar una de estas contraseñas para tu script de Python. Consulta los recursos *de http://nostarch.com/automatestuff/* para obtener instrucciones detalladas sobre cómo configurar una contraseña específica de la aplicación para tu cuenta de Google.

Pase una cadena con su dirección de correo electrónico como primer argumento y una cadena con su contraseña como segundo argumento. El `235` en el valor de retorno significa que la autenticación se ha realizado correctamente. Python lanzará una excepción `smtplib.SMTPAuthenticationError` para contraseñas incorrectas.

ADVERTENCIA

Ten cuidado con poner contraseñas en tu código fuente. Si alguien copia tu programa, ¡tendrá acceso a tu cuenta de correo! Es una buena idea llamar a `input()` y hacer que el usuario escriba la contraseña. Puede ser inconveniente tener que introducir una contraseña cada vez que ejecutas tu programa, pero este enfoque evitará que dejes tu contraseña en un archivo sin cifrar en tu ordenador donde un hacker o un ladrón de portátiles podría conseguirla fácilmente.

Enviar un correo electrónico

Una vez que haya iniciado sesión en el servidor SMTP de su proveedor de correo electrónico, puede llamar al método sendmail() para enviar realmente el correo electrónico. La llamada al método sendmail() tiene este aspecto:

```
>>> smtpObj.sendmail('my_email_address@gmail.com ', 'recipient@example.com ', 'Asunto:
Adiós.\nQuerida Alice, adiós y GRACIAS por todo el pescado. Atentamente, Bob')
{}
```

El método sendmail() requiere tres argumentos.

- Su dirección de correo electrónico como cadena (para la dirección "de" del correo electrónico)
- La dirección de correo electrónico del destinatario como una cadena o una lista de cadenas para varios destinatarios (para la dirección "para").
- El cuerpo del correo electrónico como cadena

El inicio de la cadena del cuerpo del correo *electrónico debe* comenzar con 'Asunto: \n' para el asunto del mensaje. El carácter de nueva línea '\n' separa la línea de asunto del cuerpo principal del correo electrónico.

El valor devuelto por sendmail() es un diccionario. Habrá un par clave-valor en el diccionario por cada destinatario para el que *falló la* entrega del correo electrónico. Un diccionario vacío significa que todos los destinatarios recibieron *correctamente* el correo electrónico.

Desconexión del servidor SMTP

Asegúrese de llamar al método `quit()` cuando haya terminado de enviar emails. Esto desconectará su programa del servidor SMTP.

```
>>> smtpObj.quit()
(221, b'2.0.0 cerrando conexión ko10sm23097611pbd.52 - gsmtp')
```

El `221` en el valor de retorno significa que la sesión está terminando.

Para revisar todos los pasos para conectarse e iniciar sesión en el servidor, enviar correo electrónico y desconectarse, consulte Enviar correo electrónico.

IMAP

Así como SMTP es el protocolo para enviar correo electrónico, el *Protocolo de Acceso a Mensajes de Internet (IMAP)* especifica cómo comunicarse con el servidor de un proveedor de correo electrónico para recuperar los mensajes enviados a tu dirección de correo electrónico. Python viene con un módulo `imaplib`, pero de hecho el módulo de terceros `imapclient` es más fácil de usar. Este capítulo proporciona una introducción al uso de IMAPClient; la documentación completa está en http://imapclient.readthedocs.org/.

El módulo `imapclient` descarga correos electrónicos de un servidor IMAP en un formato bastante complicado. Lo más probable es que quieras convertirlos de este formato a simples valores de cadena. El `pyzmail` hace el trabajo duro de analizar estos mensajes de correo electrónico para usted. Puedes encontrar la documentación completa de PyzMail en http://www.magiksys.net/pyzmail/.

Instala `imapclient` y `pyzmail` desde una ventana de Terminal. El Apéndice A contiene los pasos para instalar módulos de terceros.

Recuperar y eliminar correos electrónicos con IMAP

Encontrar y recuperar un correo electrónico en Python es un proceso de varios pasos que requiere los módulos de terceros imapclient y pyzmail. Sólo para darte una visión general, aquí hay un ejemplo completo de iniciar sesión en un servidor IMAP, buscar correos electrónicos, recuperarlos y luego extraer el texto de los mensajes de correo electrónico de ellos.

```
>>> importar imapclient
>>> imapObj= imapclient.IMAPClient('imap.gmail.com', SSL=TRUE)
>>> imapObj.login('my_email_address@gmail.com ', ' MY_SECRET_PASSWORD ')
'my_email_address@gmail.com Jane Doe autentificada (Éxito)'
>>> imapObj.select_folder('INBOX', readonly=True)
>>> UIDs= imapObj.search(['DESDE 05-Jul-2014'])
>>> UIDs
[40032, 40033, 40034, 40035, 40036, 40037, 40038, 40039, 40040, 40041]
>>> RAWMESSAGES= imapObj.fetch([40041], ['BODY[]', 'FLAGS'])
>>> importar pyzmail
>>> mensaje = pyzmail.PYZMESSAGE.factory(RAWMESSAGES[40041]['BODY[]'])
>>> message.GET_SUBJECT()
¡Hola!
>>> message.GET_ADDRESSES('from')
[('Edward Snowden', 'esnowden@nsa.gov')]
>>> message.GET_ADDRESSES('to')
[(Jane Doe', 'jdoe@example.com')]
>>> message.GET_ADDRESSES('cc')
[]
>>> message.GET_ADDRESSES('bcc')
[]
>>> mensaje.parte_texto != None
Verdadero
>>> message.text_part.get_payload().decode(message.text_part.CHARSET)
"Sigue el dinero.
>>> message.html_part != None
Verdadero
>>> message.html_part.get_payload().decode(message.html_part.CHARSET) '<div
dir="ltr"><div>Hasta luego, y gracias por todos los peces!<br><br></div>-
Al<br></div>\r\n'
>>> imapObj.logout()
```

No tienes que memorizar estos pasos. Después de repasar cada paso en detalle, puedes volver a este resumen para refrescar la memoria.

Conexión a un servidor IMAP

Al igual que necesitabas un objeto SMTP para conectarte a un servidor SMTP y enviar correo electrónico, necesitas un objeto IMAPClient para conectarte a un servidor IMAP y recibir correo electrónico. Primero necesitarás el nombre de dominio del servidor IMAP de tu proveedor de correo electrónico. Este será diferente del nombre de dominio del servidor SMTP. La Tabla 16-2 lista los servidores IMAP de varios proveedores de correo electrónico populares.

Tabla 16-2. Proveedores de correo electrónico y sus servidores IMAP Proveedores de correo electrónico y sus servidores IMAP

Proveedor	**Nombre de dominio del servidor IMAP**
Gmail	*imap.gmail.com*
Outlook.com/Hotmail.com	*imap-mail.outlook.com*
Correo Yahoo	*imap.mail.yahoo.com*
AT&T	*imap.mail.att.net*
Comcast	*imap.comcast.net*
Verizon	*entrante.verizon.net*

Una vez que tenga el nombre de dominio del servidor IMAP, llame a la función imapclient.IMAPClient() para crear un objeto IMAPClient. La mayoría de los proveedores de correo electrónico requieren encriptación SSL, así que pase el argumento de palabra clave ssl=True. Introduzca lo siguiente en el intérprete de comandos interactivo (utilizando el nombre de dominio de su proveedor):

```
>>> importar imapclient
>>> imapObj= imapclient.IMAPClient('imap.gmail.com', SSL=TRUE)
```

En todos los ejemplos de shell interactivo de las siguientes secciones, la variable imapObj contendrá un objeto IMAPClient devuelto por la función imapclient.IMAPClient(). En este contexto, un *cliente* es el objeto que se conecta al servidor.

Iniciar sesión en el servidor IMAP

Una vez que tengas un objeto IMAPClient, llama a su método login(), introduciendo el nombre de usuario (que suele ser tu dirección de correo electrónico) y la contraseña como cadenas.

```
>>> imapObj.login('my_email_address@gmail.com ', ' MY_SECRET_PASSWORD ')
'my_email_address@gmail.com Jane Doe autentificada (Éxito)'
```

ADVERTENCIA

Recuerde, ¡nunca escriba una contraseña directamente en su código! En su lugar, diseñe su programa para aceptar la contraseña devuelta por ***input()***.

Si el servidor IMAP rechaza esta combinación de nombre de usuario y contraseñaPython lanzará una excepción imaplib.error. Para las cuentas de Gmail, es posible que tengas que utilizar una contraseña específica de la aplicación; para obtener más información, consulta Contraseñas específicas de la aplicación de Gmail.

Búsqueda de correo electrónico

Una vez iniciada la sesión, el proceso para recuperar el correo electrónico que te interesa consta de dos pasos. En primer lugar, debes seleccionar la carpeta en la que deseas buscar. A continuación, debe llamar al método search() del objeto IMAPClient, pasándole una cadena de palabras clave de búsqueda IMAP.

Seleccionar una carpeta

Casi todas las cuentas tienen una carpeta INBOX por defecto, pero también puede obtener una lista de carpetas llamando al método list_folders() del objeto IMAPClient. Este método devuelve una lista de tuplas. Cada tupla contiene información sobre una única carpeta.
Continúe con el ejemplo de shell interactivo introduciendo lo siguiente:

```
>>> importar pprint
>>> pprint.pprint(imapObj.LIST_FOLDERS())
[('\\HasNoChildren',), '', 'Borradores'),
  (('\HasNoChildren',), '', 'Filler'),
  (('\\HasNoChildren',), '', 'INBOX'),
  (('\HasNoChildren',), '', 'Sent'),
--snip-
  (('\HasNoChildren', '\\Flagged'), '', '[Gmail]Starred'),
  (('\HasNoChildren',    '\\Trash'),    '',    '[Gmail]Trash')]
```

Este es el aspecto que podría tener el resultado si tienes una cuenta de Gmail. Los tres valores de cada una de las tuplas (por ejemplo, (('\HasNoChildren',), '/', '[Gmail]/Starred') son los siguientes)
-son los siguientes:

- Una tupla de las banderas de la carpeta. (Exactamente lo que estas banderas representan está más allá del alcance de este libro, y usted puede ignorar con seguridad este campo).
- El delimitador utilizado en la cadena de nombres para separar las carpetas padre de las subcarpetas.
- El nombre completo de la carpeta.

Para seleccionar una carpeta en la que buscar, introduzca el nombre de la como cadena en el campo Método select_folder() del objeto IMAPClient.

```
>>> imapObj.select_folder('INBOX', readonly=True)
```

Puede ignorar el valor de retorno de select_folder(). Si la carpeta seleccionada no existe, Python lanzará una excepción imaplib.error.

El argumento de la palabra clave readonly=True evita que accidentalmente realice cambios o eliminaciones en cualquiera de los correos electrónicos de esta carpeta durante las siguientes llamadas al método. A menos que desee eliminar correos electrónicos, es una buena idea establecer siempre readonly a True.

Realizar la búsqueda

Con una carpeta seleccionada, ahora puede buscar correos electrónicos con el método search() del objeto IMAPClient. El argumento de search() es una lista de cadenas, cada una formateada según las claves de búsqueda de IMAP. La Tabla 16-3 describe las distintas claves de búsqueda.

Tabla 16-3. Claves de búsqueda IMAP

Clave de búsqueda	Significado
TODOS	Devuelve todos los mensajes de la carpeta. Si solicita todos los mensajes de una carpeta de gran tamaño, es posible que se encuentre con límites de tamaño de imaplib. Consulte Límites de tamaño.
"ANTES *date*", "ON *date*", 'SINCE *date*'	Estas tres claves de búsqueda devuelven, respectivamente, los mensajes que fueron recibidos por el servidor IMAP antes, el o después de la *fecha* dada. La fecha debe tener el formato 05-Jul-2015. Además, mientras que 'DESDE 05-Jul-2015' coincidirá con mensajes del 5 de julio y posteriores, "ANTES 05-Jul-2015coincidirá sólo con mensajes anteriores al 5 de julio, pero no con el mismo 5 de julio.
'ASUNTO *cadena*', "CUERPO *cadena*', TEXTO *cadena*	Devuelve los mensajes en los que *la cadena se* encuentra en el asunto, el cuerpo o cualquiera de los dos, respectivamente. Si *la cadena* espacios, enciérrela entre comillas dobles: 'TEXTO "buscar con espacios"'.
"DE *cadena*", ""*cadena* TO", *"cadena* CC", "BCC *cadena*	Devuelve al l los mensajes en los que *la cadena* se encuentra en las direcciones de correo electrónico "de", "a", "cc" (copia oculta) o "bcc" (copia oculta), respectivamente. Si hay varias direcciones de correo electrónico en la *cadena*, sepárelas con espacios y todas entre comillas dobles: 'CC *"firstcc@example.com secondcc@example.com"'*.
VISTO, NO VISTO	Devuelve todos los mensajes con y sin la bandera *\Seen*, respectivamente. Un correo electrónico obtiene el *\Seen* flag si se ha accedido a él con una llamada al método fetch() (descrito más adelante) o si se hace clic en él cuando se está revisando el correo electrónico en un programa de correo electrónico o en un navegador web. Es más común decir que el correo electrónico ha sido "leído" en lugar de "visto", pero significan lo mismo.
""CONTESTADO"", "SIN CONTESTAR	Devuelve todos los mensajes con y sin la bandera *\Answered*, respectivamente. Un mensaje obtiene la bandera *\Answered* cuando se responde a él.

BORRADO", "NO BORRADO	Devuelve todos los mensajes con y sin el indicador *\Deleted*, respectivamente. Los mensajes de correo electrónico eliminados con el método delete_messages() reciben el indicador *\Deleted*, pero no se eliminan permanentemente hasta que se llama al método expunge() (consulte Eliminación de mensajes de correo electrónico). Tenga en cuenta que algunos proveedores de correo electrónico, como Gmail, eliminan automáticamente los mensajes de correo electrónico.
"BORRADOR", "NO BORRADOR	Devuelve todos los mensajes con y sin la bandera *\Borrador*, respectivamente. Los mensajes en borrador suelen guardarse en una carpeta separada, Borradores, en lugar de en la carpeta INBOX.
"MARCADO", "NO MARCADO	Devuelve todos los mensajes con y sin la bandera *\Flagged*, respectivamente. Esta bandera se utiliza normalmente para marcar los mensajes de correo electrónico como "Importante" o "Urgente".
'MAYOR *N*', *N* MÁS PEQUEÑO	Devuelve todos los mensajes mayores o menores que *N* bytes, respectivamente.
Tecla de búsqueda NOT	Devuelve los mensajes *que search-key no* habría devuelto.
OR *búsqueda-clave1 búsqueda-clave2'*	Devuelve los mensajes que coinciden *con* la primera o la segunda *clave de búsqueda*.

Tenga en cuenta que algunos servidores IMAP pueden tener implementaciones ligeramente diferentes para el manejo de sus banderas y claves de búsqueda. Puede ser necesario experimentar un poco en el shell interactivo para ver exactamente cómo se comportan.

Puede pasar múltiples cadenas de claves de búsqueda IMAP en el argumento de lista al método search(). Los mensajes devueltos son los que coinciden con *todas las* claves de búsqueda. Si desea que coincida *alguna de* las claves de búsqueda, utilice la clave de búsqueda OR. Para las claves de búsqueda NOT y OR, una y dos claves de búsqueda completas siguen a NOT y OR, respectivamente.

He aquí algunos ejemplos de llamadas al método search() junto con sus significados:

- **imapObj.search(['ALL'])**. Devuelve todos los mensajes de la carpeta seleccionada actualmente.
- **imapObj.search(['ON 05-Jul-2015'])**. Devuelve todos los mensajes enviados el 5 de julio de 2015.
- **imapObj.search(['DESDE 01-Ene-2015', 'ANTES 01-Feb-2015', UNSEEN'])**. Devuelve todos los mensajes enviados en enero de 2015 que no se hayan leído. (Ten en cuenta que esto significa *a partir* del 1 de enero y hasta el 1 de febrero inclusive).

- **imapObj.search(['DESDE 01-Ene-2015', 'DESDE alice@example.com'])**. Devuelve todos los mensajes de*alice@example.com* enviados desde principios de 2015.
- **imapObj.search(['DESDE 01-Ene-2015', 'NO DE alice@example.com'])**. Devuelve todos los mensajes enviados desde todos excepto*alice@example.com* desde el inicio de 2015.
- **imapObj.search(['OR FROMalice@example.com FROM bob@example.com'])**. Devuelve todos los mensajes enviados *alice@example.com*o *.bob@example.com*
- **imapObj.search(['FROM alice@example.com', 'FROM bob@example.com'])**. ¡Ejemplo trucado! Esta búsqueda nunca devolverá ningún mensaje, porque los mensajes deben coincidir con *todas las* palabras clave de búsqueda. Dado que sólo puede haber una dirección "from", es imposible que un mensaje proceda tanto de *alice@example.com* y *.bob@example.com*

El método search() no devuelve los correos electrónicos en sí, sino identificadores únicos (UID) para los correos electrónicos, como valores enteros. A continuación, puede pasar estos UID al método fetch() para obtener el contenido del correo electrónico.

Continúe con el ejemplo de shell interactivo introduciendo lo siguiente:

```
>>> UIDs= imapObj.search(['DESDE 05-Jul-2015'])
>>> UIDs
[40032, 40033, 40034, 40035, 40036, 40037, 40038, 40039, 40040, 40041]
```

Aquí, la lista de IDs de mensajes (para mensajes recibidos del 5 de julio en adelante) devuelta por search() se almacena en UIDs. La lista de UIDs devueltos en su ordenador será diferente de los mostrados aquí; son únicos para una cuenta de correo electrónico en particular. Cuando más tarde pase UIDs a otras llamadas de función, utilice los valores UID que recibió, no los impresos en los ejemplos de este libro.

Límites de tamaño

Si tu búsqueda coincide con un gran número de mensajes de correo electrónico, Python puede lanzar una excepción que diga imaplib.error: got more than 10000 bytes. Cuando esto ocurra, tendrás que desconectarte y volver a conectarte al servidor IMAP e intentarlo de nuevo.

Este límite existe para evitar que tus programas Python consuman demasiada memoria. Desafortunadamente, el límite de tamaño por defecto es a menudo demasiado pequeño. Puede cambiar este límite de 10.000 bytes a 10.000.000 bytes ejecutando este código:

```
>>> importar imaplib
>>> imaplib._MAXLINE= 10000000
```

Esto debería evitar que este mensaje de error vuelva a aparecer. Puede que quieras hacer que estas dos líneas formen parte de cada programa IMAP que escribas.

UTILIZANDO EL MÉTODO GMAIL_SEARCH( ) DE IMAPCLIENT

Si inicias sesión en el servidor *imap.gmail.com* para acceder a una cuenta de Gmail, el objeto IMAPClient proporciona una función de búsqueda adicional que imita la barra de búsqueda de la parte superior de la página web de Gmail, como se muestra en la Figura 16-1.

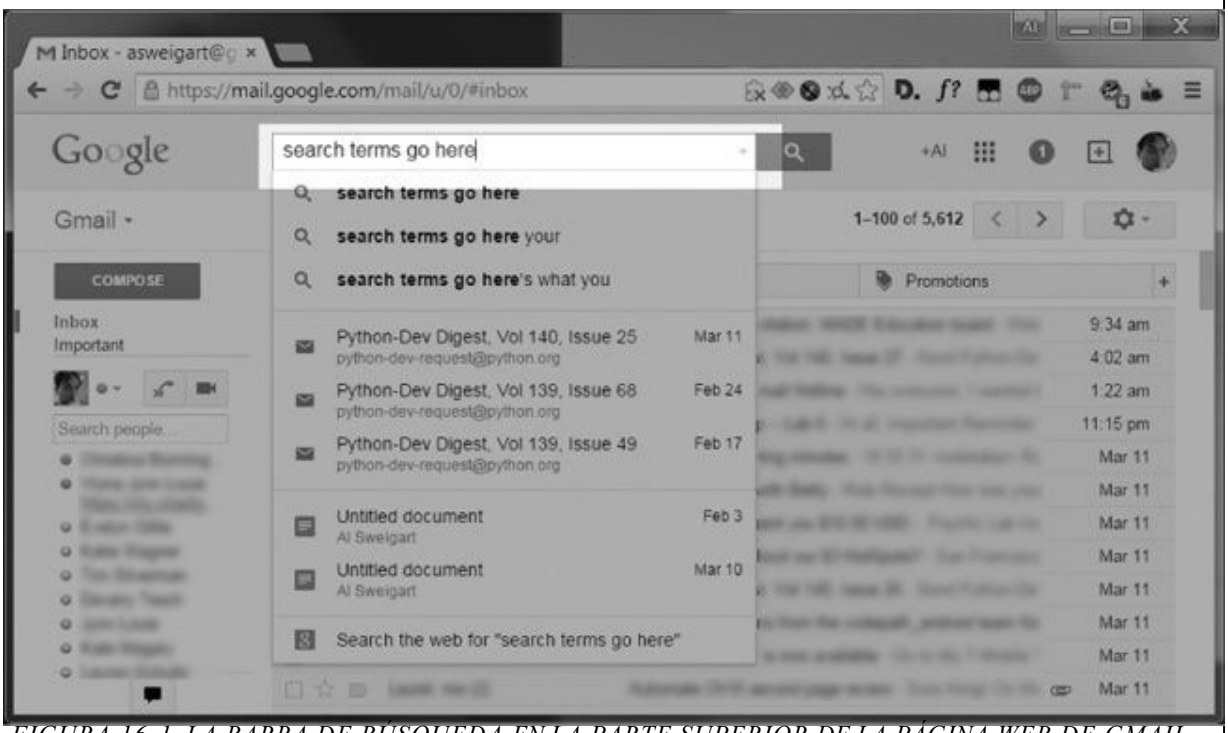


FIGURA 16-1. LA BARRA DE BÚSQUEDA EN LA PARTE SUPERIOR DE LA PÁGINA WEB DE GMAIL

En lugar de buscar con las teclas de búsqueda IMAP, puedes utilizar el motor de búsqueda más sofisticado de Gmail. Gmail realiza un buen trabajo a la hora de buscar palabras estrechamente relacionadas (por ejemplo, una búsqueda de *"driving"* también buscará *"drive"* y "*drove"*) y ordena los resultados de la búsqueda según las coincidencias más significativas. También puedes utilizar los operadores de búsqueda avanzada de Gmail (para más información, consulta http://nostarch.com/automatestuff/). Si accedes a una cuenta de Gmail, pasa los términos de búsqueda al método gmail_search() en lugar de al método search(), como en el siguiente ejemplo de shell interactivo:

```
>>> UIDs= imapObj.GMAIL_SEARCH('sentido de la vida')
>>> UIDs
[42]
```

Ah, sí, ¡ahí está ese correo electrónico con el significado de la vida! Lo estaba buscando.

Obtener un correo electrónico y marcarlo como leído

Una vez que tenga una lista de UIDs, puede llamar a la función fetch() del objeto IMAPClient para obtener el contenido real del correo electrónico.

La lista de UIDs será el primer argumento de fetch(). El segundo argumento debe ser la lista ['BODY[]'], que indica a fetch() que descargue todo el contenido del cuerpo de los mensajes especificados en la lista de UID.

Continuemos con nuestro ejemplo de shell interactivo.

```
>>> rawMessages= imapObj.fetch(UIDs, ['BODY[]'])
>>> importar pprint
>>> pprint.pprint(mensajesbrutos)
{40040: {'BODY[]': 'Entregado a: my_email_address@gmail.com\r\n'  'Recibido:
                   por 10.76.71.167 con SMTP id '
--snip--
                   '\r\n'
                   '----------=_Part_6000970_707736290.1404819487066--\r\n',
         'SEQ':    5430}}
```

Importa pprint y pasa el valor de retorno de fetch(), almacenado en la variable rawMessages, a pprint.pprint() para "imprimirlo bonito, verás que este valor de retorno es un diccionario anidado de mensajes con UIDs como claves. Cada mensaje se almacena como un diccionario con dos claves: 'BODY[]' y 'SEQ'. La clave ''BODY[]" corresponde al cuerpo del mensaje. La clave ''SEQ" corresponde a un *número de secuencia*, que tiene una función similar a la del UID. Puede .

Como puedes ver, el contenido del mensaje en la clave 'BODY[]' es bastante ininteligible. Está en un formato llamado RFC 822, que está diseñado para que lo lean los servidores IMAP. Pero no necesitas entender el formato RFC 822; más adelante en este capítulo, el módulo pyzmail le dará sentido por ti.

Cuando seleccionó una carpeta para buscar, llamó a select_folder() con el argumento de palabra clave readonly=True. Esto evitará que borre accidentalmente un correo electrónico, pero también significa que los correos electrónicos no se marcarán como leídos los recupera con el método fetch(). Si desea los correos electrónicos se marquen como leídos cuando los recupere, tendrá que pasar readonly=False a select_folder(). Si la carpeta seleccionada ya está en modo de sólo lectura, puede volver a seleccionar la carpeta actual con otra llamada a select_folder(), esta vez con el argumento de palabra clave readonly=False:

```
>>> imapObj.select_folder('INBOX', READONLY=FALSE)
```

Obtener direcciones de correo electrónico de un mensaje sin procesar

Los mensajes sin procesar devueltos por el método fetch() todavía no son muy útiles para la gente que sólo quiere leer su correo electrónico. El módulo pyzmail analiza estos mensajes sin procesar y los devuelve como objetos PyzMessage, que hacen que el asunto, el cuerpo, el campo "Para", el campo "De" y otras secciones del correo electrónico sean fácilmente accesibles para tu código Python.

Continúe el ejemplo del shell interactivo con lo siguiente (utilizando los UID de su propia cuenta de correo electrónico, no los que se muestran aquí):

```
>>> importar pyzmail
>>> mensaje = pyzmail.PyzMessage.factory(rawMessages[40041]['BODY[]'])
```

Primero, importa pyzmail. Luego, para crear un objeto PyzMessage de un correo electrónico, llama a la función pyzmail.PyzMessage.factory() y pásale la sección "BODY[]" del mensaje sin procesar. Almacena el resultado en el mensaje. Ahora el mensaje contiene un objeto PyzMessage, que tiene varios métodos que facilitan la obtención del asunto del correo electrónico, así como todas las direcciones del remitente y del destinatario. El objeto
El método get_subject() devuelve el asunto como un simple valor de cadena. El método get_addresses() devuelve una lista de direcciones para el campo que se le pase. Por ejemplo, las llamadas al método podrían tener este aspecto:

```
>>> message.get_subject()
¡Hola!
>>> message.get_addresses('from')
[('Edward Snowden', 'esnowden@nsa.gov')]
>>> message.get_addresses('to')
[(Jane Doe', 'my__email__address@gmail.com')]
>>> message.get_addresses('cc')
[]
>>> message.get_addresses('bcc')
[]
```

Tenga en cuenta que el argumento de get_addresses() es "from", 'to', 'cc' o 'bcc'. El valor de retorno de get_addresses() es una lista de tuplas. Cada tupla contiene dos cadenas: La primera es el nombre asociado a la dirección de correo electrónico, y la segunda es la propia dirección de correo electrónico. Si no hay direcciones en el campo solicitado, get_addresses() devuelve una lista en blanco. En este caso, los campos "cc" copia oculta y "bcc" copia oculta no contenían ninguna dirección, por lo que devolvían listas vacías.

Obtener el cuerpo de un mensaje en bruto

Los correos electrónicos pueden enviarse como texto plano, HTML o ambos. Los correos electrónicos en texto plano sólo contienen texto, mientras que los correos HTML pueden tener colores, fuentes, imágenes y otras características que hacen que el mensaje de correo electrónico parezca una pequeña página web. Si un correo electrónico es sólo texto plano, su objeto PyzMessage tendrá los atributos html_part establecidos en None. Del mismo modo, si un correo electrónico es sólo HTML, su objeto PyzMessage tendrá su atributo text_part establecido en None.

De lo contrario, el valor text_part o html_part tendrá un método get_payload() que devuelve el cuerpo del correo electrónico como un valor del tipo de datos bytes. (El tipo de datos bytes está fuera del alcance de este libro.) Pero esto *sigue* sin ser un valor de cadena podamos usar. ¡Uf! El último paso es llamar al método decode() sobre el valor bytes devuelto por get_payload(). El método decode() toma un argumento: la codificación de caracteres del mensaje, almacenada en el atributo text_part.charset o html_part.charset. Esto, finalmente, devolverá la cadena del cuerpo del correo electrónico.

Continúe con el ejemplo de shell interactivo introduciendo lo siguiente:

```
>>> ❶message.text_part != None
    Verdadero
    >>>   message.text_part.get_payload().decode(message.text_part.CHARSET)
❷'¡Hasta la vista, y gracias por todo el pescado!❷
➲  >>> message.html_part != None
    Verdadero
>>> ❹message.html_part.get_payload().decode(message.html_part.CHARSET)
    <div dir="ltr"><div>Hasta luego, y gracias por todos los peces!<br><br></div>-Al
    <br></div>\r\n'
```

El correo electrónico con el que estamos trabajando tiene contenido en texto plano y HTML, por lo que el objeto PyzMessage almacenado en message tiene atributos text_part y html_part no iguales a None ❶❸. Llamar a () sobre la parte_texto del mensaje y luego llamar a decode() sobre el valor de bytes devuelve una cadena de la versión de texto del correo electrónico ❷. Usar get_payload() y decode() con la parte html del mensaje devuelve una cadena de la versión HTML del correo ❹.

Eliminación de correos electrónicos

Para borrar correos electrónicos, pase una lista de UIDs de mensajes al método delete_messages() del objeto IMAPClient. Esto marca los correos electrónicos con la bandera *\Deleted*. Si se llama al método expunge(), se eliminarán permanentemente todos los mensajes con el indicador
\Borrado en la carpeta seleccionada actualmente. Considere el siguiente interactivo ejemplo de concha:

```
>>> ❶imapObj.select_folder('INBOX', READONLY=FALSE)

>>> ❷UIDs= imapObj.search(['ON 09-Jul-2015'])
    >>> UIDs
    [40066]
    >>> imapObj.DELETE_MESSAGES(UIDs)
➲   {40066: ('visto', 'borrado')}
    >>> imapObj.expunge()
    ('Éxito', [(5452, 'EXISTS')])
```

Aquí seleccionamos la bandeja de entrada llamando a select_folder() en el objeto IMAPClient y pasando "INBOX" como primer argumento; también pasamos el argumento de palabra clave readonly=False para poder borrar correos❶. Buscamos en la bandeja de entrada los mensajes recibidos en una fecha específica y almacenamos los IDs de los mensajes devueltos en UIDs

❷. Si se llama a delete_message() y se le pasan los UID, se obtiene un diccionario; cada par clave-valor es un ID de mensaje y una tupla de los indicadores del mensaje, que ahora deben incluir❸. Al llamar a expunge() se borran permanentemente los mensajes con el indicador *\Deleted* y se devuelve un mensaje de éxito si no ha habido problemas.
borrar los mensajes. Tenga en cuenta que algunos proveedores de correo electrónico, como Gmail, eliminan automáticamente los mensajes borrados con delete_messages() en lugar de esperar a que el cliente IMAP emita un comando de eliminación.

Desconexión del servidor IMAP

Cuando su programa haya terminado de recuperar o borrar correos, simplemente llame al método `logout()` del IMAPClient para desconectarse del servidor IMAP.

```
>>> imapObj.logout()
```

Si el programa se ejecuta durante varios minutos o más, es posible que el servidor IMAP *agote el tiempo de espera* o se desconecte automáticamente. En este caso, la siguiente llamada a un método del objeto `IMAPClient` lanzará una excepción como la siguiente:

```
imaplib.abort: error de socket: [WinError 10054] Una conexión existente fue
cerrada a la fuerza por el host remoto
```

En este , su programa tendrá que llamar a `imapclient.IMAPClient()` para conectarse de nuevo.

¡Uf! Ya está. Había un montón de aros por los que pasar, pero ahora tienes una forma de hacer que tus programas Python se conecten a una cuenta de correo y obtengan correos electrónicos. Siempre puedes consultar la descripción general en Recuperar y borrar correos electrónicos con IMAP cuando necesites recordar todos los pasos.

Proyecto: Envío de recordatorios por correo electrónico

Digamos que te han "ofrecido voluntario" para hacer el seguimiento de las cuotas de los socios del Club de Voluntariado Obligatorio. Se trata de un trabajo realmente aburrido, que implica mantener una hoja de cálculo con todos los que han pagado cada mes y enviar recordatorios por correo electrónico a los que no lo han hecho. En lugar de revisar la hoja de cálculo tú mismo y copiar y pegar el mismo correo electrónico a todos los que están atrasados en el pago de las cuotas, vamos -lo has adivinado- a escribir un guión que lo haga por ti.

A grandes , esto es lo que hará su programa:

- Leer datos de una hoja de cálculo Excel.
- Buscar a todos los afiliados que no hayan pagado las cuotas del último mes.
- Busca sus direcciones de correo electrónico y envíales recordatorios

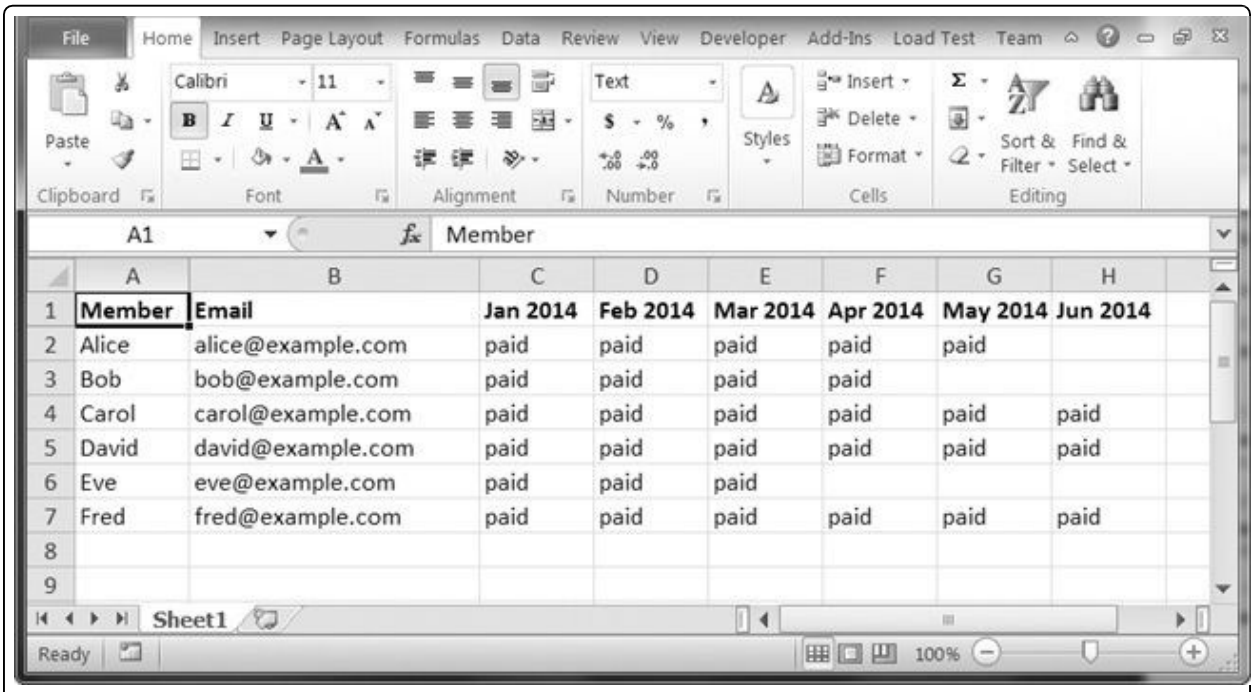
personalizados. Esto significa que su código tendrá que hacer lo siguiente:

- Abra y lea las celdas de un documento Excel con el módulo `openpyxl`. (Véase el Capítulo 12 para trabajar con archivos Excel).
- Crear un diccionario de los miembros que están atrasados en el pago de sus cuotas.
- Inicie sesión en un servidor SMTP llamando `a smtplib.SMTP()`, `ehlo()`, `starttls()` y `login()`.
- Para todos los miembros atrasados en el pago de sus cuotas, envíe un correo electrónico recordatorio personalizado llamando al método `sendmail()`.

Abre una nueva ventana del editor de archivos y guárdalo como *sendDuesReminders.py*.

Paso 1: Abrir el archivo Excel

Supongamos que la hoja de cálculo Excel que utiliza para realizar el seguimiento de los pagos de las cuotas de los socios se parece a la Figura 16-2 y se encuentra en un archivo denominado *duesRecords.xlsx*. Puede descargar este archivo de http://nostarch.com/automatestuff/.



	A	B	C	D	E	F	G	H
1	**Member**	**Email**	**Jan 2014**	**Feb 2014**	**Mar 2014**	**Apr 2014**	**May 2014**	**Jun 2014**
2	Alice	alice@example.com	paid	paid	paid	paid	paid	
3	Bob	bob@example.com	paid	paid	paid	paid		
4	Carol	carol@example.com	paid	paid	paid	paid	paid	paid
5	David	david@example.com	paid	paid	paid	paid	paid	paid
6	Eve	eve@example.com	paid	paid	paid			
7	Fred	fred@example.com	paid	paid	paid	paid	paid	paid
8								
9								

Figura 16-2. Hoja de cálculo para el seguimiento del pago de las cuotas de los afiliados

Esta hoja de cálculo contiene el nombre y la dirección de correo electrónico de cada afiliado. Cada mes hay una columna con el estado de pago de los afiliados. La celda correspondiente a cada afiliado se marca con el texto *pagado* una vez que ha abonado su cuota.

El programa tendrá que abrir *duesRecords.xlsx* y averiguar la columna del último mes llamando al método get_highest_column(). (Puedes consultar el capítulo 12 para más información sobre el acceso a celdas en archivos de hojas de cálculo Excel con el módulo openpyxl). Introduce el siguiente código en la ventana del editor de archivos:

```
#! python3
    # sendDuesReminders.py - Envía correos electrónicos basados en el estado del pago en la hoja de

    cálculo. import openpyxl, smtplib, sys

    # Abre la hoja de cálculo y obtén el último estado de las cuotas.
❶ wb= openpyxl.load_workbook('duesRecords.xlsx')
❷ hoja= wb.get_sheet_by_name("Hoja1")
```

```
➲   últimaCol= hoja.get_columna_más_alta()
❹ latestMonth= sheet.cell(row=1, column=lastCol).value #

    TODO: Comprobar el estado de pago de cada miembro.

     # TODO: Entrar en la cuenta de correo.

       # TODO: Enviar emails recordatorios.
```

Después de importar los módulos openpyxl, smtplib y sys, abrimos nuestro archivo *duesRecords.xlsx* y almacenamos el objeto Workbook resultante en wb❶. A continuaciónobtenemos la Hoja 1 y almacenamos el objeto Worksheet resultante en sheet ❷. Ahora que tenemos un objeto Hoja de trabajo, podemos acceder a filas, columnas y celdas. Almacenamos las
columna más alta en lastCol❸, y a continuación utilizamos la fila número 1 y lastCol para acceder a la celda que debe contener el mes más reciente. Obtenemos el valor de esta celda y lo almacenamos en latestMonth❹.

Paso 2: Buscar a todos los afiliados impagados

Una vez que haya determinado el número de columna del último mes (almacenado en lastCol), puede recorrer todas las filas después de la primera (que tiene los encabezados de columna) para ver qué socios tienen el texto *pagado* en la celda correspondiente a las cuotas de ese mes. Si el socio no ha pagado, puede obtener su nombre y dirección de correo electrónico de las columnas 1 y 2, respectivamente. Esta información irá al diccionario unpaidMembers, que hará un seguimiento de todos los socios que no han pagado en el mes más reciente. Añade el siguiente código a *sendDuesReminder.py*.

```
#! python3
    # sendDuesReminders.py - Envía correos electrónicos en función del estado del pago en la hoja de cálculo.

    --snip--

    # Comprueba EL ESTADO DE pago de cada miembro.
    UNPAIDMEMBERS= {}
❶ for r in range(2, sheet.GET_HIGHEST_ROW()+ 1):
❷       pago= sheet.cell(row=r, COLUMN=LASTCOL).value if pago
        != 'pagado':
➲           nombre= hoja.celda(fila=r, columna=1).valor
❸           email= sheet.cell(row=r, column=2).value
❹           UNPAIDMEMBERS[name]= email
```

Este código establece un diccionario vacío unpaidMembers y luego realiza un bucle a través de todas las filas después de la primera❶. Para cada , el valor de la columna más reciente se almacena en pago❷. Si el pago no es igual a 'pagado', entonces el valor de la primera columna se almacena en nombre❸, el valor de la segunda columna se almacena en email❹, y nombre y email se añaden a unpaidMembers❺.

Paso 3: Enviar recordatorios personalizados por correo electrónico

Una vez que tengas una lista de todos los miembros impagados, es hora de enviarles recordatorios por correo electrónico. Añade el siguiente código a tu programa, excepto con tu dirección de correo electrónico real y la información del proveedor:

```
#! python3
# sendDuesReminders.py - Envía correos electrónicos en función del estado del pago en la hoja de cálculo.

--snip--

# Iniciar sesión en la cuenta de correo electrónico.
smtpObj= smtplib.SMTP('smtp.gmail.com', 587)
smtpObj.ehlo()
smtpObj.STARTTLS()
smtpObj.login('my_email_address@gmail.com ', SYS.argv[1])
```

Crea un objeto SMTP llamando a smtplib.SMTP() y pasándole el nombre de dominio y el puerto de tu proveedor. Llama a ehlo() y starttls(), y luego llama a login() y pásale tu dirección de correo electrónico y sys.argv[1], que almacenará tu cadena de contraseña. Introducirás la contraseña como argumento de línea de comandos cada vez que ejecutes el , para evitar guardar tu contraseña en tu código fuente.

Una vez que su programa haya iniciado sesión en su cuenta de correo electrónico, debería revisar el diccionario de miembros impagados y enviar un correo electrónico personalizado a la dirección de correo electrónico de cada miembro. Añade lo siguiente a *sendDuesReminders.py*:

```
#! python3
    # sendDuesReminders.py - Envía correos electrónicos en función del estado del pago en la hoja de cálculo.

    --snip--

    # Enviar CORREOS ELECTRÓNICOS recordatorios.
    para nombre, email en MIEMBROSNOPAGADOS.ITEMS():
❶ body= "Asunto: %S CUOTAS impagadas.\nEstimado %S,\nLos REGISTROS muestran que
    no ha pagado las CUOTAS de %S. Por favor, realice EL pago LO antes POSIBLE. " %
    (últimoMes, nombre, últimoMes)

    ❷ print('Enviando email a %S...' % email)

➲       SENDMAILSTATUS=
        smtpObj.sendmail('my_email_address@gmail.com', email,
        body)

❹       si SENDMAILSTATUS != {}:
            print('HUBO un problema al enviar el correo electrónico a %S: %S' % (email,
            SENDMAILSTATUS))
    smtpObj.quit()
```

Este código hace un bucle a través de los nombres y correos electrónicos en unpaidMembers. Para cada miembro que no ha pagado, personalizamos un mensaje con el último mes y el nombre del miembro, y almacenamos el mensaje en el cuerpo❶. Imprimimos una salida diciendo que estamos enviando un correo electrónico a la dirección de correo electrónico de este ❷. Luego llamamos a sendmail(), pasándole la dirección from y el mensaje personalizado ❸. Almacenamos el valor de retorno en sendmailStatus.

Recuerde que el método sendmail() devolverá un valor de diccionario no vacío si el servidor SMTP informó de un error al enviar ese correo electrónico en particular. La última parte del bucle for en ❹ comprueba si el diccionario devuelto no está vacío, y si lo está, imprime la dirección de correo electrónico del destinatario y el diccionario devuelto.

Una vez que el programa ha terminado de enviar todos los correos, se llama al método quit() para desconectarse del servidor SMTP.

Cuando ejecutes el programa, la salida será algo parecido a esto:

```
Enviar correo electrónico a
alice@example.com... Enviar
correo electrónico a
bob@example.com...
Enviando correo electrónico a eve@example.com...
```

Los destinatarios recibirán un correo electrónico parecido al de la Figura 16-3.



Figura 16-3. Un correo electrónico enviado automáticamente desde sendDuesReminders.py

Envío de mensajes de texto con Twilio

La mayoría de las personas suelen estar más cerca de sus teléfonos que de sus ordenadores, por lo que los mensajes de texto pueden ser una forma más inmediata y fiable de enviar notificaciones que el correo electrónico. Además, la brevedad de los mensajes de texto hace más probable que una persona llegue a leerlos.

En esta sección, aprenderás a registrarte en el servicio gratuito Twilio y a utilizar su módulo Python para enviar mensajes de texto. Twilio *es un servicio de pasarela SMS*, lo que significa que es un servicio que te permite enviar mensajes de texto desde tus programas. Aunque estarás limitado en el número de textos que puedes enviar al mes y los textos irán precedidos de las palabras *Enviado desde una cuenta de prueba de Twilio*, este servicio de prueba es probablemente adecuado para tus programas personales. La prueba gratuita es indefinida; no tendrá que cambiar a un plan de pago más adelante.

Twilio no es el único servicio de pasarela SMS. Si prefieres no utilizar Twilio, puedes encontrar servicios alternativos buscando en línea *pasarela sms gratuita*, *python sms api*, o incluso *alternativas a twilio*.

Antes de registrarte en una cuenta Twilio, instala el módulo `twilio`. El Apéndice A contiene más información sobre la instalación de módulos de terceros.

NOTA

Esta sección es específica para los Estados Unidos. Twilio ofrece servicios de mensajes de texto SMS para países fuera los Estados Unidos, pero esos detalles no se tratan en este libro. Sin embargo, el módulo `twilio` *y sus funciones funcionarán igual fuera de Estados Unidos. Consulta* http://twilio.com/ *para obtener más información.*

Suscripción a una cuenta Twilio

Visita *http://twilio.com/* y rellena el formulario de registro. Una vez que te registrado, tendrás que verificar el número de teléfono móvil al que quieres enviar los mensajes. (Esta verificación es necesaria para evitar que la gente utilice el servicio para enviar mensajes de texto a números de teléfono aleatorios).

Tras recibir el mensaje con el número de verificación, introdúzcalo en el sitio web de Twilio para demostrar que es el propietario del teléfono móvil que está verificando. Ahora podrá enviar mensajes de texto a este número de teléfono utilizando el módulo `twilio`.

Twilio proporciona a su cuenta de prueba un número de teléfono para utilizarlo como remitente de los mensajes de texto. Necesitarás dos datos más: el SID de tu cuenta y el token de autenticación. Puede encontrar esta información en la página Panel cuando haya iniciado sesión en su cuenta Twilio. Estos valores actúan como tu nombre de usuario y contraseña de Twilio cuando inicias sesión desde un programa Python.

Envío de mensajes de texto

Una vez que hayas instalado el módulo twilio, te hayas registrado para obtener una cuenta Twilio, hayas verificado tu número de teléfono, hayas registrado un número de teléfono Twilio, y hayas obtenido el SID de tu cuenta y el auth token, por fin estarás listo para enviarte mensajes de texto desde tus scripts Python.

Comparado con todos los pasos de registro, el código Python es bastante sencillo. Con el ordenador conectado a Internet, introduzca lo siguiente en el shell interactivo, sustituyendo los valores de las variables accountSID, authToken, myTwilioNumber y myCellPhone por su información real:

```
>>> ❶from twilio.rest import TwilioRestClient
    >>> accountSID= 'ACxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
    >>> authToken= 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
>>> ❷twilioCli= TwilioRestClient(accountSID, authToken)
    >>> myTwilioNumber= '+14955551234'
    >>> miTeléfonoCelular= '+14955558888'
➲   >>> mensaje= twilioCli.messages.create(body='Sr. Watson - Venga aquí - Quiero .',
    from_=myTwilioNumber, to=myCellPhone)
```

Unos instantes después de escribir la última línea, deberías recibir un mensaje de texto que diga *Enviado desde tu cuenta de prueba Twilio - Sr. Watson - Ven aquí - Quiero verte*.

Debido a la forma en que el módulo twilio está , necesitas importarlo usando from twilio.rest import TwilioRestClient, no sólo import twilio❶. Almacena el SID de tu cuenta en accountSID y tu auth token en authToken y luego llama a TwilioRestClient() y pásale accountSID y authToken. La llamada a TwilioRestClient() devuelve un objeto TwilioRestClient❷. Este objeto tiene un atributo messages, que a su vez tiene un método create() que puedes utilizar para enviar mensajes de texto. Este es el método que instruirá a los servidores de Twilio para enviar su mensaje de texto. Después de almacenar tu número de Twilio y tu número de teléfono móvil en myTwilioNumber y myCellPhone, respectivamente, llama a create() y pásale argumentos de palabras clave que especifiquen el cuerpo del mensaje de texto, el número del remitente (myTwilioNumber) y el número del destinatario (myCellPhone)❸.

El objeto Message devuelto por el método create() tendrá información sobre el mensaje de texto enviado. Continúe con el ejemplo de shell interactivo introduciendo lo siguiente:

```
>>> mensaje.to
```

```
'+14955558888'
>>> mensaje.from_ '+14955551234'
>>> cuerpo.mensaje
"Sr. Watson, venga aquí, quiero ".
```

Los atributos to, from_ y body deben contener tu número de móvil, número Twilio y mensaje, respectivamente. Ten en cuenta que el número de teléfono de envío está en el atributo from_ -con un guión bajo al final- y no en from. Esto se debe a que from es una palabra clave en Python (la has visto utilizada en la forma from modulename import * de la sentencia import, por ejemplo), por lo que no puede utilizarse como nombre de atributo. Continúa el ejemplo del shell interactivo con lo siguiente:

```
>>> ESTADO.mensaje
en cola
>>> mensaje.fecha_creación
datetime.datetime(2015, 7, 8, 1, 36, 18)
>>> mensaje.fecha_envio== Ninguno
Verdadero
```

El atributo status debe proporcionar una cadena. Los atributos date_created y date_sent deben proporcionar un objeto datetime si el mensaje se ha creado y enviado. Puede parecer extraño que el atributo status tenga el valor 'queued' y que el atributo date_sent tenga el valor None cuando ya has recibido el mensaje de texto. Esto se debe a que usted capturó el objeto Message en la variable message *antes de que* el texto fuera realmente enviado. Tendrás que recuperar el objeto Message para ver su estado más actualizado y la fecha de envío. Cada mensaje de Twilio tiene un identificador de cadena único (SID) que se puede utilizar para obtener la última actualización del objeto Mensaje. Continúe con el ejemplo interactivo introduciendo lo siguiente:

```
>>> mensaje.sid
    'SM09520de7639ba3af137c6fcb7c5f4b51'
>>> ❶updatedMessage= twilioCli.MESSAGES.get(message.sid)
    >>> UPDATEDMESSAGE.STATUS
    "entregado
    >>> UPDATEDMESSAGE.date_sent
    datetime.datetime(2015, 7, 8, 1, 36, 18)
```

Ingresando message.sid te muestra el SID largo de este mensaje. Pasando este SID al método get() del cliente Twilio ❶, puedes recuperar un nuevo objeto Mensaje con la información más actualizada. En este nuevo objeto Mensaje, los atributos estado y fecha_enviado son correctos.

El atributo de estado se establecerá con uno de los siguientes valores de cadena: 'en cola', 'enviando', 'enviado', 'entregado', 'no entregado' o 'fallido'. Estos estados se explican por sí mismos, pero para obtener detalles más precisos, eche un vistazo a los recursos en http://nostarch.com/automatestuff/.

RECIBIR MENSAJES DE TEXTO CON PYTHON

Por desgracia, recibir mensajes de texto con Twilio es un poco más complicado que enviarlos. Twilio requiere que tengas un sitio web que ejecute su propia aplicación web. Eso está fuera del alcance de este libro, pero puedes encontrar más detalles en los recursos para este libro (http://nostarch.com/automatestuff/).

Proyecto "Módulo "Just Text Me

La persona a la que más a menudo enviarás mensajes de texto desde tus programas probablemente seas tú. Los mensajes de texto son una forma estupenda de enviarte notificaciones cuando estás lejos del ordenador. Si has automatizado una tarea aburrida con un programa que tarda un par de horas en ejecutarse, puedes hacer que te avise con un mensaje de texto cuando haya terminado. O puede que tengas un programa que se ejecuta con regularidad y que a veces necesita ponerse en contacto contigo, como un programa que comprueba el tiempo y te envía un mensaje de texto para recordarte que lleves un paraguas.

Como ejemplo sencillo, aquí tienes un pequeño programa Python con una función `textmyself()` que envía un mensaje que se le pasa como argumento de cadena. Abre una nueva ventana del editor de archivos e introduce el siguiente código, sustituyendo el SID de la cuenta, el auth token y los números de teléfono por tu propia información. Guárdalo como *textMyself.py*.

```
#! python3
    # textMyself.py - Define la función textmyself() que escribe un mensaje #
    pasado como una cadena.

    # Valores preestablecidos:
    accountSID= 'ACxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
    authToken= 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx' myNumber=
    '+15559998888'
    twilioNumber= '+15552225678'

    from twilio.rest import TwilioRestClient

❶ def textmyself(mensaje):
❷       twilioCli= TwilioRestClient(accountSID, authToken)
➲       twilioCli.messages.create(body=mensaje,    from_=twilioNumber,    to=miNúmero)
```

Este programa almacena un SID de cuenta, un auth token, un número de envío y un número de recepción. A continuación, define `textmyself()` para que tome el argumento❶, cree un objeto `TwilioRestClient`❷, y llame a `create()` con el mensaje que le pasaste.
❸.

Si quieres que la función `textmyself()` esté disponible para tus otros programas, simplemente coloca el archivo *textMyself.py* en la misma carpeta que el ejecutable de Python (*C:\Python34* en Windows, *usrlocal/lib/python3.4* en OS X, y *usrbin/python3* en Linux). Ahora puedes utilizar la función en tus otros programas. Siempre que quieras que uno de tus programas te envíe un mensaje de texto, sólo tienes que añadir lo siguiente:

```
importar textmyself
textmyself.textmyself('La tarea aburrida ha terminado.')
```

Tienes que registrarte en Twilio y escribir el código de envío de SMS una sola vez. Después de eso, son sólo dos líneas de código para enviar un texto desde cualquiera de sus otros programas.

Resumen

Nos comunicamos por Internet y a través de redes de telefonía móvil de decenas de formas distintas, pero predominan el correo electrónico y los mensajes de texto. Tus programas pueden comunicarse a través de estos canales, lo que les proporciona nuevas y potentes funciones de notificación. Incluso puedes escribir programas que se ejecuten en distintos ordenadores y que se comuniquen entre sí directamente a través del correo electrónico, de forma que un programa envíe correos con SMTP y el otro los recupere con IMAP.

smtplib de Python proporciona funciones para utilizar el SMTP para enviar correos electrónicos a través del servidor SMTP de tu proveedor de correo electrónico. Del mismo modo, los módulos de terceros imapclient y pyzmail te permiten acceder a servidores IMAP y recuperar correos electrónicos que hayan enviado.
Aunque IMAP es un poco más complicado que SMTP, también es bastante potente y le permite buscar correos electrónicos concretos, descargarlos y analizarlos para extraer el asunto y el cuerpo como valores de cadena.

El envío de mensajes de texto es un poco diferente del correo electrónico, ya que, a diferencia de éste, se necesita algo más que una conexión a Internet para enviar textos SMS. Afortunadamente, servicios como Twilio ofrecen módulos que te permiten enviar mensajes de texto desde tus programas. Una vez realizado el proceso de configuración inicial, podrás enviar mensajes de texto con sólo un par de líneas de código.

Con estos módulos en su conjunto de habilidades, usted será capaz de programar las condiciones específicas en las que sus programas deben enviar notificaciones o recordatorios. Ahora tus programas tendrán un alcance mucho mayor que el ordenador en el que se ejecutan.

Preguntas prácticas

P: 1. ¿Cuál es el protocolo para enviar correo electrónico? ¿Para comprobar y recibir correo electrónico?

P: 2. ¿Qué cuatro funciones/métodos `smtplib` debe llamar para iniciar sesión en un servidor SMTP?

P: 3. ¿Qué dos funciones/métodos de `imapclient` debe llamar para iniciar sesión en un servidor IMAP?

P: 4. ¿Qué tipo de argumento se pasa a `imapObj.search()`?

P: 5. ¿Qué haces si tu código recibe un mensaje de error que dice `que tiene más de 10000 bytes`?

P: 6. El módulo `imapclient` maneja la conexión a un servidor IMAP y la búsqueda de correos electrónicos. Cual es el modulo que se encarga de leer los correos que `imapclient` recolecta?

P: 7. ¿Qué tres datos necesita Twilio para poder enviar mensajes de texto?

Proyectos prácticos

Para practicar, escribe programas que hagan lo siguiente.

Asignación aleatoria de tareas por correo electrónico

Escribe un programa que tome una lista de direcciones de correo electrónico de personas y una lista de tareas que deben realizarse y asigne tareas aleatoriamente a las personas. Envía por correo electrónico a cada persona las tareas asignadas. Si te sientes ambicioso, lleva un registro de las tareas asignadas anteriormente a cada persona para asegurarte de que el programa no asigne a nadie la misma tarea que hizo la última vez. Otra posibilidad es programar el programa para que se ejecute automáticamente una vez a la semana.

Una pista: si pasas una lista a la función random.choice(), ésta devolverá un elemento de la seleccionado al azar. Parte de tu código podría ser así:

```
tareas = ['platos', "baño", "aspiradora", 'pasear perro'] randomChore=
random.choice(tareas)
chores.remove(randomChore) # esta tarea ya está ocupada, así que elimínala
```

Recordatorio del paraguas

El capítulo 11 te enseñó a utilizar el módulo requests para obtener datos de http://weather.gov/. Escribe un programa que se ejecute justo antes de levantarte por la mañana y compruebe si va a llover ese día. Si es así, haz que el programa te envíe un mensaje de texto recordándote que te pongas un paraguas antes de salir de casa.

Baja automática

Escribe un programa que escanee tu cuenta de correo electrónico, encuentre todos los enlaces para darse de baja en todos tus correos y los abra automáticamente en un navegador. Este programa tendrá que iniciar sesión en el servidor IMAP de su proveedor de correo electrónico y descargar todos sus correos electrónicos. Puede utilizar BeautifulSoup (tratado en el Capítulo 11) para buscar cualquier instancia en la que aparezca la palabra *unsubscribe* dentro de una etiqueta de enlace HTML.

Una vez que tenga una lista de estas URL, puede utilizar webbrowser.open() para abrir automáticamente todos estos enlaces en un navegador.

Tendrás que completar manualmente los pasos adicionales para darte de baja de estas listas. En la mayoría de los casos, esto implica hacer clic en un enlace para confirmar.

Pero este script le ahorra tener que revisar todos sus correos electrónicos en busca de enlaces para darse de baja. Puedes pasar este script a tus amigos para que lo ejecuten en sus cuentas de correo electrónico. (Sólo asegúrate de que tu contraseña de correo electrónico no esté codificada en el código fuente).

Control del ordenador a través del correo electrónico

Escribe un programa que compruebe una cuenta de correo electrónico cada 15 minutos en busca de cualquier instrucción que le envíes y que ejecute esas instrucciones automáticamente. Por ejemplo, BitTorrent es un sistema de descarga peer-to-peer. Con un programa gratuito de BitTorrent, como qBittorrent, puedes descargar grandes archivos multimedia desde tu ordenador. Si envías por correo electrónico al programa un enlace BitTorrent (completamente legal, nada pirata), el programa comprobará su correo electrónico, encontrará este mensaje, extraerá el enlace y lanzará qBittorrent para empezar a descargar el . De esta forma, puede hacer que el ordenador de su casa comience la descarga mientras usted está fuera, y la descarga (completamente legal, para nada pirata) puede estar terminada para cuando usted vuelva a casa.

El capítulo 15 cubre cómo lanzar programas en su ordenador utilizando la función subprocess.Popen(). Por ejemplo, la siguiente llamada lanzaría el programa qBittorrent, junto con un archivo torrent:

```
qbProcess= subprocess.Popen(["C:Archivos de programa (x86)qBittorrent.exe',
"shakespeare_complete_works.torrent"])
```

Por supuesto, querrás que el programa se asegure de que los correos electrónicos proceden de ti. En particular, es posible que desee exigir que los correos electrónicos contengan una contraseña, ya que es bastante trivial para los hackers falsificar una dirección "de" en los correos electrónicos. El programa debería borrar los correos que encuentre para no repetir las instrucciones cada vez que compruebe la cuenta de correo. Como función adicional, haz que el programa te envíe un correo electrónico o un mensaje de texto de confirmación cada vez que ejecute un comando. Como no estará frente a la computadora que está ejecutando el programa, es una buena idea usar las funciones de registro (ver Capítulo 10) para escribir un registro en un archivo de texto que pueda revisar si aparecen errores.

qBittorrent (así como otras aplicaciones BitTorrent) tiene una función que le permite salir automáticamente una vez finalizada la descarga. El Capítulo 15 explica cómo puede determinar cuándo una aplicación iniciada se ha detenido con el método wait() para objetos Popen. La llamada al método wait() se bloqueará hasta que qBittorrent se haya detenido, y entonces su programa podrá enviarle un correo electrónico o un mensaje de texto notificándole que la descarga ha finalizado.

Hay un montón de posibles características que podrías añadir a este proyecto. Si te quedas atascado, puedes descargar un ejemplo de implementación de este programa en http://nostarch.com/automatestuff/.

Capítulo 17. Manipulación de imágenes Manipulación de imágenes

Si tienes una cámara digital o incluso si te limitas a subir fotos de tu teléfono a Facebook, probablemente te cruces todo el tiempo con archivos de imágenes digitales. Puede que sepas utilizar programas gráficos básicos, como Microsoft Paint o Paintbrush, o incluso aplicaciones más avanzadas como Adobe Photoshop. Pero si necesitas editar un número masivo de imágenes, editarlas a mano puede ser un trabajo largo y aburrido.

Entre en Python. Pillow es un módulo de Python para interactuar con archivos de imagen. El módulo tiene varias funciones que facilitan recortar, redimensionar y editar el contenido de una imagen. Con el poder de manipular imágenes de la misma manera que lo harías con software como Microsoft Paint o Adobe Photoshop, Python puede editar automáticamente cientos o miles de imágenes con facilidad.

Fundamentos de la imagen por ordenador

Con el fin de manipular una imagen, es necesario comprender los conceptos básicos de cómo los ordenadores tratan los colores y las coordenadas en las imágenes y cómo se puede trabajar con colores y coordenadas en Pillow. Pero antes de continuar, instale el módulo `pillow`. Consulte el Apéndice A para obtener ayuda sobre la instalación de módulos de terceros.

Colores y valores RGBA

Los programas informáticos suelen representar un color en una imagen como un *valor RGBA*. Un valor RGBA es un grupo de números que especifican la cantidad de rojo, verde, azul y *alfa* (o transparencia) de un color. Cada uno de estos valores componentes es un número entero de 0 (ninguno) a 255 (el máximo). Estos valores RGBA se asignan a *píxeles* individuales; un píxel es el punto más pequeño de un solo color que el que puede mostrar la pantalla del ordenador (como puedes imaginar, hay millones de píxeles en una pantalla). El valor RGB de un píxel le indica con precisión qué tono de color debe mostrar. Las imágenes también tienen un valor alfa para crear valores RGBA. Si una imagen se muestra en la pantalla sobre una imagen de fondo o un fondo de escritorio, el valor alfa determina qué parte del fondo se puede "ver a través" del píxel de la imagen.

En Pillow, los valores RGBA se representan mediante una tupla de cuatro valores enteros. Por ejemplo, el color rojo se representa por (255, 0, 0, 255). Este color tiene la máxima cantidad de rojo, sin verde ni azul, y el máximo valor alfa, lo que significa que es totalmente opaco. El verde se representa con (0, 255, 0, 255) y el azul con (0, 0, 255, 255). El blanco, la combinación de todos los colores, es (255, 255, 255, 255), mientras que el negro, que no tiene ningún color, es (0, 0, 0, 255).

Si un color tiene un valor alfa de 0, es invisible, y no importa realmente cuáles sean los valores RGB. Después de todo, el rojo invisible tiene el mismo aspecto que el negro invisible.

Pillow utiliza los nombres de color estándar que utiliza HTML. La Tabla 17-1 enumera una selección de nombres de colores estándar y sus valores.

Tabla 17-1. Nombres de colores estándar y sus valores RGBA

Nombre	Valor RGBA	Nombre	Valor RGBA
Blanco	(255, 255, 255, 255)	Rojo	(255, 0, 0, 255)
Verde	(0, 128, 0, 255)	Azul	(0, 0, 255, 255)
Gris	(128, 128, 128, 255)	Amarillo	(255, 255, 0, 255)
Negro	(0, 0, 0, 255)	Morado	(128, 0, 128, 255)

Pillow ofrece la función ImageColor.getcolor() para que no tengas que memorizar valores RGBA para los colores que quieras utilizar. Esta función toma un

como primer argumento, y la cadena 'RGBA' como segundo argumento, y devuelve una tupla RGBA.

> **COLORACIÓN CMYK Y RGB**
>
> En primaria aprendiste que mezclando pinturas rojas, amarillas y azules se pueden formar otros colores; por ejemplo, puedes mezclar azul y amarillo para hacer pintura verde. Esto se conoce como *modelo sustractivo del color* y se aplica a tintes, tintas y pigmentos. Por eso las impresoras en color tienen cartuchos de tinta *CMYK*: la tinta *cian* (azul), *magenta* (roja), *amarilla* y *negra* se pueden mezclar para formar cualquier color.
>
> Sin embargo, la física de la luz utiliza lo que se denomina un *modelo de color aditivo*. Al combinar la luz (como la que emite la pantalla de tu ordenador), los colores rojo, verde y azul pueden combinarse para formar cualquier otro color. Por eso los valores *RGB* representan el color en los programas informáticos.

Para ver cómo funciona esta función, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> ❶from PIL import ImageColor

>>> ❷ImageColor.getcolor('red', 'RGBA')
(255, 0, 0, 255)
➲  >>> ImageColor.getcolor('ROJO', 'RGBA')
(255, 0, 0, 255)
>>>  ImageColor.getcolor('Negro',  'RGBA')
(0, 0, 0, 255)
>>>  ImageColor.getcolor('chocolate',  'RGBA')
(210, 105, 30, 255)
>>> ImageColor.getcolor('Azul aciano', 'RGBA')
(100, 149, 237, 255)
```

Primero, necesitas importar el módulo ImageColor de PIL ❶ (no de Pillow; verás por qué en un momento). La cadena de nombre de color que pasas a ImageColor.getcolor() no distingue mayúsculas de minúsculas, así que pasar 'red' ❷ y pasar 'RED' ❸ te dan la misma tupla RGBA. También puede pasar un color más inusual nombres, como "chocolate" y "Cornflower Blue".

Pillow admite un gran número de nombres de color, desde "aliceblue" hasta "whitesmoke". Puedes consultar la lista completa de más de 100 nombres de colores estándar en los recursos de http://nostarch.com/automatestuff/.

Coordenadas y tuplas de casilla

Los píxeles una imagen se indican con coordenadas x e y, que especifican respectivamente su posición horizontal y vertical en la imagen. *El origen* es el píxel situado en la esquina superior izquierda de la imagen y se especifica con la notación (0, 0). El primer cero representa la coordenada x, que comienza en cero en el origen y aumenta de izquierda a derecha. El segundo cero representa la coordenada y, que empieza en cero en el origen y aumenta yendo de hacia abajo en la imagen. Conviene repetirlo: las coordenadas y aumentan en sentido descendente, que es lo contrario de lo que recuerda de las coordenadas y en clase de matemáticas. La Figura 17-1 muestra cómo funciona este sistema de coordenadas.

Muchas de las funciones y métodos de Pillow toman como argumento *una tupla de cajas*. Esto significa que Pillow espera una tupla de cuatro coordenadas enteras que representan una región rectangular en una imagen. Los cuatro enteros son, en orden, siguientes:

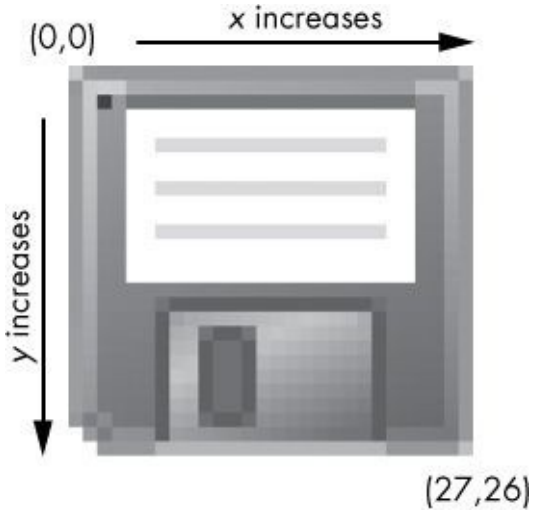


Figura 17-1. Las coordenadas x e y de una imagen de 27×26 de algún tipo de antiguo dispositivo de almacenamiento de datos.

- ***Izquierda:*** La coordenada x del borde izquierdo de la caja.
- ***Superior:*** La coordenada y del borde superior de la caja.
- ***Derecha:*** La coordenada x de un píxel a la derecha del borde más a la derecha de la caja. Este entero debe ser mayor que el entero izquierdo.
- ***Fondo:*** La coordenada y de un píxel por debajo del borde inferior de la caja. Este entero debe ser mayor que el entero superior.

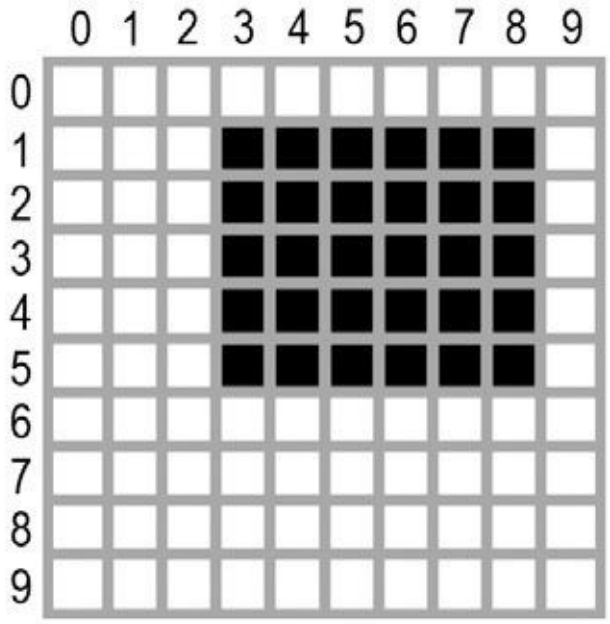


Figura 17-2. El área representada por la tupla de cajas (3, 1, 9, 6)

Tenga en cuenta que la caja incluye las coordenadas izquierda y superior y llega hasta las coordenadas derecha e inferior, pero no las incluye. Por ejemplo, la tupla caja `(3, 1, 9, 6)` representa todos los píxeles de la caja negra de la Figura 17-2.

Manipulación de imágenes con Pillow

Ahora que sabe cómo funcionan los colores y las coordenadas en Pillow, utilicemos Pillow para manipular una imagen. La Figura 17-3 es la imagen que se utilizará para todos los ejemplos de shell interactivo de este capítulo. Puedes descargarla de http://nostarch.com/automatestuff/.

Una vez que tengas el archivo de imagen *Zophie.png* en tu directorio de trabajo actual, estarás listo para cargar la imagen de Zophie en Python, así:

```
>>> from PIL import Imagen
>>> catIm= Image.open('zophie.png')
```



Figura 17-3. Mi gato Zophie. La cámara pesa 4 kilos (que es mucho para un gato).

Para cargar la imagen, importa el módulo Image de Pillow y llama a

Image.open(), pasándole el nombre de archivo de la imagen. A continuación, puede almacenar la imagen cargada en una variable como CatIm. El nombre del módulo de Pillow es PIL para hacerlo compatible con un módulo anterior llamado Python Imaging Library, por lo que debe ejecutar from PIL import Image en lugar de from Pillow importar Imagen. Debido a la forma en que los creadores de Pillow configuraron el módulo pillow, debe utilizar la forma from PIL import Image de la sentencia import, en lugar de simplemente import PIL.

Si el archivo de imagen no está en el directorio de trabajo actual, cambie el directorio de trabajo a la carpeta que contiene el archivo de imagen llamando a la función os.chdir().

```
>>> importar os
>>> os.chdir('C:\carpeta_con_archivo_de_imagen')
```

La función Image.open() devuelve un valor del tipo de datos objeto Image, que es como Pillow representa una imagen como un valor Python. Puede cargar un objeto Image desde un archivo de imagen (de cualquier formato) pasando a la función Image.open() una cadena con el nombre del archivo. Cualquier cambio que hagas en el objeto Image puede ser guardado
a un archivo de imagen (también de cualquier formato) con el método save(). Todas las rotaciones, redimensionamiento, recorte, dibujo y otras manipulaciones de la imagen se harán a través de llamadas a métodos de este objeto Imagen.

Para acortar los ejemplos de este capítulo, asumiré que ha importado el módulo Image de Pillow y que tiene la imagen Zophie almacenada en una variable llamada catIm. Asegúrese de que el archivo *zophie.png* está en el directorio de trabajo actual para la función Image.open() pueda . De lo contrario, también tendrá que especificar la ruta absoluta completa en el argumento string de Image.open().

Trabajar con el tipo de datos de imagen

Un objeto Imagen tiene varios atributos útiles que proporcionan información básica sobre el archivo de imagen desde el que se ha cargado: su anchura y altura, el nombre del archivo y el formato de los gráficos (como JPEG, GIF o PNG).

Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> from PIL import Imagen
    >>> catIm= Image.open('zophie.png')
    >>> tamaño.catIm
❶ (816, 1088)
>>> ❷anchura, altura= catIm.size
➲   >>> anchura
    816
>>> ❹altura
    1088
    >>> catIm.filename
    zophie.png
    >>> catIm.format
    PNG
    >>> catIm.format_description
    Gráficos de red portátiles
>>> ❺catIm.save('zophie.jpg')
```

Después de crear un objeto Imagen a partir de *Zophie.png* y almacenar el objeto Imagen en catIm, podemos ver que el atributo tamaño del objeto contiene una tupla de la anchura y la altura de la imagen en píxeles ❶. Podemos asignar los valores de la tupla a las variables anchura y altura ❷ para acceder con anchura ❸ y altura ❹ individualmente. El atributo filename describe el nombre del archivo original. Los atributos format y format_description son cadenas que describen el formato de imagen del archivo original (siendo format_description un poco más verboso).

Finalmente, llamando al método save() y pasándole 'zophie.jpg' guarda una nueva imagen con el nombre de archivo *zophie.jpg* en tu disco duro ❺. Pillow ve que la extensión del archivo es *.jpg* y guarda automáticamente la imagen utilizando el formato de imagen JPEG. Ahora deberías tener dos imágenes, *zophie.png* y *zophie.jpg*, en tu disco duro. Aunque estos archivos se basan en la misma imagen, no son idénticos debido a sus diferentes formatos.

Pillow también proporciona la función Image.new(), que devuelve un objeto Image -muy parecido a Image.open(), excepto que la imagen representada por el objeto de Image.new() estará en blanco. Los argumentos de Image.new() son los siguientes:

- La cadena 'RGBA', que establece el modo de color en RGBA. (Hay otros modos en los que este libro no entra).
- El tamaño, como una tupla de dos enteros de la anchura y la altura de la nueva imagen.
- El color de fondo con el que debe comenzar la imagen, como una tupla de cuatro enteros de un valor RGBA. Puede utilizar el valor de retorno de la función ImageColor.getcolor() para este argumento. Alternativamente, Image.new() también admite simplemente pasar la cadena del nombre del color estándar.

Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> from PIL import Imagen
>>> ❶im= Image.new('RGBA', (100, 200), 'purple')
    >>> im.save('imagenpúrpura.png')
        >>> ❷im2= Image.new('RGBA', (20, 20))
        >>> im2.save('imagentransparente.png')
```

Aquí creamos un objeto Imagen para una imagen de 100 píxeles de ancho y 200 píxeles de alto, con un fondo ❶ morado. Esta imagen se guarda en el archivo *purpleImage.png*. Volvemos a llamar a Image.new() para crear otro objeto Image, esta vez pasando (20, 20) para las dimensiones y nada para el color de fondo ❷. El negro invisible, (0, 0, 0, 0), es el color por defecto utilizado si no se especifica ningún argumento de color, por lo que la segunda imagen tiene un fondo transparente; guardamos este cuadrado transparente de 20×20 en *transparentImage.png*.

Recortar imágenes

Recortar una imagen significa seleccionar una región rectangular dentro de una imagen y eliminar todo lo que está fuera del rectángulo. El método crop() de los objetos Image toma una tupla box y devuelve un objeto Image que representa la imagen recortada. El recorte no se realiza in situ, es decir, el objeto Image original no se recorta.
se deja intacto, y el método crop() devuelve un nuevo objeto Image. Recuerde que una tupla recuadrada -en este caso, la sección recortada- incluye la columna izquierda y la fila superior de píxeles, pero sólo llega hasta la columna derecha y la fila inferior de píxeles, y *no* las incluye.

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> croppedIm= catIm.crop((335, 345, 565, 560))
>>> croppedIm.save('cropped.png')
```

Esto crea un nuevo objeto Image para la imagen recortada, almacena el objeto en croppedIm, y luego llama a save() sobre croppedIm para guardar la imagen recortada en *cropped.png*. El nuevo archivo *cropped.png* se creará a partir de la imagen original, como en la Figura 17-4.

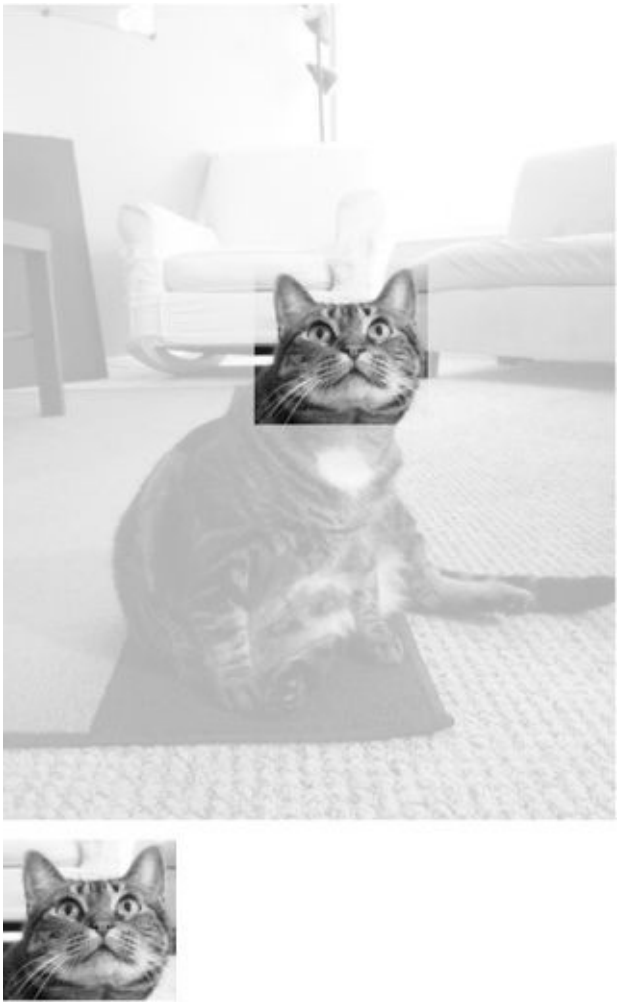


Figura 17-4. La nueva imagen será sólo la sección recortada de la imagen original.

Copiar y pegar imágenes en otras imágenes

El método copy() devolverá un nuevo objeto Image con la misma imagen que el objeto Image sobre el que fue invocado. Esto es útil si necesita hacer cambios en una imagen pero también quiere mantener una versión intacta de la original. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> catIm= Image.open('zophie.png')
>>> catCopyIm= catIm.copy()
```

Las variables catIm y catCopyIm contienen dos objetos Image distintos, ambos la misma imagen. Ahora que tiene un objeto Image almacenado en catCopyIm, puede modificar catCopyIm como desee y guardarlo en un nuevo nombre de archivo, dejando *zophie.png* intacto. Por ejemplo, intentemos modificar catCopyIm con el método paste().

El método paste() es llamado sobre un objeto Image y pega otra imagen sobre él. Continuemos con el ejemplo del shell pegando una imagen más pequeña sobre catCopyIm.

```
>>> faceIm= catIm.crop((335, 345, 565, 560))
>>> faceIm.size
(230, 215)
>>> catCopyIm.paste(caraIm, (0, 0))
>>> catCopyIm.paste(caraIm, (400, 500))
>>> catCopyIm.save('pegado.png')
```

Primero pasamos a crop() una tupla de caja para el área rectangular en *zophie.png* que contiene la cara de Zophie. Esto crea un objeto Image que representa un recorte de 230×215, que almacenamos en faceIm. Ahora podemos pegar faceIm en catCopyIm. El método paste() recibe dos argumentos: un objeto Image "fuente" y una tupla de las coordenadas x e y en las que se desea pegar la esquina superior izquierda de la fuente. Image en el objeto Image principal. Aquí llamamos a paste() dos veces en catCopyIm, pasando (0, 0) la primera y (400, 500) la segunda. Esto pega faceIm en catCopyIm dos veces: una con la esquina superior izquierda de faceIm en (0, 0) en catCopyIm, y otra con la esquina superior izquierda de faceIm en (400, 500).
Por último, guardamos la imagen catCopyIm modificada en *pasted.png*. La imagen pasted.png se parece a la Figura 17-5.

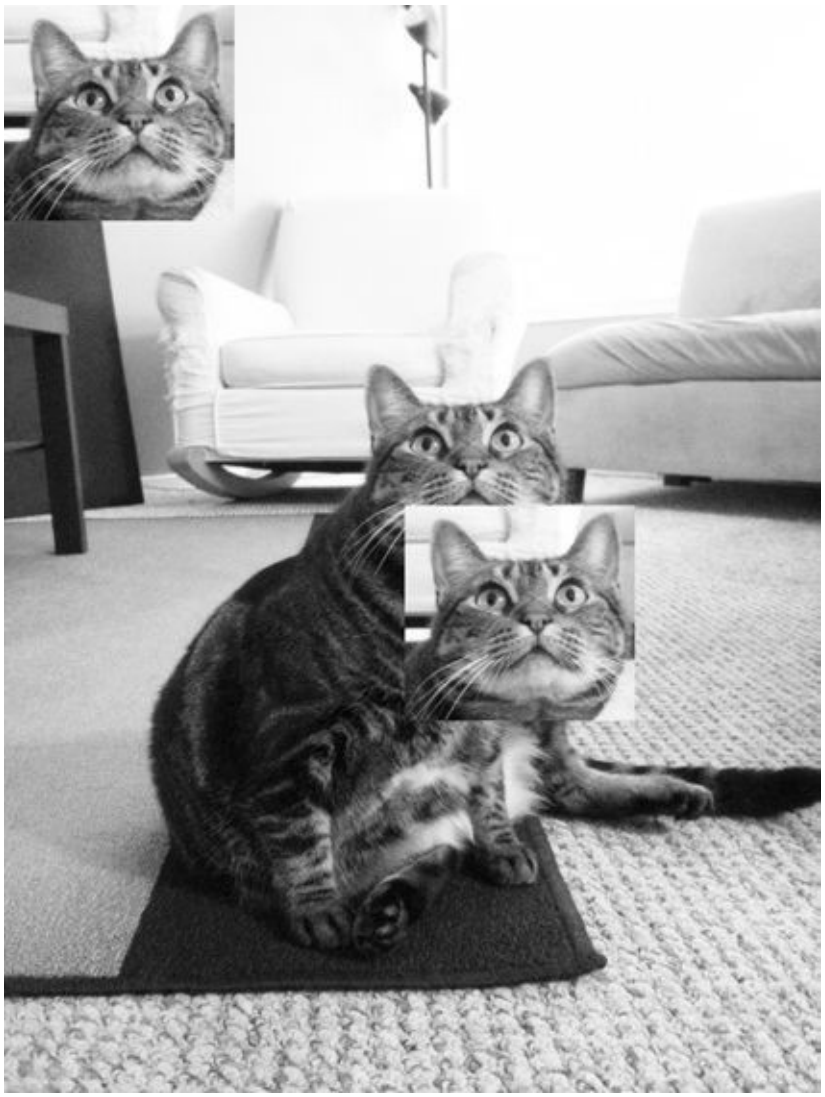


Figura 17-5. La gata Zophie, con su cara pegada dos veces

NOTA

A pesar de sus nombres, los métodos copy() *y* paste() *de Pillow no utilizan el portapapeles de su ordenador.*

Tenga en cuenta que el método paste() modifica su objeto Image *en el lugar*; no devuelve un objeto Image con la imagen pegada. Si desea llamar a pegar() pero también una versión intacta de la imagen original, tendrá que copiar primero la imagen y luego llamar a pegar() en esa copia.

Digamos que quieres poner en mosaico la cabeza de Zophie a lo largo de toda la imagen, como en la Figura 17-6. Puede lograr este efecto con sólo un par de bucles for. Continúe con el ejemplo del shell interactivo introduciendo lo siguiente:

```
>>> catImWidth, catImHeight= catIm.size
    >>> faceImWidth, faceImHeight= faceIm.size
❶    >>> catCopyDos= catIm.copy()

>>> ❷for left in range(0, catImWidth, faceImWidth):
➲              for top in range(0, catImHeight, faceImHeight):
                    print(izquierda, arriba)
                    catCopyTwo.paste(faceIm,    (izquierda,
                    arriba))
    0  0
    0  215
    0  430
    0  645
    0  860
    0  1075
    230 0
    230 215
    --snip--
    690 860
    690 1075
    >>> catCopyTwo.save('tiled.png')
```

Aquí almacenamos la anchura y la altura de catIm en catImWidth y catImHeight. En hacemos una copia de catIm y la almacenamos en catCopyTwo. Ahora que tenemos una copia en la que podemos pegar, empezamos el bucle para pegar faceIm en catCopyTwo. La variable izquierda del bucle for exterior comienza en 0 y aumenta en faceImWidth(230)❷. La variable superior del bucle for interior comienza en 0 y aumenta en faceImHeight(215)❸. Estos bucles for anidados producen valores para left y top para pegar una rejilla de imágenes faceIm sobre el objeto Image catCopyTwo, como en la Figura 17-6. Para ver cómo funcionan nuestros bucles anidados, imprimimos left y top. Una vez completado el pegado, guardamos el catCopyTwo modificado en *tiled.png*.



Figura 17-6. Bucles for *anidados utilizados con* paste() *para duplicar la cara del gato (un duplicado, si se quiere).*

PEGAR PÍXELES TRANSPARENTES

Normalmente, los píxeles transparentes se pegan como píxeles blancos. Si la imagen que desea pegar tiene píxeles transparentes, pase el objeto Imagen como tercer argumento para que no se pegue un rectángulo sólido. Este tercer argumento es el Imagen "máscara". Una máscara es un objeto Imagen donde el valor alfa es significativo, pero los valores rojo, verde y azul son ignorados. La máscara indica a la función paste() qué píxeles debe copiar y cuáles debe dejar transparentes. El uso avanzado de máscaras está más allá de este libro, pero desea pegar una imagen que tiene píxeles transparentes, pase el objeto Image de nuevo como tercer argumento.

Cambiar el tamaño de una imagen

El método resize() se ejecuta sobre un objeto Image y devuelve un nuevo objeto Image con la anchura y altura especificadas. Acepta una tupla de dos enteros como argumento, que representa la nueva anchura y altura de la imagen devuelta. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> ❶anchura, altura= catIm.size

>>> ❷quartersizedIm= catIm.RESIZE((int(anchura          2), int(altura       2)))
    >>>   QUARTERSIZEDIM.save('QUARTERSIZED.png')
➲   >>> svelteIm= catIm.RESIZE((anchura, altura+ 300))
    >>> SVELTEIM.save('SVELTE.png')
```

Aquí asignamos los dos valores de la tupla catIm.size a las variables width y height❶.

El uso de anchura y altura en lugar de catIm.size[0] y catIm.size[1] hace que el resto del código sea más legible.

La primera llamada a resize() pasa int(width / 2) para la nueva anchura e int(height / 2) para la nueva altura❷, por lo que el objeto Image devuelto por resize() tendrá la mitad de la longitud y anchura de la imagen original, o un cuarto del tamaño total de la imagen original. El método resize() sólo acepta enteros en su argumento de tupla, que es por lo que necesita envolver ambas divisiones por 2 en una llamada a int().

Este redimensionamiento mantiene las mismas proporciones para la anchura y la altura. Pero la nueva anchura y altura pasadas a resize() no tienen por qué ser proporcionales a la imagen original. La variable svelteIm contiene un objeto Imagen que tiene la anchura original pero una altura 300 píxeles mayor ❸, lo que da a Zophie un aspecto más esbelto.

Tenga en cuenta que el método resize() no edita el objeto Image en su lugar, sino que devuelve un nuevo objeto Image.

Girar y voltear imágenes

Las imágenes pueden rotarse con el método rotate(), que devuelve un nuevo objeto Image de la imagen rotada y deja el objeto Image original sin cambios. El argumento de rotate() es un único entero o flotante que representa el número de grados para rotar la imagen en sentido contrario a las agujas del reloj. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>>     catIm.rotate(90).save('rotated90.png')
>>> catIm.rotate(180).save('rotated180.png')
>>> catIm.rotate(270).save('rotated270.png')
```

Observe cómo puede *encadenar* llamadas a métodos llamando a save() directamente sobre el objeto Image devuelto por rotate(). La primera llamada a rotate() y save() crea un nuevo objeto Imagen que representa la imagen rotada 90 grados en sentido contrario a las agujas del reloj y guarda la imagen rotada en *rotated90.png*. La segunda y tercera llamada hacen lo mismo, pero con 180 y 270 . Los resultados son los siguientes
Figura 17-7.

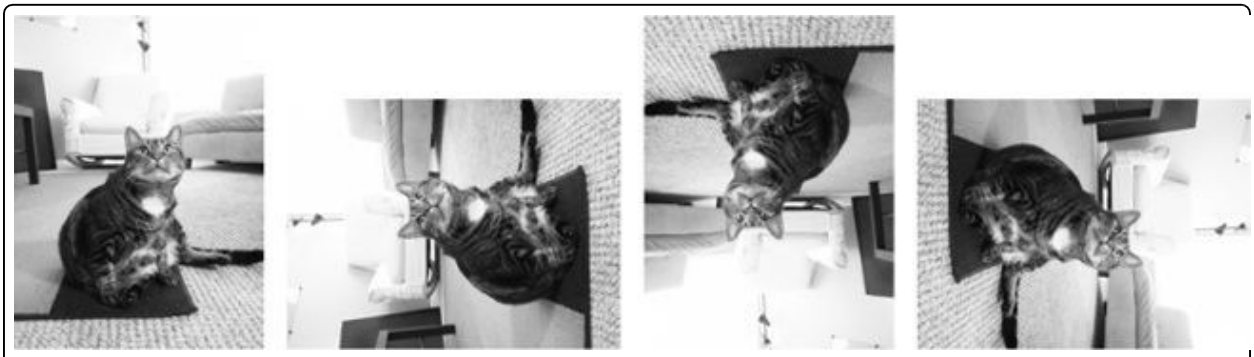


Figura 17-7. La imagen original (izquierda) y la imagen girada 90, 180 y 270 grados en el sentido contrario a las agujas del reloj.

Observe que la anchura y la altura de la imagen cambian cuando la imagen se gira 90 ó 270 grados. Si rota una imagen en otra cantidad, se mantienen las dimensiones originales de la imagen. En Windows, se utiliza un fondo negro para rellenar los huecos creados por la rotación, como en la Figura 17-8. En OS X, se utilizan píxeles transparentes para rellenar los huecos creados por la rotación. En OS X, se utilizan píxeles transparentes para rellenar los huecos.

El método rotate() tiene un argumento de palabra clave opcional expand que puede establecerse a True para ampliar las dimensiones de la imagen para que quepa toda la nueva imagen rotada. Por ejemplo, introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> catIm.rotate(6).save('rotated6.png')
```

```
>>> catIm.rotate(6, expand=True).save('rotated6_expanded.png')
```

La primera llamada rota la imagen 6 grados y la guarda en *rotate6.png* (ver la imagen de la izquierda de la Figura 17-8). La segunda llamada rota la imagen 6 grados con expand establecido a True y la guarda en *rotate6_expanded.png* (ver la imagen a la derecha de la Figura 17-8).

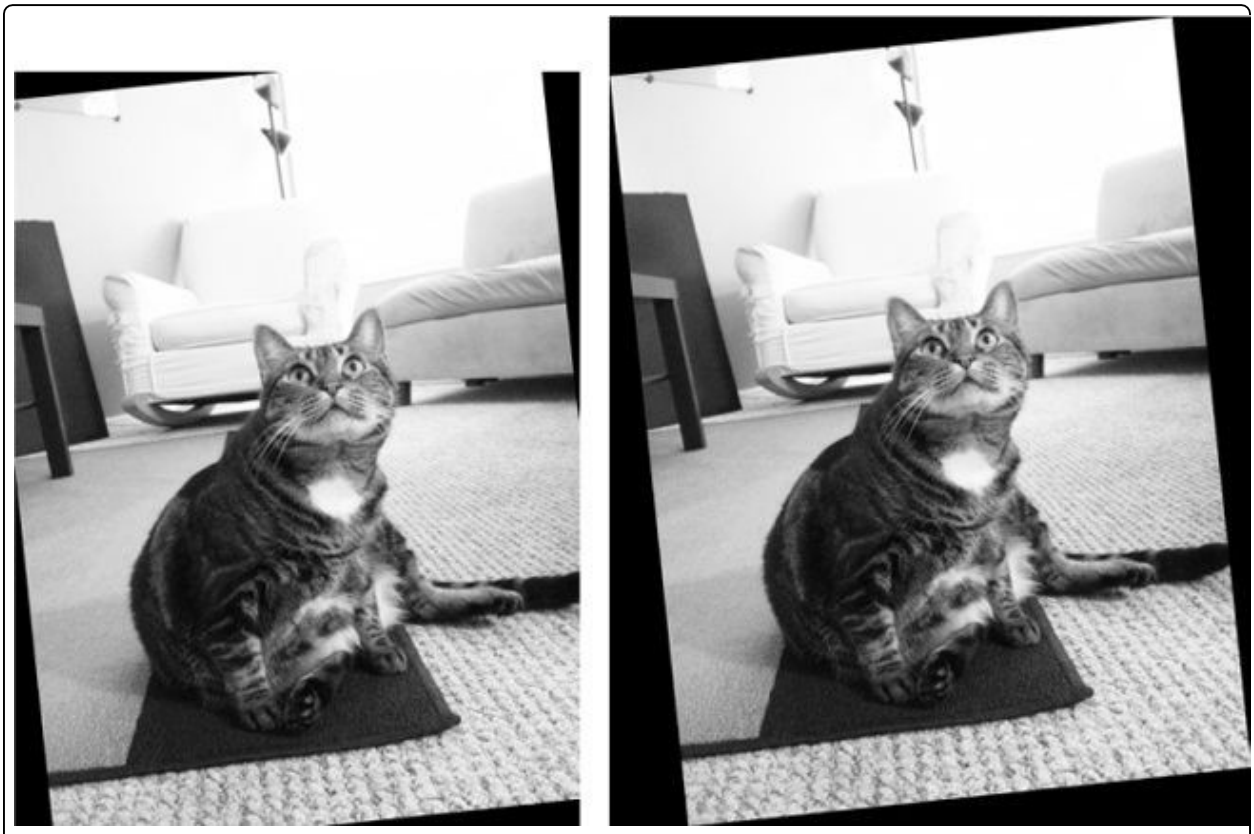


Figura 17-8. La imagen girada 6 grados normalmente (izquierda) y con expand=True *(derecha)*

También puede obtener una imagen invertida con el método transpose(). Debe pasar Image.FLIP_LEFT_RIGHT o Image.FLIP_TOP_BOTTOM al método transpose(). Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>>  catIm.TRANSPOSE(Image.FLIP_LEFT_RIGHT).save('horizontal_flip.png')
>>> catIm.TRANSPOSE(Image.FLIP_TOP_BOTTOM).save('vertical_flip.png')
```

Al igual que rotate(), transpose() crea un nuevo objeto Image. Aquí pasamos Image.FLIP_LEFT_RIGHT para voltear la imagen horizontalmente y luego guardamos el resultado en *horizontal_flip.png*. Para voltear la imagen verticalmente, pasamos Image.FLIP_TOP_BOTTOM y guardamos el resultado en *vertical_flip.png*. El resultado es como el de la Figura 17-9.



Figura 17-9. La imagen original (izquierda), volteada horizontalmente (centro) y verticalmente (derecha).

Cambiar píxeles individuales

El color de un píxel individual puede recuperarse o establecerse con los métodos `getpixel()` y `putpixel()`. Ambos métodos toman una tupla que representa las coordenadas x e y del píxel. El método `putpixel()` también toma una tupla adicional como argumento para el color del píxel. Este argumento de color es una tupla RGBA de cuatro enteros o una tupla RGB de tres enteros. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> ❶im= Image.new('RGBA', (100, 100))

>>> ❷im.getpixel((0, 0))
    (0, 0, 0, 0)
➲   >>> para x en range(100):
            para y en rango(50):
❹               im.putpixel((x, y), (210, 210, 210))
    >>> from PIL import ImageColor

>>> ❺para x en rango(100):
            para y en rango(50, 100):
❻               im.putpixel((x, y), ImageColor.getcolor('darkgray', 'RGBA'))
    >>> im.getpixel((0, 0))
    (210, 210, 210, 255)
    >>> im.getpixel((0, 50))
    (169, 169, 169, 255)
    >>> im.save('ponerPixel.png')
```

En ❶ hacemos una nueva imagen que es un cuadrado transparente de 100×100. Llamar a `getpixel()` en algunas coordenadas de esta imagen devuelve `(0, 0, 0, 0)` porque la imagen es transparente ❷. Para colorear los píxeles de esta imagen, podemos utilizar bucles `for` anidados para recorrer todos los píxeles de la mitad superior de la imagen ❸ y colorear cada uno de ellos.

pixel usando `putpixel()`❹. Aquí pasamos a `putpixel()` la tupla RGB `(210, 210, 210)`, un gris claro.

Digamos que queremos colorear la mitad inferior de la imagen en gris oscuro pero no conocemos la tupla RGB para el gris oscuro. El método `putpixel()` no acepta un nombre de color estándar como `'darkgray'`, así que tienes que usar `ImageColor.getcolor()` para obtener una tupla de color de `'darkgray'`. Recorre en bucle los píxeles de la mitad inferior de la imagen ❺ y pasa a `putpixel()` el valor de retorno de `ImageColor.getcolor()`.
❻, y ahora debería tener una imagen que es de color gris claro en su mitad superior y oscuro gris en la mitad inferior, como se muestra en la Figura 17-10. Puede llamar a `getpixel()` en algunas coordenadas para confirmar que el color en cualquier píxel dado es el esperado. Finalmente, guarde la imagen en *putPixel.png*.

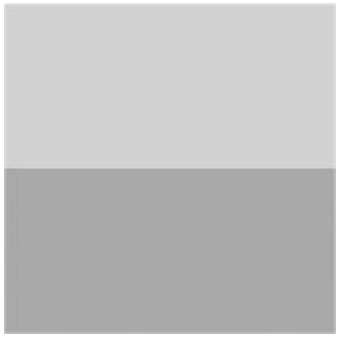


Figura 17-10. La imagen putPixel.png

Por supuesto, dibujar un píxel a la vez en una imagen no es muy conveniente. Si necesita dibujar formas, utilice las funciones ImageDraw que se explican más adelante en este capítulo.

Proyecto: Añadir un logotipo

Supongamos que tienes el aburrido trabajo de cambiar el tamaño de miles de imágenes y añadir una pequeña marca de agua con el logotipo en la esquina de cada una. Hacerlo con un programa gráfico básico como Paintbrush o Paint llevaría una eternidad. Una aplicación gráfica más sofisticada como Photoshop puede hacer el procesamiento por lotes, pero ese software cuesta cientos de dólares. Vamos a escribir un script para hacerlo en su lugar.

Digamos que la Figura 17-11 es el logotipo que desea añadir a la esquina inferior derecha de cada imagen: un icono de un gato negro con un borde blanco, con el resto de la imagen transparente.



Figura 17-11. El logotipo a añadir a la imagen.

A grandes , esto es lo que debe hacer el programa:

- Carga la imagen del logotipo.
- Recorre todos los archivos *.png* y *.jpg* del directorio de trabajo.
- Compruebe si la imagen es más ancha o más alta que 300 píxeles.
- Si es así, reduzca la anchura o la altura (la que sea mayor) a 300 píxeles y reduzca proporcionalmente la otra dimensión.
- Pega la imagen del logotipo en la esquina.
- Guarde las imágenes modificadas en otra carpeta.

Esto significa que el código tendrá que hacer lo siguiente:

- Abra el archivo *catlogo.png* como objeto Imagen.
- Recorre las cadenas devueltas por os.listdir('.').

- Obtiene la anchura y la altura de la imagen a partir del atributo size.
- Calcula la nueva anchura y altura de la imagen redimensionada.
- Llama al método resize() para cambiar el tamaño de la imagen.
- Llama al método paste() para pegar el logo.
- Llama al método save() para guardar los cambios, usando el nombre de archivo original.

Paso 1: Abrir la imagen del logotipo

Para este proyecto, abre una nueva ventana del editor de archivos, introduce el siguiente código y guárdalo como *resizeAndAddLogo.py*:

```
#! python3
    # resizeAndAddLogo.py - Redimensiona todas las imágenes del directorio de trabajo #
    actual para que quepan en un cuadrado de 300x300, y añade catlogo.png a la
    esquina inferior derecha.
    importar os
    from PIL import Imagen

❶ SQUARE_FIT_SIZE= 300
❷ LOGO_FILENAME= 'catlogo.png'

➲   logoIm= Image.open(LOGO_FILENAME)
❹ logoWidth, logoHeight= logoIm.size

    # TODO: Recorrer todos los ficheros del directorio de

    trabajo. # TODO: Comprobar si la imagen necesita ser

    redimensionada.

    # TODO: Calcular la nueva anchura y altura a redimensionar. #

    TODO: Redimensionar la imagen.

    # TODO: Añadir el logo.

    # TODO: Guardar cambios.
```

Al establecer las constantes SQUARE_FIT_SIZE ❶ y LOGO_FILENAME ❷ al inicio del programa, hemos facilitado su modificación posterior. Digamos que el logotipo que está añadiendo no es el icono del gato, o digamos que está reduciendo la dimensión más grande de las imágenes de salida a algo distinto de 300 píxeles. Con estas constantes al principio del programa, sólo tienes que abrir el código, cambiar esos valores una vez, y ya está. (O puedes hacer que los valores de estas constantes se tomen de los argumentos de la línea de comandos). Sin estas ,
tendría que buscar en el código todas las instancias de 300 y 'catlogo.png' y sustituirlas por los valores de su nuevo proyecto. En resumen, el uso de constantes hace que tu programa sea más generalizado.

El objeto Image del logotipo se devuelve desde Image.open()❸. Para facilitar la lectura,

logoWidth y logoHeight se asignan a los valores de logoIm.size❹. El resto del

programa es un esqueleto de comentarios TODO por ahora.

Paso 2: Recorrer todos los archivos y abrir las imágenes

Ahora necesita encontrar cada archivo *.png* y *.jpg* en el directorio de trabajo actual. Tenga en cuenta que no desea añadir la imagen del logotipo a la propia imagen del logotipo, por lo que el programa debe omitir cualquier imagen con un nombre de archivo que sea el mismo que LOGO_FILENAME. Añade lo siguiente a tu código:

```
#! python3
    # resizeAndAddLogo.py - Redimensiona todas las imágenes del directorio de trabajo #
    actual para que quepan en un cuadrado de 300x300, y añade catlogo.png a la
    esquina inferior derecha.

    importar os
    from PIL import Imagen

    --snip--

    os.makedirs("withLogo",    exist_ok=True)
    # Recorre todos los ARCHIVOS del directorio de trabajo.
❶ para nombre de archivo en OS.listdir('.'):
❷       if not (filename.ENDSWITH('.png') or filename.ENDSWITH('.jpg')) \ o
            nombre_de_archivo== LOGO_FILENAME:
➲            continue # omitir LOS ARCHIVOS que no sean de imagen y el propio archivo del logotipo

❹       im= Image.open(filename)
        anchura, altura= im.size
    --snip--
```

En primer lugar, la llamada a os.makedirs() crea una carpeta withLogo para almacenar las imágenes terminadas con logotipos, en lugar de sobrescribir los archivos de imagen originales. El argumento exist_ok=True evitará que os.makedirs() lance una excepción si *withLogo* ya existe. Mientras se recorren todos los ficheros del directorio de trabajo con os.listdir('.')❶, la sentencia larga if ❷ comprueba si cada nombre de fichero no termina en *.png* o *.jpg*. Si es así-o si el archivo es la propia imagen del logo-entonces el bucle debería saltárselo y usar continue ❸ para pasar al siguiente archivo. *Si* el nombre del archivo termina con '.png' o '.jpg' (y no es el archivo del logotipo), puede abrirlo como un objeto Imagen ❹ y establecer la anchura y la altura.

Paso 3: Cambiar el tamaño de las imágenes

El programa debe cambiar el tamaño de la imagen sólo si la anchura o la altura es mayor que SQUARE_FIT_SIZE (300 píxeles, en este caso), así que ponga todo el código de cambio de tamaño dentro de una sentencia `if` que compruebe las variables de `anchura` y `altura`. Añade el siguiente código a tu programa:

```
#! python3
    # resizeAndAddLogo.py - Redimensiona todas las imágenes del directorio de trabajo #
    actual para que quepan en un cuadrado de 300x300, y añade catlogo.png a la
    esquina inferior derecha.

    importar os
    from PIL import Imagen

    --snip--

        # Comprueba si la imagen necesita ser REDIMENSIONADA.
        si anchura> SQUARE_FIT_SIZE y altura> SQUARE_FIT_SIZE: #
            Calcula la nueva anchura y altura a la que
            redimensionar. if width> height:
❶               altura= int((SQUARE_FIT_SIZE / anchura) * altura)
                anchura= SQUARE_FIT_SIZE
            si no:
❷               anchura= int((SQUARE_FIT_SIZE / altura) * anchura)
                altura= SQUARE_FIT_SIZE

            # Redimensiona la imagen.
            print('Redimensionando %s...' % (nombre de fichero))
        ➲     im= im.RESIZE((anchura, altura))

--snip--
```

Si hay que cambiar el tamaño de la imagen, hay que averiguar si ancha o alta. Si `la anchura` es mayor que `la altura`, entonces la altura debe reducirse en la misma proporción en que se reduciría la anchura❶. Esta proporción es el valor `de SQUARE_FIT_SIZE` dividido por la anchura actual. El nuevo valor de `altura` es esta proporción multiplicada por el valor de `altura` actual. Dado que la división devuelve un valor flotante y `resize()` requiere que las dimensiones sean enteras, recuerde convertir el resultado a un entero con la función `int()`. Finalmente, el nuevo valor de `anchura` simplemente se ajustará a SQUARE_FIT_SIZE.

Si `la altura` es mayor o igual `que la anchura` (ambos casos se tratan en la función cláusula `else`), entonces se realiza el mismo cálculo, excepto con la `altura` y

variables de anchura intercambiadas❷.

Una vez que width y height contengan las nuevas dimensiones de la imagen, páselas a la función método resize() y almacenar el objeto Image devuelto en im❸.

Paso 4: Añadir el logotipo y guardar los cambios

Tanto si la imagen se ha redimensionado como si no, el logotipo debe pegarse en la inferior derecha. El lugar exacto en el que debe pegarse el logotipo depende tanto del tamaño de la imagen como del tamaño del logotipo. La figura 17-12 muestra calcular la posición de pegado. La coordenada izquierda para pegar el logo será la anchura de la imagen menos la anchura del logo; la coordenada superior para pegar el logo será la altura de la imagen menos la altura del logo.

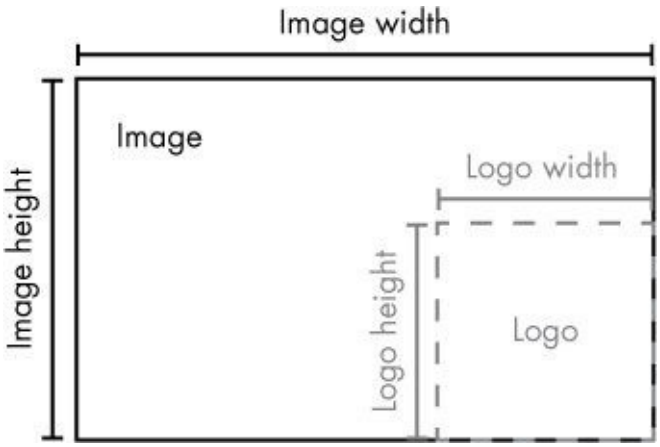


Figura 17-12. Las coordenadas izquierda y superior para colocar el logotipo en la esquina inferior derecha deben ser la anchura/altura de la imagen menos la anchura/altura del logotipo.

Después de que su código pegue el logotipo en la imagen, debería guardar la modificación Objeto imagen. Añade lo siguiente a tu programa:

```
#! python3
    # resizeAndAddLogo.py - Redimensiona todas las imágenes del directorio de trabajo #
    actual para que quepan en un cuadrado de 300x300, y añade catlogo.png a la
    esquina inferior derecha.

    importar os
    from PIL import Imagen

--snip--

    # Comprueba si la imagen necesita ser redimensionada.
    --snip--

    # Añadir el logotipo.
❶   print('Añadiendo logo a %s...' % (nombre de fichero))
❷   im.paste(logoIm, (width - logoWidth, height - logoHeight), logoIm) #

    Guardar CAMBIOS.
```

```
➲ im.save(os.path.join('withLogo', filename))
```

El nuevo código imprime un mensaje indicando al usuario que se está añadiendo el logo❶, pega `logoIm` sobre `im` en las coordenadas calculadas❷, y guarda los cambios en un nombre de archivo en el directorio *withLogo*❸. Cuando se ejecuta este programa con el archivo *zophie.png* como la única imagen en el directorio de trabajo, la salida se verá así:

```
Redimensionar zophie.png...
Añadir logotipo a zophie.png...
```

La imagen *zophie.png* será cambiada a una imagen de 225×300 píxeles que se parece a la Figura 17-13. Recuerde que el método `paste()` no pegará los píxeles de transparencia si no pasa el `logoIm` como tercer argumento. Este programa puede cambiar automáticamente el tamaño y "logo-ificar" cientos de imágenes en sólo un par de minutos.

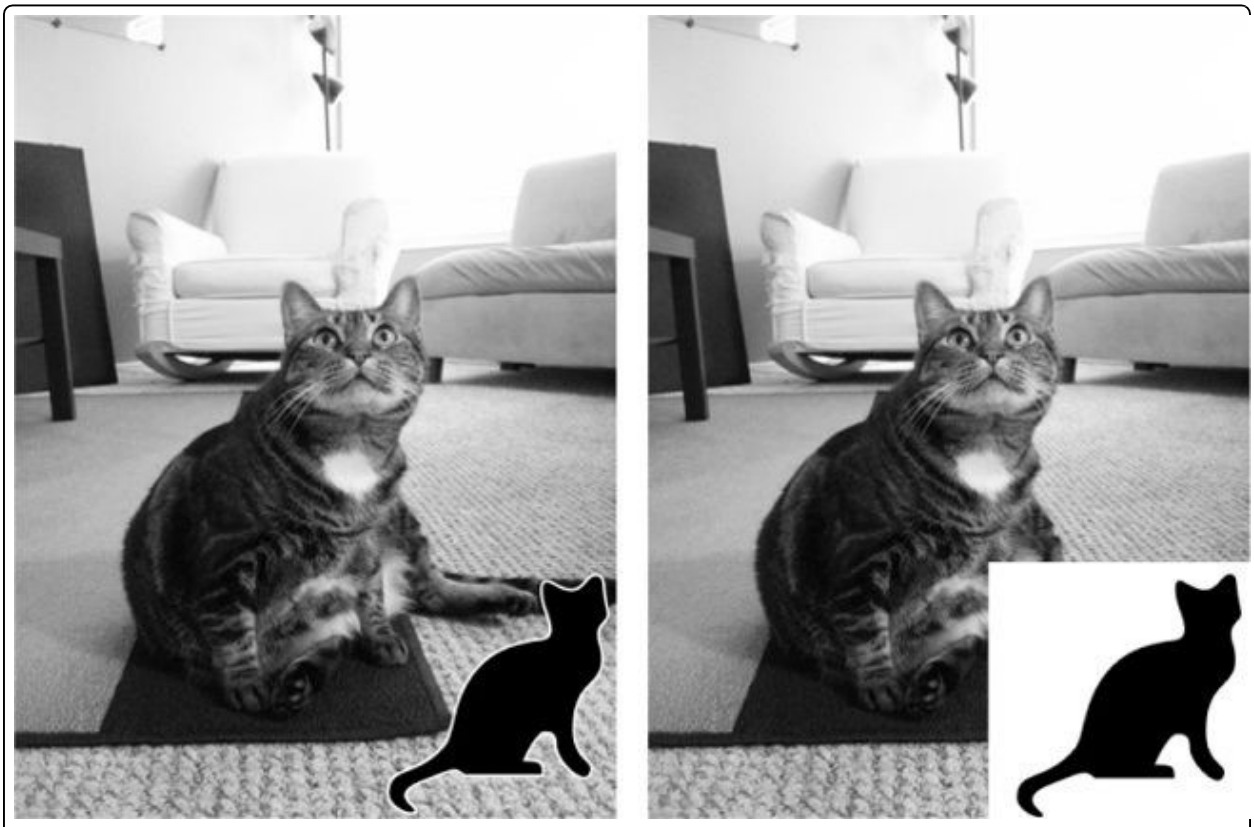


Figura 17-13. La imagen zophie.png redimensionada y el logotipo añadido (izquierda). Si olvida el tercer argumento, los píxeles transparentes del logotipo se copiarán como píxeles blancos sólidos (derecha).

Ideas para programas similares

Poder componer imágenes o modificar su tamaño por lotes puede ser útil en muchas aplicaciones. Podrías escribir programas similares para hacer lo siguiente:

- Añade texto o la URL de un sitio web a las imágenes.
- Añade marcas de tiempo a las imágenes.
- Copia o mueve imágenes a diferentes carpetas en función de su tamaño.
- Añade una marca de agua casi transparente a una imagen para evitar que otros la copien.

Dibujar sobre imágenes

Si necesita dibujar líneas, rectángulos, círculos u otras formas simples en una imagen, utilice el módulo ImageDraw de Pillow. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> from PIL import Image, ImageDraw
>>> im= Image.new('RGBA', (200, 200), 'white')
>>> dibujar= ImageDraw.Draw(im)
```

En primer lugar, importamos Image e ImageDraw. Luego creamos una nueva imagen, en este caso, una imagen blanca de 200×200, y almacenamos el objeto Image en im. Pasamos el objeto Image a la función ImageDraw.Draw() para recibir un objeto ImageDraw. Este objeto tiene varios métodos para dibujar formas y texto sobre un objeto Image. Guarde el objeto ImageDraw en una variable como draw para que pueda usarlo fácilmente en el siguiente ejemplo.

Dibujar formas

Los siguientes métodos de ImageDraw dibujan varios tipos de formas en la imagen. Los parámetros de `relleno` y `contorno` de estos métodos son opcionales y serán blancos por defecto si no se especifican.

Puntos

El método `point(xy, fill)` dibuja píxeles individuales. El argumento *xy* representa una lista de los puntos que desea dibujar. La lista puede ser una lista de tuplas de coordenadas xy, como `[(x, y), (x, y), ...]`, o una lista de coordenadas sin tuplas, como `[x1, y1, x2, y2, ...]`. El argumento de relleno es el color de los puntos y puede ser una tupla RGBA o una cadena con un nombre de color, como `"rojo"`. El argumento de *relleno* es opcional.

Líneas

El método `line(xy, fill, width)` dibuja una línea o una serie de líneas. *xy* es una lista de tuplas, como `[(x, y), (x, y), ...]`, o una lista de enteros, como `[x1, y1, x2, y2, ...]`. Cada punto es uno de los puntos de conexión de las líneas estás dibujando. El argumento opcional *fill* es el color de las líneas, como un tupla RGBA o nombre del color. El argumento opcional *anchura* es la anchura de las líneas y por defecto es 1 si no se especifica.

Rectángulos

El método `rectangle(xy, fill, outline)` dibuja un rectángulo. El argumento *xy* es una tupla de caja de la forma (*izquierda, arriba, derecha, abajo*). Los valores *left* y *top* especifican las coordenadas x e y de la esquina superior izquierda del rectángulo, mientras que *right* y bottom especifican la esquina inferior derecha. El argumento opcional *relleno* es el color que rellenará el interior del rectángulo. El argumento opcional *outline* es el color del contorno del rectángulo.

Elipses

El método `ellipse(xy, fill, outline)` dibuja una elipse. Si la anchura y la altura de la elipse son idénticas, este método dibujará un círculo. El argumento *xy* es una tupla de caja (*izquierda*, *arriba*, *derecha*, *abajo*) que representa una caja que contiene exactamente la elipse. El argumento opcional *relleno* es el color del interior de la elipse, y el argumento opcional *contorno* es el color del contorno *de* la elipse.

el contorno de la elipse.

Polígonos

El método polygon(*xy, fill, outline*) dibuja un polígono arbitrario. El argumento *xy* es una lista de tuplas, como `[(x, y), (x, y), ...]`, o enteros, como `[x1, y1, x2, y2, ...]`, que representan los puntos de conexión de los lados del polígono. El último par de coordenadas se conectará automáticamente al primer par. El argumento opcional *fill* es el color del interior del polígono.
y el argumento opcional *outline* es el color del contorno del polígono.

Ejemplo de dibujo

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> from PIL import Image, ImageDraw
    >>> im= Image.new('RGBA', (200, 200), 'white')
    >>> dibujar= ImageDraw.Draw(im)

>>> ❶draw.line([(0, 0), (199, 0), (199, 199), (0, 199), (0, 0)], fill='negro')

>>> ❷draw.rectangle((20, 30, 60, 60), fill='blue')

➲   >>> draw.ELLIPSE((120, 30, 160, 60), fill='rojo')

>>> ❹draw.polygon(((57, 87), (79, 62), (94, 85), (120, 90), (103, 113)),
    fill='marrón')

>>> ❺for i in range(100, 200, 10):
            draw.line([(i, 0), (200, i - 100)], fill='verde')

    >>> im.save('dibujo.png')
```

Después de crear un objeto Image para una imagen blanca de 200×200, pasarlo a ImageDraw.Draw() para obtener un objeto ImageDraw, y almacenar el objeto ImageDraw en draw, puede llamar a los métodos de dibujo en draw. Aquí hacemos un fino contorno negro en los bordes de la imagen❶, un rectángulo azul con su esquina superior izquierda en (20, 30) e inferior derecha en (60, 60) ❷, una elipse roja definida por una caja desde (120, 30) hasta (160, 60) ❸, un polígono marrón con cinco puntos ❹, y un patrón de líneas verdes dibujadas con un bucle for ❺. El archivo *drawing.png* tendrá el aspecto de la Figura 17-14.

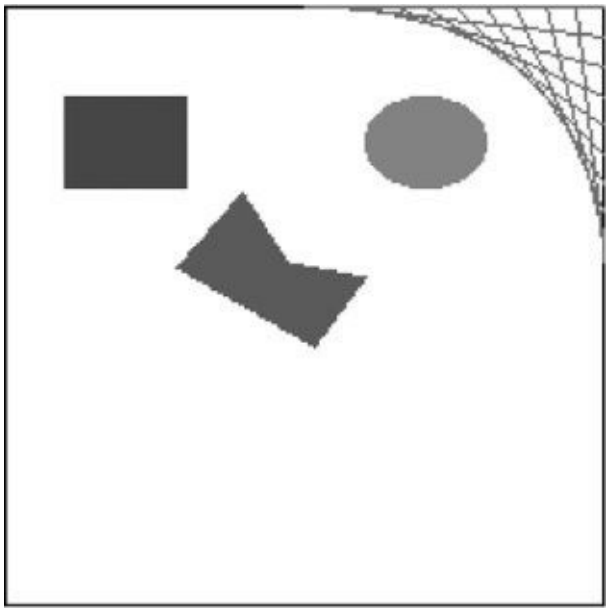


Figura 17-14. La imagen drawing.png resultante

Existen otros métodos de dibujo de formas para los objetos ImageDraw. La documentación completa está disponible en http://pillow.readthedocs.org/en/latest/reference/ImageDraw.html.

Texto de dibujo

El objeto ImageDraw también tiene un método text() para dibujar texto en una imagen. El método text() toma cuatro argumentos: *xy*, *texto*, *relleno* y *fuente*.

- El argumento *xy* es una tupla de dos enteros que especifica la esquina superior izquierda del cuadro de texto.
- El argumento *texto* es la cadena de texto que desea escribir.
- El argumento opcional *fill* es el color del texto.
- El argumento opcional *fuente* es un objeto ImageFont, utilizado para establecer el tipo de letra y el tamaño del texto. Esto se describe con más detalle en la siguiente sección.

Dado que a menudo es difícil saber de antemano qué tamaño tendrá un bloque de texto en una fuente determinada, el módulo ImageDraw también ofrece un método textsize(). Su primer argumento es la cadena de texto que desea medir, y su segundo argumento es un objeto ImageFont opcional. El método textsize() devolverá una tupla de dos enteros con la anchura y la altura que tendría el texto de la fuente si se escribiera en la imagen. Puede utilizar esta anchura y altura para calcular exactamente dónde desea colocar el texto en la imagen.

Los tres primeros argumentos de text() son sencillos. Antes de utilizar text() para dibujar texto sobre una imagen, veamos el cuarto opcional, el objeto ImageFont.

Tanto text() como textsize() toman un objeto ImageFont opcional como argumento final. Para crear uno de estos objetos, ejecute primero lo siguiente:

```
>>> from PIL import ImageFont
```

Ahora que ha importado el módulo ImageFont de Pillow, puede llamar a la función ImageFont.truetype(), que recibe dos argumentos. El primer argumento es una cadena para el archivo *TrueType* de la fuente, es decir, el archivo real de la fuente que se encuentra en su disco duro. Un archivo TrueType tiene la extensión *.ttf* y normalmente se encuentra en las siguientes carpetas:

- En Windows: *C:\Windows\Fonts*
- En OS X: *LibraryFonts* y */SystemLibraryFonts*
- En Linux: *usrshare/fonts/truetype*

En realidad, no es necesario introducir estas rutas como parte de la cadena del archivo TrueType

porque Python sabe buscar automáticamente las fuentes en estos directorios. Pero Python mostrará un error si no puede encontrar la fuente especificada.

El segundo argumento de ImageFont.truetype() es un número entero para el tamaño de la fuente en *puntos* (en lugar de, digamos, píxeles). Tenga en cuenta que Pillow crea imágenes PNG de 72 píxeles por pulgada por defecto, y un punto es 1/72 de pulgada.

Introduzca lo siguiente en el intérprete de comandos interactivo, sustituyendo FONT_FOLDER por el nombre real de la carpeta que utiliza su sistema operativo:

```
>>> from PIL import Image, ImageDraw, ImageFont
    >>> importar os
>>> ❶im= Image.new('RGBA', (200, 200), 'white')
>>> ❷dibujar= ImageDraw.Draw(im)
➲  >>> draw.text((20, 150), 'Hola', fill='morado')
    >>> fontsFolder= 'FONT_FOLDER' # p.ej. 'LIBRARY/FONTS'
>>> ❹arialFont= ImageFont.truetype(os.path.join(FONTSFOLDER, 'arial.ttf'), 32)
>>> ❺draw.text((100, 150), 'Howdy', fill='gray', font=arialFont)
    >>> im.save('texto.png')
```

Después de importar Image, ImageDraw, ImageFont y os, creamos un objeto Image para una nueva imagen blanca de 200×200 ❶ y creamos un objeto ImageDraw a partir del objeto Image ❷. Usamos text() para dibujar *Hola* en (20, 150) en morado ❸. No pasamos el cuarto argumento opcional en esta llamada a text(), por lo que el tipo de letra y el tamaño de este texto no están personalizados.

Para establecer un tipo de letra y tamaño, primero almacenamos el nombre de la carpeta (como *LibraryFonts*) en fontsFolder. Luego llamamos a ImageFont.truetype(), pasándole el archivo *.ttf* de la fuente que queremos, seguido de un tamaño de fuente entero ❹. Guarda el objeto Font que obtengas de ImageFont.truetype() en una variable como arialFont, y luego pasa la variable a text() en el argumento final de la palabra clave. La llamada a text() en ❺
dibuja *Howdy* en (100, 150) en gris en Arial de 32 puntos.

El archivo *text.png* resultante tendrá el aspecto de la Figura 17-15.



Figura 17-15. La imagen text.png resultante

Resumen

Las imágenes están formadas por una colección de píxeles, cada uno de los cuales tiene un valor RGBA para su color y es direccionable mediante coordenadas x e y. Dos formatos de imagen comunes son JPEG y PNG. El módulo de almohada puede manejar estos dos formatos de imagen y otros.

Cuando una imagen se carga en un objeto Image, sus dimensiones de anchura y altura se almacenan como una tupla de dos enteros en el atributo size. Los objetos del tipo de datos Imagen también tienen métodos para manipulaciones comunes de imágenes: recortar(), copiar(), pegar(), redimensionar(), rotar() y transponer(). Para guardar el objeto Image en un archivo de imagen, llame al método save().

Si desea que su programa dibuje formas sobre una imagen, utilice los métodos de ImageDraw para dibujar puntos, líneas, rectángulos, elipses y polígonos. El módulo también proporciona métodos para dibujar texto en un tipo y tamaño de fuente de su elección.

Aunque aplicaciones avanzadas (y caras) como Photoshop proporcionan funciones automáticas de procesamiento por lotes, puedes usar scripts de Python para hacer muchas de las mismas modificaciones de forma gratuita. En los capítulos anteriores, escribiste programas Python para tratar con archivos de texto plano, hojas de cálculo, PDFs y otros formatos.

Con el módulo de almohadas, habrás ampliado tus posibilidades de programación al tratamiento de imágenes.

Preguntas prácticas

P: 1. ¿Qué es un valor RGBA?

P: 2. ¿Cómo puedes obtener el valor RGBA de 'CornflowerBlue' del módulo Pillow?

P: 3. ¿Qué es una tupla de caja?

P: 4. ¿Qué función devuelve un objeto Imagen para, digamos, un archivo de imagen llamado *zophie.png*?

P: 5. ¿Cómo puede averiguar la anchura y la altura de la imagen de un objeto Imagen?

P: 6. ¿Qué método llamarías para obtener el objeto Imagen de una imagen de 100×100, excluyendo el cuarto inferior izquierdo de la misma?

P: 7. Después de realizar cambios en un objeto Imagen, ¿cómo podría guardarlo como un archivo de imagen?

P: 8. ¿Qué módulo contiene el código de dibujo de formas de Pillow?

P: 9. Los objetos imagen no tienen métodos de dibujo. ¿Qué tipo de objeto sí los tiene? ¿Cómo se obtiene este tipo de objeto?

Proyectos prácticos

Para practicar, escribe programas que hagan lo siguiente.

Ampliar y arreglar los programas de proyectos de los capítulos

El programa *resizeAndAddLogo.py* de este capítulo trabaja con archivos PNG y JPEG, pero Pillow soporta muchos más formatos que sólo estos dos. Amplíe *resizeAndAddLogo.*py para procesar también imágenes GIF y BMP.

Otro pequeño problema es que el programa modifica los archivos PNG y JPEG sólo si sus extensiones de archivo están en minúsculas. Por ejemplo, procesará *zophie.png* pero no *zophie.PNG*. Cambie el código para que la comprobación de la extensión del archivo no distinga entre mayúsculas y minúsculas.

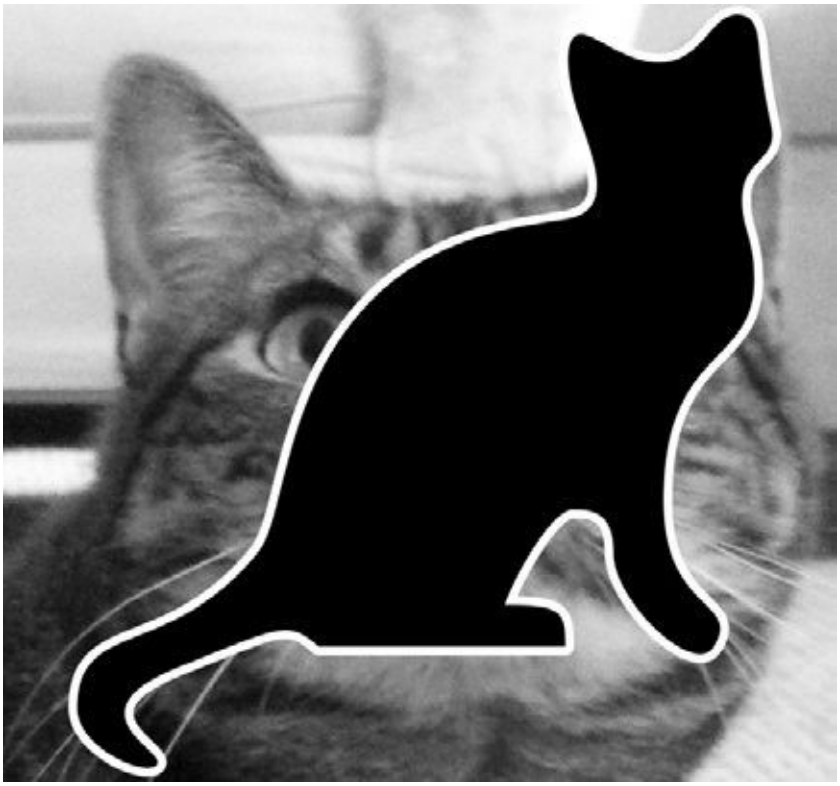


Figura 17-16. Cuando la imagen no es mucho más grande que el logotipo, los resultados se ven feos.

Por último, el logotipo añadido en la esquina inferior derecha está pensado para ser sólo una pequeña marca, pero si la imagen tiene aproximadamente el mismo tamaño que el propio logotipo, el resultado como el de la Figura 17-16. Modifique *resizeAndAddLogo.py* para que la imagen tenga al menos el doble de anchura y altura que la imagen del logotipo antes de pegar el logotipo.
En caso contrario, no debería añadir el logotipo.

Identificación de carpetas de fotos en el disco duro

Tengo la mala costumbre de transferir archivos de mi cámara digital a carpetas temporales en algún lugar del disco duro y luego olvidarme de estas carpetas. Estaría bien escribir un programa que pudiera escanear todo el disco duro y encontrar estas "carpetas de fotos" sobrantes.

Escribe un programa que recorra todas las carpetas de tu disco duro y encuentre posibles carpetas de fotos. Por supuesto, primero tendrás que definir qué consideras que una "carpeta de fotos"; digamos que es cualquier carpeta en la que más de la mitad de archivos sean fotos. ¿Y cómo se define qué archivos son fotos?

En primer lugar, un archivo de foto debe tener la extensión *.png* o *.jpg*. Además, las fotos son imágenes grandes; la anchura y la altura de un archivo de foto deben ser superiores a 500 píxeles. Es una apuesta segura, ya que la mayoría de las fotos de cámaras digitales varios miles de píxeles de ancho y alto.

A modo de pista, he aquí un esqueleto aproximado de lo que podría este programa:

```
#! python3 #
Importa módulos y escribe comentarios para describir este programa.

for foldername, subfolders, filenames in os.walk('C:\\'):
    numPhotoFiles= 0
    numNonPhotoFiles= 0
    para nombre de archivo en nombres de archivo:
        # Comprueba si la extensión del archivo no
        es .png o .jpg. if TODO:
            numNonPhotoFiles+= 1
            continue      # saltar al siguiente

        nombre de archivo # Abrir archivo de

        imagen usando Pillow.

        # Comprueba si la anchura y la altura son
        superiores a 500. if TODO:
            # Image is large enough to be considered a photo.
            numPhotoFiles+= 1
        si no:
            # La imagen es demasiado pequeña
            para ser una foto. numNonPhotoFiles+= 1

    # Si más de la mitad de los archivos eran
    fotos, # imprima la ruta absoluta de la
    carpeta. if TODO:
        print(TODO)
```

Cuando el programa se ejecuta, debe imprimir en la pantalla la ruta absoluta de las carpetas de fotos.

Tarjetas de asiento personalizadas

El capítulo 13 incluía un proyecto de práctica para crear invitaciones personalizadas a partir de una lista de invitados en un archivo de texto plano. Como proyecto adicional, utilice el módulo `de almohadas` para crear imágenes para tarjetas de asiento personalizadas para sus invitados. Para cada uno de los invitados listados en el archivo *guests.txt* de los recursos en http://nostarch.com/automatestuff/, genere un archivo de imagen con el nombre del invitado y alguna decoración floral. En los recursos de *http://nostarch.com/automatestuff/* encontrará una imagen de flores de dominio público.

Para asegurarse de que cada tarjeta de asiento tenga el mismo tamaño, añada un rectángulo negro en los bordes de la imagen de la invitación para que, cuando se la imagen, haya una pauta para cortar. Los archivos PNG que produce Pillow están configurados a 72 píxeles por pulgada, por lo que una tarjeta de 4×5 pulgadas requeriría una imagen de 288×360 píxeles.

Capítulo 18. Controlar el teclado y el ratón con la automatización GUI

Conocer varios módulos de Python para editar hojas de cálculo, descargar archivos y lanzar programas es útil, pero a veces simplemente no hay módulos para las aplicaciones con las que necesitas trabajar. Las herramientas por excelencia para automatizar tareas en tu ordenador son los programas que escribes y que controlan directamente el teclado y el ratón. Estos programas pueden controlar otras aplicaciones pulsaciones virtuales de teclado y ratón, como si estuvieras sentado frente al ordenador e interactuando con las aplicaciones. Esta técnica se conoce como *automatización de la interfaz gráfica de usuario (GUI)*. Con la automatización de la GUI, tus programas pueden hacer cualquier cosa que haga un usuario humano sentado frente al ordenador, excepto derramar café sobre el teclado.

Piense en la automatización de la interfaz gráfica como si programara un brazo robótico. Puedes programar el brazo robótico para que escriba en tu teclado y mueva el ratón por ti. Esta técnica es especialmente útil para tareas que implican hacer muchos clics sin sentido o rellenar formularios.

El módulo pyautogui tiene funciones para simular los movimientos del ratón, los clics en los botones y el desplazamiento de la rueda del ratón. Este capítulo cubre sólo un subconjunto de las características de PyAutoGUI; puedes encontrar la documentación completa en http://pyautogui.readthedocs.org/.

Instalación del módulo pyautogui

El módulo pyautogui puede enviar pulsaciones virtuales de teclas y clics de ratón a Windows, OS X y Linux. Dependiendo del sistema operativo que estés usando, puede que tengas que instalar algunos otros módulos (llamados *dependencias*) antes de poder instalar PyAutoGUI.

- En Windows, no hay que instalar ningún otro módulo.
- En OS , ejecute **sudo pip3 install pyobjc-framework-Quartz**, **sudo pip3 install pyobjc-core**, y luego **sudo pip3 install pyobjc**.
- En Linux, ejecuta **sudo pip3 install python3-xlib**, **sudo apt-get install scrot**, **sudo apt-get install python3-tk** y **sudo apt-get install python3-dev**. (Scrot es un programa de captura de pantalla que utiliza PyAutoGUI).

Una vez instaladas estas dependencias, ejecute **pip install pyautogui** (o **pip3** en OS X y Linux) para instalar PyAutoGUI.

El Apéndice A contiene información completa sobre la instalación de módulos de terceros. Para comprobar si PyAutoGUI se ha instalado correctamente, ejecute **import pyautogui** desde el intérprete de comandos interactivo y compruebe si aparece algún mensaje de error.

Mantener el rumbo

Antes de lanzarte a la automatización de una GUI, debes saber cómo escapar de los problemas que puedan surgir. Python puede mover el ratón y teclear a una velocidad increíble. De hecho, puede ser demasiado rápido para que otros programas puedan seguirle el ritmo. Además, si algo va mal pero tu programa sigue moviendo el ratón, será difícil saber qué está haciendo exactamente el programa o cómo recuperarse del problema. Al igual que las escobas encantadas *de El aprendiz de brujo* de Disney, que no paraban de llenar -y luego de llenar en exceso- la bañera de Mickey con agua, su programa podría descontrolarse aunque esté siguiendo sus instrucciones a la perfección. Detener el programa puede ser difícil si el ratón se mueve por sí solo, impidiéndole hacer clic en la ventana IDLE para cerrarla. Afortunadamente, hay varias formas de prevenir o recuperarse de los problemas de automatización de la GUI.

Apagar todo cerrando la sesión

Tal vez la forma más sencilla de detener un programa de automatización GUI fuera de control es cerrar la sesión, lo que cerrará todos los programas en ejecución. En Windows y Linux, la tecla de cierre de sesión es CTRL-ALT-DEL. En OS , es⌘-SHIFT-OPTION-Q. Al cerrar la sesión, perderás todo el trabajo que no hayas guardado, pero al menos no tendrás que esperar a que se reinicie el ordenador.

Pausas y FailSafes

Puedes decirle a tu script que espere después de cada llamada a una función, dándote una pequeña ventana para tomar el control del ratón y el teclado si algo va mal. Para hacer esto, establece la variable pyautogui.PAUSE al número de segundos que quieres que espere. Por ejemplo, después de establecer pyautogui.PAUSE= 1.5, cada llamada a una función PyAutoGUI esperará un segundo y medio después de realizar su acción. Las instrucciones que no sean PyAutoGUI no tendrán esta pausa.

PyAutoGUI también tiene una función a prueba de fallos. Si mueves el cursor del ratón a la esquina superior izquierda de la pantalla, PyAutoGUI lanzará la excepción pyautogui.FailSafeException. Tu programa puede manejar esta excepción con sentencias try y except o dejar que la excepción bloquee tu programa. En cualquier caso, la función a prueba de fallos detendrá el programa si mueves rápidamente el ratón hacia arriba y hacia la izquierda todo lo que puedas. Puede desactivar esta función configurando pyautogui.FAILSAFE= False. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar pyautogui
>>> pyautogui.PAUSE= 1
>>> pyautogui.FAILSAFE= Verdadero
```

Aquí importamos pyautogui y establecemos pyautogui.PAUSE a 1 para una pausa de un segundo después de cada llamada a una función. Establecemos pyautogui.FAILSAFE a True para activar la función a prueba de fallos.

Control del movimiento del ratón

En esta sección, aprenderás cómo mover el ratón y seguir su posición en la pantalla usando PyAutoGUI, pero primero necesitas entender cómo PyAutoGUI trabaja con coordenadas.

Las funciones de ratón de PyAutoGUI utilizan coordenadas x e y. La Figura 18-1 muestra el sistema de coordenadas para la pantalla del ordenador; es similar al sistema de coordenadas usado para las imágenes, discutido en el Capítulo 17. El origen, donde x e y son cero, está en la esquina superior izquierda de la pantalla. *El origen*, donde *x* e *y* son cero, está en la esquina superior izquierda de la pantalla. Las coordenadas x aumentan hacia la derecha, y las coordenadas y aumentan hacia abajo. Todas las coordenadas son números enteros positivos; no hay coordenadas negativas.

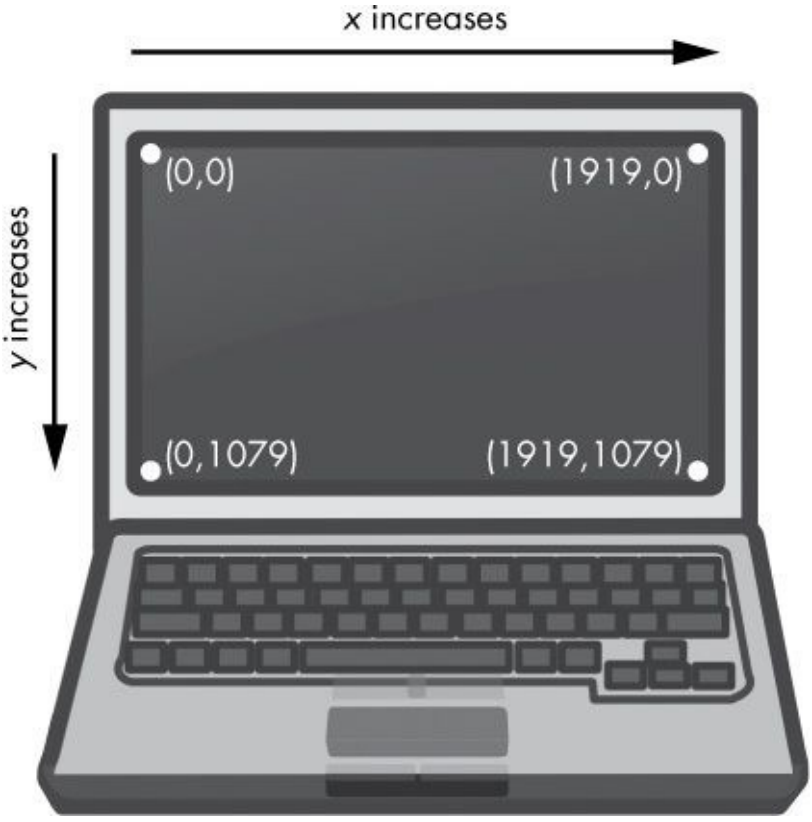


Figura 18-1. Coordenadas de una pantalla de ordenador con resolución 1920×1080

Tu *resolución* es el número de píxeles de ancho y alto de tu pantalla. Si la resolución de tu pantalla es 1920×1080, la coordenada de la esquina superior izquierda será (0, 0), y la de la esquina inferior derecha será (1919, 1079).

La función pyautogui.size() devuelve una tupla de dos enteros de la anchura y altura de la pantalla en píxeles. Introduce lo siguiente en el intérprete de comandos interactivo:

```
>>> importar pyautogui
>>> pyautogui.tamaño()
(1920, 1080)
>>> anchura, altura= pyautogui.size()
```

pyautogui.size() devuelve (1920, 1080) en un ordenador con una resolución de 1920×1080; dependiendo de la resolución de tu pantalla, tu valor de retorno puede ser diferente. Puedes almacenar la anchura y altura de pyautogui.size() en variables como width y height para una mejor legibilidad en tus programas.

Mover el ratón

Ahora que entiendes las coordenadas de la pantalla, a mover el ratón. La función pyautogui.moveTo() moverá instantáneamente el cursor del ratón a una posición especificada en la pantalla. Los valores enteros de las coordenadas x e y constituyen el primer y segundo argumento de la función, respectivamente. Un argumento opcional de duración (integer o float) especifica el número de segundos que se tardará en mover el ratón hasta el destino. Si se omite, el valor por defecto es 0 para un movimiento instantáneo. (Todos los argumentos de duración en las funciones PyAutoGUI son opcionales.) Introduce lo siguiente en el intérprete de comandos interactivo:

```
>>> importar pyautogui
>>> para i en rango(10):
        pyautogui.moveTo(100,   100,   duración=0,25)
        pyautogui.moveTo(200,   100,   duración=0,25)
        pyautogui.moveTo(200,   200,   duración=0,25)
        pyautogui.moveTo(100,   200,   duración=0,25)
```

Este ejemplo mueve el cursor del ratón en el sentido de las agujas del reloj en un patrón cuadrado entre las cuatro coordenadas proporcionadas un total de diez veces. Cada movimiento dura un cuarto de segundo, como se especifica en el argumento duration=0.25. Si no hubieras pasado un tercer argumento a ninguna de las llamadas de pyautogui.moveTo(), el cursor del ratón se habría teletransportado instantáneamente de un punto a otro.

La función pyautogui.moveRel() mueve el cursor del ratón *en relación* a su posición actual. El siguiente ejemplo mueve el ratón en el mismo patrón cuadrado, excepto que comienza el cuadrado desde donde el ratón se encuentre en la pantalla cuando el código comienza a ejecutarse:

```
>>> importar pyautogui
>>> para i en rango(10):
        pyautogui.moveRel(100, 0, duration=0.25)
        pyautogui.moveRel(0, 100, duration=0.25)
        pyautogui.moveRel(-100, 0, duration=0.25)
        pyautogui.moveRel(0, -100, duration=0.25)
```

pyautogui.moveRel() también toma tres argumentos: cuántos píxeles mover horizontalmente a la derecha, cuántos píxeles mover verticalmente hacia abajo, y (opcionalmente) cuánto tiempo debería tardar en completar el . Un número entero negativo para el primer o segundo argumento hará que el ratón se mueva hacia la izquierda o hacia arriba, respectivamente.

Obtener la posición del ratón

Puedes determinar la posición actual del ratón llamando a la función pyautogui.position(), que devolverá una tupla de las posiciones *x* e *y* del cursor del ratón en el momento de la llamada a la función. Introduce lo siguiente en el intérprete de comandos interactivo, moviendo el ratón después de cada llamada:

```
>>> pyautogui.position()
(311, 622)
>>> pyautogui.position()
(377, 481)
>>> pyautogui.position()
(1536, 637)
```

Por supuesto, sus valores de retorno variarán dependiendo de dónde se encuentre el cursor del ratón.

Proyecto: "¿Dónde está el ratón ahora mismo?"

Ser capaz de determinar la posición del ratón es una parte importante de la configuración de tus scripts de automatización de GUI. Pero es casi imposible averiguar las coordenadas exactas de un píxel con sólo mirar la pantalla. Sería práctico tener un programa que mostrara constantemente las coordenadas x e y del cursor del ratón a medida que lo mueves.

A grandes , esto es lo que debe hacer tu programa:

- Muestra las coordenadas x e y actuales del cursor del ratón.
- Actualiza estas coordenadas a medida que el ratón se mueve por la

pantalla. Esto significa que su código tendrá que hacer lo siguiente:

- Llama a la función position() para obtener las coordenadas actuales.
- Borra las coordenadas previamente impresas imprimiendo \b caracteres de retroceso en la pantalla.
- Maneja la excepción KeyboardInterrupt para que el usuario pueda presionar CTRL-C para salir.

Abre una nueva ventana del editor de archivos y guárdalo como *mouseNow.py*.

Paso 1: Importar el módulo

Comienza tu programa con lo siguiente:

```
#! python3
# mouseNow.py - Muestra la posición actual del cursor del ratón. import
pyautogui
print('Pulsa Ctrl-C para salir')
#TODO: Obtener e imprimir las coordenadas del ratón.
```

El principio del programa importa el módulo pyautogui e imprime un recordatorio al usuario de que tiene que pulsar CTRL-C para salir.

Paso 2: Configurar el código de salida y el bucle infinito

Puedes usar un bucle while infinito para imprimir constantemente las coordenadas actuales del ratón desde mouse.position(). En cuanto al código que cierra el programa, necesitarás atrapar la excepción KeyboardInterrupt, que se lanza cada vez que el usuario presiona CTRL-C. Si no manejas esta excepción, se mostrará un feo mensaje de error al usuario. Añade lo siguiente a tu programa:

```
#! python3
    # mouseNow.py - Muestra la posición actual del cursor del ratón. import
    pyautogui
    print('Pulsa Ctrl-C para salir')
    Inténtalo:
        mientras sea verdad:
            # TODO: Obtener e imprimir las COORDENADAS del raton.
❶ excepto KeyboardInterrupt:
❷       print('Hecho.')
```

Para manejar la excepción, encierra el bucle infinito while en una sentencia try. Cuando el usuario pulse CTRL-C, la ejecución del programa pasará a la cláusula except ❶ y Hecho. se imprimirá en una nueva línea❷.

Paso 3: Obtener e imprimir las coordenadas del ratón

El código dentro del bucle while debería obtener las coordenadas actuales del ratón, formatearlas para que se vean bien, e imprimirlas. Añade el siguiente código al interior del bucle while:

```
#! python3
# mouseNow.py - Muestra la posición actual del cursor del ratón. import
pyautogui
print('Pulsa Ctrl-C para salir')
--snip--
        # Obtener e imprimir las COORDENADAS del
        ratón. x, y= pyautogui.position()
        positionStr= 'X: '+ str(x).rjust(4)+ ' Y: '+ str(y).RJUST(4)
--snip--
```

Utilizando el truco de la asignación múltiple, las variables x e y reciben los valores de los dos enteros devueltos en la tupla de pyautogui.position(). Pasando x e y a la función str(), puedes obtener formas de cadena de las coordenadas enteras. El método de cadena rjust() las justificará a la derecha para que ocupen el mismo espacio, independientemente de que la coordenada tenga uno, dos, tres o cuatro caracteres. dígitos. Al concatenar las coordenadas de la cadena justificadas a la derecha con las etiquetas 'X: ' e ' Y: ' obtenemos una cadena formateada ordenadamente, que se almacenará en positionStr.

Al final de tu programa, añade el siguiente código:

```
#! python3
    # mouseNow.py - Muestra la posición actual del cursor del ratón.
    --snip--
            print(posiciónStr, fin='')
❶           print('\b' * len(positionStr), end='', FLUSH=TRUE)
```

Esto imprime realmente positionStr en la pantalla. El argumento end='' de print() evita que el carácter de nueva línea por defecto se añada al final de la línea impresa. Es posible borrar el texto ya impreso en la pantalla
-pero sólo para la línea de más reciente. Una vez que se imprime un carácter de nueva línea, no puede borrar nada impreso antes.

Para borrar texto, imprima el carácter de escape de retroceso \b. Este carácter especial borra un carácter al final de la línea actual en la pantalla. La línea en ❶ utiliza la replicación de cadena para producir una cadena con tantos caracteres \b como la longitud de la cadena almacenada en positionStr, que tiene el efecto de borrar el

positionStr cadena que se imprimió por última vez.

Por una razón técnica que escapa al alcance de este libro, pase siempre flush=True a las llamadas print() que imprimen caracteres de retroceso. De lo contrario, la pantalla podría no actualizar el texto como se desea.

Como el bucle while se repite tan rápidamente, el usuario no se dará cuenta de que está borrando y volviendo a imprimir el número entero en la pantalla. Por ejemplo, si la coordenada x es 563 y el ratón se mueve un píxel a la derecha, parecerá que sólo el 3 de 563 se cambia por un 4.

Cuando ejecute el programa, sólo se imprimirán dos líneas. Deberían parecerse a algo como esto:

```
Pulsa Ctrl-C para salir.
X: 290 Y: 424
```

La primera línea muestra la instrucción de pulsar CTRL-C para salir. La segunda línea con las coordenadas del ratón cambiará a medida que muevas el ratón por la pantalla. Usando este programa, podrás averiguar las coordenadas del ratón para tus scripts de automatización GUI.

Control de la interacción con el ratón

Ahora que ya sabes mover el ratón y saber dónde está en la pantalla, estás listo para empezar a hacer clic, arrastrar y desplazarte.

Hacer clic con el ratón

Para enviar un clic virtual del ratón a tu ordenador, llama al método pyautogui.click(). Por defecto, este clic utiliza el botón izquierdo del ratón y tiene lugar donde se encuentre el cursor del ratón. Puedes pasar las coordenadas x e y del clic como primer y segundo argumento opcionales si quieres que tenga lugar en algún lugar distinto de la posición actual del ratón.

Si quieres especificar qué botón del ratón utilizar, incluye el argumento de la palabra clave button, con un valor de 'izquierdo', "central" o 'derecho'. Por ejemplo, pyautogui.click(100, 150, button='left') hará clic con el botón izquierdo del ratón en las coordenadas (100, 150), mientras que pyautogui.click(200, 250, button='right') hará clic con el botón derecho en (200, 250).

Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar pyautogui
>>> pyautogui.click(10, 5)
```

Debería ver que el puntero del ratón se desplaza hasta cerca de la esquina superior izquierda de la pantalla y hace clic una vez. Un "clic" completo se define como pulsar el botón del ratón hacia abajo y luego soltarlo de nuevo hacia arriba sin mover el cursor. También puedes clic llamando a pyautogui.mouseDown(), que sólo presiona el del ratón hacia abajo, y pyautogui.mouseUp(), que sólo suelta el botón. Estas funciones tienen los mismos argumentos que click(), y de hecho, la función click() es sólo una envoltura conveniente alrededor de estas dos llamadas a funciones.

Para mayor comodidad, la función pyautogui.doubleClick() realizará dos clics con el botón izquierdo del ratón, mientras que las funciones pyautogui.rightClick() y pyautogui.middleClick() realizarán un clic con los botones derecho y central del ratón, respectivamente.

Arrastrar el ratón

Arrastrar significa mover el ratón mientras mantienes pulsado uno de los botones del ratón. Por ejemplo, puedes mover archivos de una carpeta a otra arrastrando los iconos de las carpetas, o desplazar citas en una aplicación de calendario.

PyAutoGUI proporciona las funciones pyautogui.dragTo() y pyautogui.dragRel() para arrastrar el cursor del ratón a una nueva posición o a una posición relativa a la actual. Los argumentos de dragTo() y dragRel() son los mismos que moveTo() y moveRel(): la coordenada horizontal, la coordenada y/movimiento vertical, y una duración de opcional. (OS X no arrastra correctamente cuando el ratón se mueve demasiado rápido, por lo que se recomienda pasar un argumento de duración).

Para probar estas funciones, abra una aplicación de dibujo gráfico como Paint en Windows, Paintbrush en OS X o GNU Paint en Linux. (Si no tienes una aplicación de dibujo, puedes usar la que está en línea en http://sumopaint.com/.) Yo usaré PyAutoGUI para dibujar en estas aplicaciones.

Con el cursor del ratón sobre el lienzo de la aplicación de dibujo y la herramienta Lápiz o Pincel seleccionada, introduce lo siguiente en una nueva ventana del editor de archivos y guárdalo como *spiralDraw.py*:

```
importar pyautogui, tiempo
❶ time.sleep(5)
❷ pyautogui.click()      # click para poner el programa de dibujo en
   foco distancia= 200
   mientras la distancia> 0:
➲      pyautogui.dragRel(distancia, 0, duración=0.2)          # mover a la derecha
❹      distancia= distancia - 5
❺      pyautogui.dragRel(0, distancia, duración=0.2)          # mover hacia abajo
❻      pyautogui.dragRel(-distancia, 0, duración=0.2)         # mover a la
       izquierda distancia= distancia - 5
       pyautogui.dragRel(0, -distancia, duración=0.2)         # mover hacia arriba
```

Cuando ejecutes este programa, habrá un retraso de cinco segundos ❶ para que muevas el cursor del ratón sobre la ventana del programa de dibujo con la herramienta Lápiz o Pincel seleccionada. Entonces *spiralDraw.py* tomará el control del ratón y hará clic para poner el programa de dibujo en foco ❷. Una ventana está *enfocada* cuando tiene un cursor parpadeante activo, y las acciones que realices -como escribir o, en este caso, arrastrar el ratón- afectarán a esa ventana. Una vez que el programa de dibujo está en foco, *spiralDraw.py* dibuja un patrón espiral cuadrado como el de la Figura 18-2.

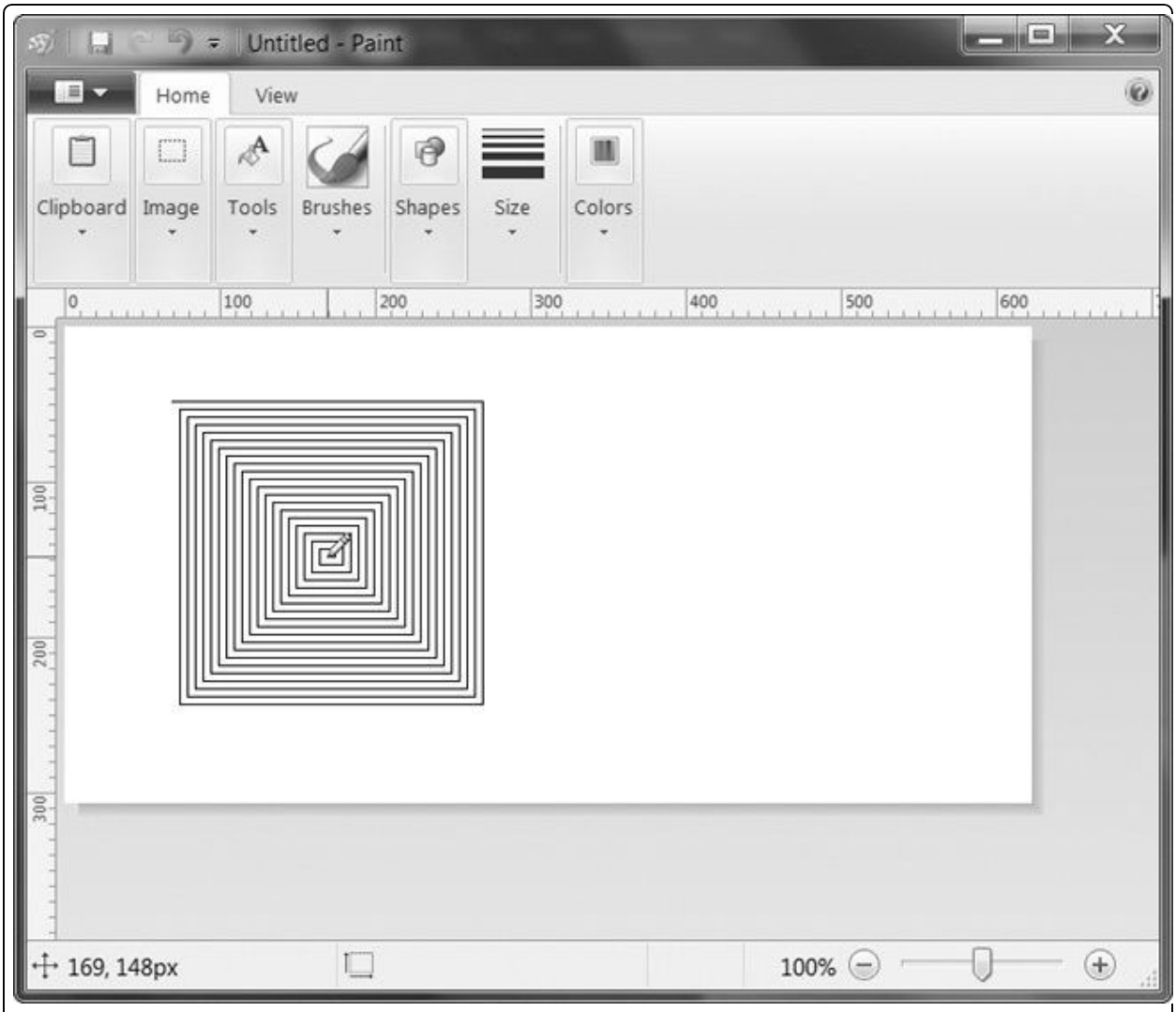


Figura 18-2. Resultados del ejemplo pyautogui.dragRel()

La variable `distancia` comienza en `200`, por lo que en la primera iteración del bucle `while`, la primera llamada `dragRel()` arrastra el cursor 200 píxeles a la derecha, tardando 0,2 segundos❸. la `distancia` se disminuye entonces a 195❹, y la segunda llamada `dragRel()` arrastra el cursor 195 píxeles hacia abajo❺. La tercera llamada `dragRel()` arrastra el cursor -195 horizontalmente (195 a la izquierda)❻, la `distancia` se disminuye a 190, y la última llamada `dragRel()` arrastra el cursor 190 píxeles hacia arriba. En cada iteración, la función
el ratón se arrastra hacia la derecha, hacia abajo, hacia la izquierda y hacia arriba, y la `distancia` es ligeramente menor que en la iteración anterior. Haciendo un bucle sobre este , puedes mover el cursor del ratón para dibujar una espiral cuadrada.

Podrías dibujar esta espiral a mano (o mejor dicho, con el ratón), pero tendrías que trabajar despacio para ser tan preciso. ¡PyAutoGUI puede hacerlo en unos segundos!

NOTA

Puede hacer que su código dibuje la imagen utilizando las funciones de dibujo del módulo de almohada *- vea el Capítulo 17 para más información. Pero usar la automatización GUI le permite hacer uso de las herramientas de dibujo avanzadas que los programas gráficos pueden proporcionar, como gradientes, diferentes pinceles, o el cubo de relleno.*

Desplazamiento del ratón

La última función del ratón de PyAutoGUI es scroll(), a la que pasas un argumento entero para saber cuántas unidades quieres desplazar el ratón hacia arriba o hacia abajo. El tamaño de una unidad varía para cada sistema operativo y aplicación, así que tendrás que experimentar para ver exactamente hasta dónde se desplaza en tu situación particular. El desplazamiento tiene lugar en la posición actual del cursor del ratón. Pasando un entero positivo se desplaza hacia arriba, y pasando un entero negativo se desplaza hacia abajo. Ejecute lo siguiente en el shell interactivo de IDLE mientras el cursor del ratón está sobre la ventana de IDLE:

```
>>> pyautogui.scroll(200)
```

Verás que IDLE se desplaza brevemente hacia arriba y luego hacia abajo. El desplazamiento hacia abajo ocurre porque IDLE se desplaza automáticamente hacia abajo después de ejecutar una instrucción. Introduce este código en su lugar:

```
>>> importar pyperclip
>>> NÚMEROS= ''
>>> para i en rango(200):
        NÚMEROS= NÚMEROS+ str(i)+ '\n'

>>> pyperclip.copy(NÚMEROS)
```

Esto importa pyperclip y establece una cadena vacía, números. A continuación, el código recorre 200 números y añade cada número a numbers, junto con una nueva línea. Después de pyperclip.copy(numbers), el portapapeles se cargará con 200 líneas de números. Abre una nueva ventana del editor de archivos y pega el texto en ella. Esto le dará una ventana de texto grande para intentar desplazarse en ella. Introduce el siguiente código en el intérprete de comandos interactivo:

```
>>> importar tiempo, pyautogui
>>> time.sleep(5); pyautogui.scroll(100)
```

En la segunda línea, introduces dos comandos separados por punto y coma, lo que indica a Python que ejecute los comandos como si estuvieran en líneas separadas. La única diferencia es que el shell interactivo no te pedirá que introduzcas datos entre las dos instrucciones. Esto es importante para este ejemplo porque queremos que la llamada a pyautogui.scroll() para que suceda automáticamente después de la espera. (Ten en cuenta que aunque poner dos comandos en una línea puede ser útil en el shell interactivo, aún así deberías tener cada instrucción en una línea separada en tus programas).

Después de pulsar ENTER para ejecutar el código, tendrá cinco segundos para hacer clic en el archivo

para ponerla en foco. Una vez finalizada la pausa, la llamada pyautogui.scroll() hará que la ventana del editor de archivos se desplace hacia arriba tras el retardo de cinco segundos.

Trabajar con la pantalla

Tus programas de automatización GUI no tienen que hacer clic y escribir a ciegas. PyAutoGUI tiene funciones de captura de pantalla que pueden crear un archivo de imagen basado en el contenido actual de la pantalla. Estas funciones también pueden devolver un objeto Pillow `Image` de la apariencia de la pantalla actual. Si has estado saltando en este libro, querrás leer el Capítulo 17 e instalar el módulo `pillow` antes de continuar con esta sección.

En ordenadores Linux, es necesario instalar el programa scrot para utilizar las funciones de captura de pantalla en PyAutoGUI. En una ventana de Terminal, ejecuta **`sudo apt-get install scrot`** para instalar este programa. estás en Windows u OS X, sáltate este paso y continúa con la sección.

Obtener una captura de pantalla

Para hacer capturas de pantalla en Python, llama a la función pyautogui.screenshot(). Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar pyautogui
>>> im= pyautogui.SCREENSHOT()
```

La variable im contendrá el objeto Imagen de la captura de pantalla. Ahora puede llamar a los métodos del objeto Image en la im, como cualquier otro objeto Image. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> im.getpixel((0, 0))
(176, 176, 175)
>>> im.getpixel((50, 200))
(130, 135, 144)
```

Pase a getpixel() una tupla de coordenadas, como (0, 0) o (50, 200), y le dirá el color del píxel en esas coordenadas de su imagen. El valor devuelto por getpixel() es una tupla RGB de tres enteros para la cantidad de rojo, verde y azul en el píxel. (No hay un cuarto valor para alfa, porque las imágenes de capturas de pantalla son totalmente opacas). Así es como sus programas pueden "ver" lo que hay actualmente en la pantalla.
pantalla.

Análisis de la captura de pantalla

Digamos que uno de los pasos en su programa de automatización GUI es hacer clic en un botón gris. Antes de llamar al método `click()`, podrías hacer una captura de pantalla y mirar el píxel donde el script está a punto de hacer clic. Si no es del mismo gris que el botón gris, entonces tu programa sabe que algo va mal. Tal vez la ventana se ha movido inesperadamente, o tal vez un cuadro de diálogo emergente ha bloqueado el botón. En este punto, en lugar de continuar -y posiblemente causar estragos al hacer clic en la cosa equivocada- su programa puede "ver" que no está haciendo clic en la cosa correcta y detenerse.

La función pixelMatchesColor() de PyAutoGUI devuelve True si el píxel en las coordenadas x e y de la pantalla coincide con el color dado. El primer y segundo argumento son enteros para las coordenadas x e y, y el tercer argumento es una tupla de tres enteros para el color RGB que el píxel de la pantalla debe coincidir. Introduzca lo siguiente en el intérprete de comandos interactivo:

```
>>> importar pyautogui
    >>> im= pyautogui.SCREENSHOT()
>>> ❶im.getpixel((50, 200))
    (130, 135, 144)
>>> ❷pyautogui.PIXELMATCHESCOLOR(50, 200, (130, 135, 144))
    Verdadero
➲   >>> pyautogui.PIXELMATCHESCOLOR(50, 200, (255, 135, 144))
    Falso
```

Después de tomar una captura de pantalla y usar getpixel() para obtener una tupla RGB para el color de un píxel en coordenadas específicas ❶, pasa las mismas coordenadas y tupla RGB a pixelMatchesColor() ❷, que debería devolver True. A continuación, cambie un valor en la tupla RGB y llame de nuevo a pixelMatchesColor() para las mismas coordenadas ❸. Esto debería devolver false. Este método puede ser útil para llamar siempre que sus programas de automatización GUI estén a punto de llamar a click(). Tenga en cuenta que el color en las coordenadas dadas debe coincidir *exactamente*. Si es incluso ligeramente diferente -por ejemplo, (255, 255, 254) en lugar de (255, 255, 255)- entonces pixelMatchesColor() devolverá False.

Proyecto: Ampliación del programa mouseNow

Puedes extender el proyecto *mouseNow.py* de este capítulo para que no sólo dé las coordenadas x e y de la posición actual del cursor del ratón, sino que también dé el color RGB del píxel bajo el cursor. Modifica el código dentro del bucle `while` de *mouseNow.py* para que tenga este aspecto:

```
#! python3
# mouseNow.py - Muestra la posición actual del cursor del ratón.
--snip--
        positionStr= 'X: '+ str(x).rjust(4)+ ' Y: '+ str(y).rjust(4)
        pixelColor= pyautogui.screenshot().getpixel((x, y)) positionStr+=
        ' RGB: ('+ str(pixelColor[0]).rjust(3) positionStr+= ', '+
        str(pixelColor[1]).rjust(3) positionStr+= ', '+
        str(pixelColor[2]).rjust(3)+ ')' print(positionStr, end='')
--snip--
```

Ahora, cuando ejecutes *mouseNow.py*, la salida incluirá el valor de color RGB del píxel bajo el cursor del ratón.

```
Pulsa Ctrl-C para salir.
X:   406 Y:      17 RGB: (161, 50, 50)
```

Esta información, junto con la función `pixelMatchesColor()`debería facilitar la adición de comprobaciones de color de píxel a sus scripts de automatización de GUI.

Reconocimiento de imágenes

¿Pero qué pasa si no sabes de antemano dónde debe hacer clic PyAutoGUI? Puedes usar el reconocimiento de imagen. Dale a PyAutoGUI una imagen de lo que quieres haga clic y deja que averigüe las coordenadas.

Por ejemplo, si previamente ha realizado una captura de pantalla para capturar la imagen de un botón Enviar en *submit.png*, la función locateOnScreen() devolverá las coordenadas donde se encuentra esa imagen. Para ver cómo funciona locateOnScreen(), intente hacer una captura de pantalla de un área pequeña de su pantalla; después guarde la imagen e introduzca lo siguiente en el intérprete de comandos interactivo, sustituyendo 'submit. png' por el nombre de archivo de su captura de pantalla:

```
>>> importar pyautogui
>>> pyautogui.locateOnScreen('submit.png')
(643, 745, 70, 29)
```

La tupla de cuatro enteros que devuelve locateOnScreen() tiene la coordenada x del borde izquierdo, la coordenada y del borde superior, la anchura y la altura del primer lugar de la pantalla donde se encontró la imagen. Si está probando esto en su ordenador con su propia captura de pantalla, su valor de retorno será diferente del que se muestra aquí.

Si la imagen no se encuentra en la pantalla, locateOnScreen() devolverá None. Tenga en cuenta que la imagen en la pantalla debe coincidir perfectamente con la imagen proporcionada para ser reconocida. Si la imagen tiene un píxel de diferencia, locateOnScreen() devolverá None.

Si la imagen puede encontrarse en varios lugares de la pantalla, locateAllOnScreen() devolverá un objeto Generator, que puede pasarse a list() para que devuelva una lista de tuplas de cuatro enteros. Habrá una tupla de cuatro enteros por cada lugar de la donde se encuentre la imagen. Continúe con el ejemplo de shell interactivo introduciendo lo siguiente (y sustituyendo 'submit.png' por su propio nombre de archivo de imagen):

```
>>>    list(pyautogui.locateAllOnScreen('submit.png'))
[(643, 745, 70, 29), (1007, 801, 70, 29)]
```

Cada una de las tuplas de cuatro enteros representa un área de la pantalla. Si su imagen sólo se encuentra en un área, entonces el uso de list() y locateAllOnScreen() sólo devuelve una lista que contiene una tupla.

Una vez que tenga la tupla de cuatro enteros para el área de la pantalla donde se encontró la imagen, puede hacer clic en el centro de esta área pasando la tupla a la función center() para devolver las coordenadas x e y del centro del área. Introduzca lo siguiente en el intérprete de comandos interactivo, sustituyendo los argumentos por su propio nombre de archivo, tupla de cuatro enteros y par de coordenadas:

```
>>> pyautogui.locateOnScreen('submit.png')
(643, 745, 70, 29)
>>> pyautogui.center((643, 745, 70, 29))
(678, 759)
>>>   pyautogui.click((678,    759))
```

Una vez que tenga las coordenadas centrales de center(), al pasar las coordenadas a click() debería hacer clic en el centro del área de la pantalla que coincida con la imagen que pasó a locateOnScreen().

Control del teclado

PyAutoGUI también tiene funciones para enviar pulsaciones virtuales a tu ordenador, lo que te permite rellenar formularios o introducir texto en aplicaciones.

Enviar una cadena desde el teclado

La función pyautogui.typewrite() envía pulsaciones virtuales al ordenador. Lo que hacen estas pulsaciones depende de qué ventana y campo de texto tienen el foco. Es posible que desee enviar primero un clic de ratón en el campo de texto que desea con el fin de asegurarse de que tiene el foco.

Como ejemplo sencillo, utilicemos Python para escribir automáticamente las palabras *¡Hola mundo!* en una ventana del editor de archivos. Primero, abre una nueva ventana de edición de archivos y colócala en la esquina superior izquierda de tu pantalla para que PyAutoGUI haga clic en el lugar correcto para enfocarla. A continuación, introduce lo siguiente en el intérprete de comandos interactivo:

```
>>> pyautogui.click(100, 100); pyautogui.typewrite('¡Hola mundo!')
```

Fíjese en cómo colocar dos comandos en la misma línea, separados por un punto y coma, evita que el shell interactivo le pida que introduzca datos entre la ejecución de las dos instrucciones. Esto evita que accidentalmente traigas una nueva ventana al foco entre las llamadas click() y typewrite(), lo que estropearía el ejemplo.

Python enviará primero un clic virtual del ratón a las coordenadas (100, 100), que debería hacer clic en la ventana del editor de archivos y ponerla en foco. La llamada a typewrite() enviará el texto *¡Hola mundo!* a la ventana, haciendo que se vea como en la Figura 18-3. ¡Ahora tiene código que puede escribir por usted!

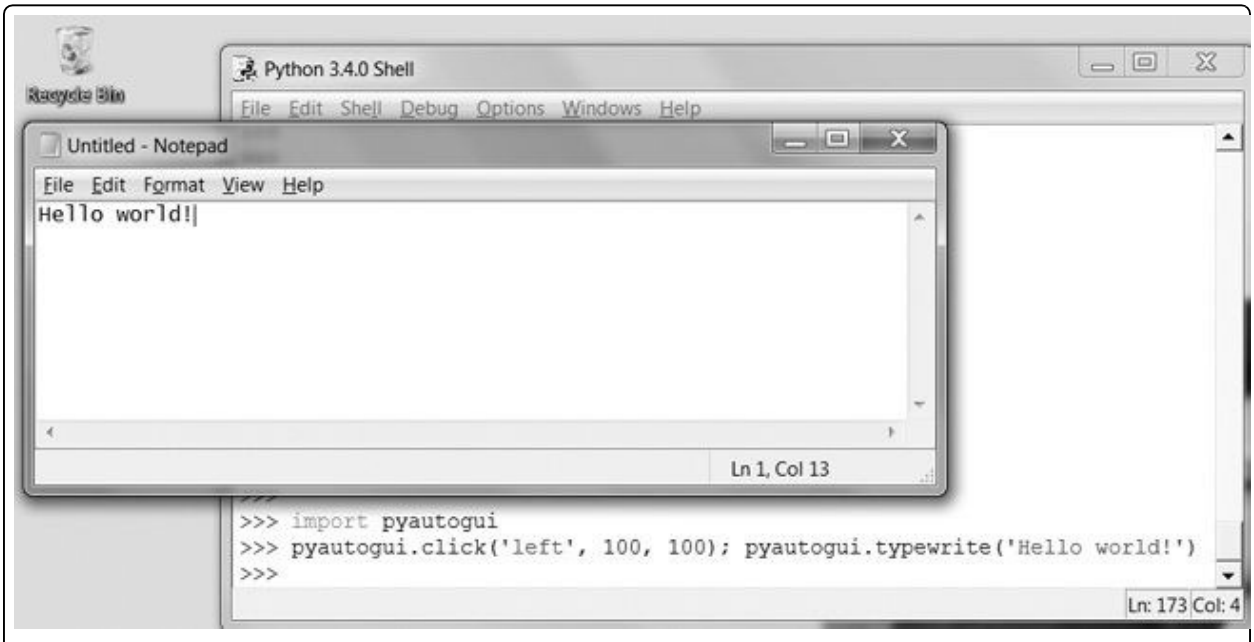


Figura 18-3. Usando PyAutogGUI para hacer clic en la ventana del editor de archivos y escribir en ella ¡Hola mundo!

Por defecto, la función `typewrite()` escribirá la cadena completa instantáneamente. Sin embargo, puede pasar un segundo argumento opcional para añadir una breve pausa entre cada carácter. Este segundo argumento es un valor entero o flotante del número de segundos de pausa. Por ejemplo, `pyautogui.typewrite('¡Hola mundo!', 0.25)` esperará un cuarto de segundo después de teclear *H*, otro cuarto de segundo después de *e*, y así sucesivamente. Este efecto gradual de máquina de escribir puede ser útil para aplicaciones lentas que no pueden procesar las pulsaciones de teclas lo suficientemente rápido como para seguir el ritmo de PyAutoGUI.

Para caracteres como *A* o *!*, PyAutoGUI simulará automáticamente mantener pulsada la tecla SHIFT también.

Nombres clave

No todas las teclas son fáciles de representar con caracteres de texto simples. Por ejemplo, ¿cómo representar SHIFT o la tecla de flecha izquierda como un único carácter? En PyAutoGUI, estas teclas del teclado se representan con valores de cadena cortos: `'esc'` para la ESC o `'enter'` para la tecla ENTER.

En lugar de un único argumento de cadena, se puede pasar a typewrite() una lista de estas cadenas de teclas del teclado. Por ejemplo, la siguiente llamada pulsa la tecla A, luego la tecla B, luego la tecla de flecha izquierda dos veces, y finalmente las teclas X e Y:

```
>>> pyautogui.typewrite(['a', 'b', 'izquierda', 'izquierda', 'X', 'Y'])
```

Como al pulsar la tecla de flecha izquierda se mueve el cursor del teclado, la salida será *XYab*. La Tabla 18-1 lista las cadenas de teclas de PyAutoGUI que puedes pasar a typewrite() para simular la pulsación de cualquier combinación de teclas.

También puedes examinar la lista pyautogui.KEYBOARD_KEYS para ver todas las posibles cadenas de teclas que PyAutoGUI aceptará. La cadena 'shift' se refiere a la tecla SHIFT izquierda y es equivalente a `'shiftleft'`. Lo mismo se aplica para las cadenas `'ctrl'`, `'alt'`, y `'win'`; todas se refieren a la tecla izquierda.

Tabla 18-1. Atributos de PyKeyboard

Cadena de teclas del teclado	Significado
'a', b', 'c', 'A', 'B', 'C', '1', '2', '3', '!', '@', '#', etc.	Las teclas para caracteres individuales
'enter' (o 'return' o '\n')	La tecla ENTER
'esc'	La tecla ESC
'shiftleft', 'shiftright'	Las teclas SHIFT izquierda y derecha
altleft, altright	Las teclas ALT izquierda y derecha
ctrlizquierda, ctrlderecha	Las teclas CTRL izquierda y derecha
'tab' (o '\t')	La tecla TAB
retroceso", "suprimir	Las teclas RETROCESO y SUPR

pageup', 'pagedown'	Las teclas PAGE UP y PAGE DOWN
inicio', 'fin'	Las teclas HOME y END
arriba, abajo, izquierda, derecha	Las teclas de flecha arriba, abajo, izquierda y derecha
f1, f2, f3, etc.	Las teclas F1 a F12
'volumemute", "volumedown", "volumeup"	Las teclas para silenciar, bajar el volumen y subir el volumen (algunos teclados no tienen estas teclas, pero su sistema operativo podrá seguir entendiendo estas pulsaciones simuladas).
pausa	La tecla PAUSA
capslock', 'numlock", 'scrolllock'	Teclas BLOQ MAYÚSCULAS, BLOQ NUM y BLOQ DESPLAZAMIENTO
"insertar	La tecla INS o INSERT
'printscreen'	La tecla PRTSC o IMPRIMIR PANTALLA
winleft', 'winright'	Las teclas WIN izquierda y derecha (en Windows)
comando	La tecla Command (⌘) (en OS X) 'option' La tecla OPTION (en OS X)

Pulsar y soltar el teclado

Al igual que las funciones mouseDown() y mouseUp()pyautogui.keyDown() y pyautogui.keyUp() enviarán pulsaciones y liberaciones de teclas virtuales al ordenador. Se les pasa una cadena de teclas (ver Tabla 18-1) como argumento. Para mayor comodidad, PyAutoGUI proporciona la función pyautogui.press(), que llama a estas dos funciones para simular la pulsación completa de una tecla.

Ejecute el siguiente código, que escribirá un carácter de signo de dólar (obtenido manteniendo pulsada la tecla MAYÚS y pulsando 4):

```
>>> pyautogui.keyDown('shift'); pyautogui.PRESS('4'); pyautogui.keyUp('shift')
```

Esta línea pulsa SHIFT, pulsa (y suelta) 4, y luego suelta SHIFT. Si necesita escribir una cadena en un campo de texto, la función typewrite() es más adecuada. Pero para aplicaciones que toman comandos de una sola tecla, la función press() es el enfoque más simple.

Combinaciones de teclas de acceso rápido

Una *tecla de acceso rápido* o *atajo* es una combinación de teclas para invocar alguna función de la aplicación. La tecla de acceso rápido habitual para copiar una selección es CTRL-C (en Windows y Linux) o⌘ -C (en OS X). El usuario pulsa y mantiene pulsada la CTRL, luego pulsa la tecla C y, a continuación, suelta las teclas C y CTRL. Para hacer esto con
las funciones keyDown() y keyUp() de PyAutoGUI, tendrías que introducir lo siguiente:

```
pyautogui.keyDown('ctrl')
pyautogui.keyDown('c')
pyautogui.keyUp('c')
pyautogui.keyUp('ctrl')
```

Esto es bastante complicado. En su lugar, utiliza la función pyautogui.hotkey(), que toma múltiples argumentos de cadena de teclas del teclado, las pulsa en orden y las suelta en el orden inverso. Para el ejemplo de CTRL-C, el código sería simplemente el siguiente:

```
pyautogui.hotkey('ctrl', 'c')
```

Esta función es especialmente útil para combinaciones de teclas de acceso rápido más grandes. En Word, la combinación de teclas CTRL-ALT-SHIFT-S muestra el panel Estilo. En lugar de hacer ocho llamadas a funciones diferentes (cuatro llamadas a keyDown() y cuatro llamadas a keyUp()), puede llamar simplemente a **hotkey('ctrl', 'alt', 'shift', 's').**

Con una nueva ventana del editor de archivos IDLE en la esquina superior izquierda de su pantalla, introduzca lo siguiente en el intérprete de comandos interactivo (en OS , sustituya **'alt'** por **'ctrl'):**

```
>>> importar pyautogui, tiempo
    >>> def commentAfterDelay():
❶           pyautogui.click(100, 100)
❷         pyautogui.typewrite('En IDLE, Alt-3 comenta una línea.')
          time.sleep(2)
➲         pyautogui.hotkey('alt', '3')

    >>> commentAfterDelay()
```

Esto define una función commentAfterDelay() que, cuando se llama, hará clic en la ventana del editor de archivos para enfocarla ❶, escribirá *En IDLE, Atl-3 comenta una línea* ❷, hará una pausa de 2 segundos, y luego simulará pulsar la tecla de acceso rápido ALT-3 (o CTRL-3 en OS X) ❸. Este atajo de teclado añade dos caracteres # a la línea actual, . (Este es un truco útil para saber cuando se escribe su propio

código en IDLE).

Revisión de las funciones de PyAutoGUI

Dado que este capítulo abarca muchas funciones diferentes, a continuación se ofrece un resumen rápido de referencia:

- **moveTo(x, y)**. Mueve el cursor del ratón a las coordenadas *x* e *y* dadas.
- **moveRel(xDESPLAZAMIENTO, yDESPLAZAMIENTO)**. Mueve el cursor del ratón respecto a su posición actual.
- **arrastrarHacia(x, y)**. Mueve el cursor del ratón mientras se mantiene pulsado el botón izquierdo.
- **dragRel(xDESPLAZAMIENTO, yDESPLAZAMIENTO)**. Mueve el cursor del ratón respecto a su posición actual mientras se mantiene pulsado el botón izquierdo.
- **click(x, y, botón)**. Simula un clic (botón izquierdo por defecto).
- **rightClick()**. Simula el clic del botón derecho.
- **middleClick()**. Simula un en el botón central.
- **doubleClick()**. Simula un doble del botón izquierdo.
- **mouseDown(x, y, botón)**. Simula la pulsación del botón dado en la posición *x*, *y*.
- **mouseUp(x, y, botón)**. Simula soltar el botón dado en la posición *x*, *y*.
- **scroll(UNIDADES)**. Simula la rueda de desplazamiento. Un argumento positivo desplaza hacia arriba; un argumento negativo desplaza hacia abajo.
- **typewrite(mensaje)**. Escribe los caracteres de la cadena de mensajes dada.
- **typewrite([tecla1, tecla2, tecla3])**. Escribe las cadenas de teclas del teclado dadas.
- **PULSAR(tecla)**. Pulsa la cadena de teclas del teclado dada.
- **keyDown(tecla)**. Simula la pulsación de la tecla dada.
- **keyUp(tecla)**. Simula la pulsación de la tecla dada.
- **hotkey([tecla1, tecla2, tecla3])**. Simula pulsar las cadenas de teclas del teclado dadas en orden y luego soltarlas en orden inverso.
- **CAPTURA DE PANTALLA()**. Devuelve una captura de pantalla como un Imagen. (Ver Capítulo 17 para información sobre objetos Imagen).

Proyecto: Rellenador automático de formularios

De todas las tareas aburridas, rellenar formularios es la más temida. Es lógico que ahora, en el proyecto del capítulo final, vayas a acabar con ella. Supongamos que tienes una gran cantidad de datos en una hoja de cálculo y tienes que volver a escribirlos tediosamente en la interfaz de formularios de otra aplicación, sin que ningún becario lo haga por ti. Aunque algunas aplicaciones tienen una función de importación que te permite cargar una hoja de cálculo con la información, a veces parece que no hay otra manera que hacer clic y escribir sin pensar durante horas y . Has llegado hasta aquí en este libro; sabes que*, por supuesto,* hay otra manera.

El formulario para este proyecto es un formulario de Google Docs que puede encontrar en http://nostarch.com/automatestuff. Se parece a la Figura 18-4.

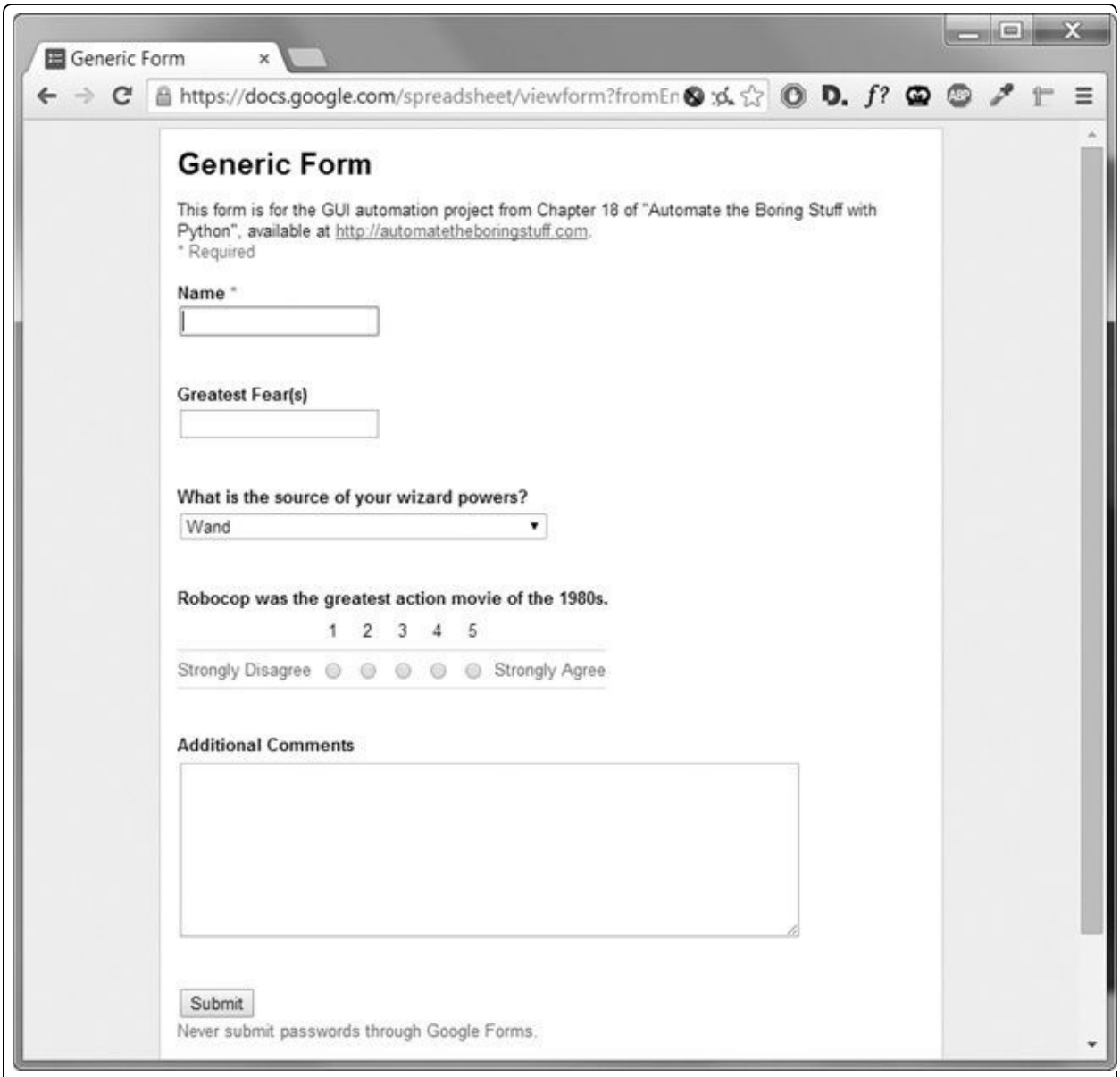


Figura 18-4. El formulario utilizado para este proyecto

A grandes , esto es lo que debe hacer tu programa:

- Haga clic en el primer campo de texto del formulario.
- Desplácese por el formulario, introduciendo información en cada campo.
- Pulse el botón Enviar.
- Repita el proceso con la siguiente serie de datos.

Esto significa que su código tendrá que hacer lo siguiente:

- Llama a pyautogui.click() para hacer clic en el formulario y en el botón Enviar.

- Llama a pyautogui.typewrite() para introducir texto en los campos.
- Maneja la excepción KeyboardInterrupt para que el usuario pueda presionar CTRL-C para salir.

Abre una nueva ventana del editor de archivos y guárdalo como *formFiller.py*.

Paso 1: Determinar los pasos

Antes de escribir el código, hay que averiguar las pulsaciones exactas de las teclas y los clics que rellenarán el formulario una vez. El script *mouseNow.py* en Project: "¿Dónde está el ratón ahora mismo?" puede ayudarte a averiguar las específicas del ratón coordenadas del ratón. Sólo necesitas conocer las coordenadas del primer campo de texto. Después de hacer clic en el primer campo, sólo tienes que pulsar TAB para mover el foco al siguiente campo.
Esto le ahorrará tener que averiguar las coordenadas x e y para hacer clic en cada campo.

Estos son los pasos para introducir datos en el formulario:

1. Haz clic en el campo Nombre. (Utiliza *mouseNow.py* para determinar las coordenadas después de maximizar la ventana del navegador. En OS X, es posible que tengas que hacer clic dos veces: una para poner el navegador en foco y otra para hacer clic en el campo Nombre).
2. Escriba un nombre y pulse TAB.
3. Escribe tu mayor temor y pulsa TAB.
4. Pulsa la tecla de flecha abajo el número de veces correcto para seleccionar la fuente de energía del mago: una vez para *varita*, dos veces para *amuleto*, tres veces para *bola de cristal* y cuatro veces para *dinero*. A continuación, pulsa TAB. (Ten en cuenta que en OS X, tendrás que pulsar la tecla de flecha abajo una vez más para cada opción. En algunos navegadores, es posible que también tengas que pulsar la ENTER).
5. Pulsa la tecla de flecha derecha para seleccionar la respuesta a la pregunta sobre RoboCop. Púlsala una vez para *2*, dos veces para *3*, tres veces para *4* o cuatro veces para *5*; o simplemente pulsa la barra espaciadora para seleccionar *1* (que aparece resaltado por defecto). A continuación, pulse
   TAB.
6. Escriba un comentario adicional y pulse TAB.
7. Pulse la tecla ENTER para "hacer clic" en el botón Enviar.
8. Tras enviar el formulario, el navegador le llevará a una página en la que deberá hacer clic en un enlace para volver a la página del formulario.

Tenga en cuenta que si ejecuta este programa de nuevo más tarde, puede que tenga que actualizar las coordenadas del clic del ratón, ya que la ventana del navegador puede haber cambiado de posición. Para evitar esto, asegúrese siempre de que la ventana del navegador está maximizada antes de encontrar las coordenadas del primer campo del formulario. Además, los diferentes navegadores en diferentes sistemas operativos pueden funcionar de forma ligeramente diferente a los pasos que se dan aquí, así que compruebe que estas combinaciones de teclas funcionan en su ordenador antes de ejecutar el programa.

Paso 2: Establecer coordenadas

Carga el formulario de ejemplo que has descargado (Figura 18-4) en un navegador y maximiza la ventana de tu navegador. Abra una nueva ventana de Terminal o de línea de comandos para ejecutar el script *mouseNow.py*, y luego pase el ratón sobre el campo Nombre para averiguar sus coordenadas x e y. Estos números se asignarán a la variable nameField en tu programa. Además, averigua las coordenadas x e y y Valor de la tupla RGB del botón azul Enviar. Estos valores se asignarán a la submitButton y submitButtonColor, respectivamente.

A continuación, rellena algunos datos ficticios para el formulario y haz clic en **Enviar**. Necesitas ver el aspecto de la siguiente página para poder utilizar *mouseNow.py* para encontrar las coordenadas del enlace *Enviar otra respuesta* en esta nueva página.

Haz que tu código fuente tenga el siguiente aspecto, asegurándote de sustituir todos los valores en cursiva por las coordenadas que has determinado a partir de tus propias pruebas:

```
#! python3
# formFiller.py - Rellena automáticamente el

formulario. import pyautogui, time

# Establece las coordenadas correctas para tu ordenador.
nameField= (648, 319)
submitButton= (651, 817)
submitButtonColor= (75, 141, 249)
submitAnotherLink= (760, 224)

# TODO: Dar al usuario la oportunidad de matar el

script. # TODO: Esperar hasta que la página del

formulario se haya cargado.

# TODO: Rellenar el campo Nombre.

# TODO: Rellenar el campo Mayor(es) Miedo(s).

# TODO: Rellenar el campo Fuente de Poderes delMago. #

TODO: Rellenar el campo RoboCop.

# TODO: Rellene el campo Comentarios adicionales.

# TODO: Haga clic en Enviar.

# TODO: Esperar hasta que la página del formulario se haya cargado.
```

```
# TODO: Haga clic en el enlace Enviar otra respuesta.
```

Ahora necesitas los datos que realmente quieres introducir en este formulario. En el mundo real, estos datos podrían venir de una hoja de cálculo, un archivo de texto plano, o un sitio web, y requeriría código adicional para cargar en el programa. Pero para este proyecto, simplemente codificarás todos estos datos en una variable. Añade lo siguiente a tu programa:

```
#! python3
# formFiller.py - Rellena automáticamente el formulario.

--snip--

formData= [{'name': 'Alice', 'fear': 'EAVESDROPPERS', 'source': 'wand', 'robocop': 4,
            'COMENTARIOS': 'Saluda a Bob de mi parte'},
            {'nombre': 'Bob', 'miedo': 'ABEJAS', 'fuente': 'amuleto', 'robocop':
            4, 'COMMENTS': 'n/a'},
            {'name': 'Carol', 'fear': 'PUPPETS', 'source': 'crystal ball',
            'robocop': 1, 'COMMENTS': 'Por favor, saca las MARIONETAS de la sala
            de descanso.'},
            {'nombre': 'Alex Murphy', 'miedo': 'ED-209', 'fuente': 'dinero',
            'robocop': 5, 'COMENTARIOS': 'Proteger a los inocentes. Servir a la confianza
            pública. Uphold the .'},
           ]

--snip--
```

La lista formData contiene cuatro diccionarios para cuatro nombres diferentes. Cada diccionario tiene nombres de campos de texto como claves y respuestas como valores. El último paso es configurar la variable PAUSE de PyAutoGUI para que espere medio segundo después de cada llamada a una función. Añade lo siguiente a tu programa después de la sentencia de asignación formData:

```
pyautogui.PAUSE= 0.5
```

Paso 3: Empezar a introducir datos

Un bucle for iterará sobre cada uno de los diccionarios de la lista formData, pasando los valores del diccionario a las funciones PyAutoGUI que escribirán virtualmente en los campos de texto.

Añade el siguiente código a tu programa:

```
#! python3
    # formFiller.py - Rellena automáticamente el formulario.

    --snip--

    para persona en formData:
        # Dar al usuario la oportunidad de matar el script.
        print('>>> 5 SEGUNDOS DE PAUSA PARA QUE EL USUARIO PRESIONE CTRL-C <<<')
❶       time.sleep(5)

        # Espera hasta que la página del formulario se HAYA cargado.
❷       while not pyautogui.PIXELMATCHESCOLOR(submitButton[0], submitButton[1],
        submitButtonColor):
            time.sleep(0.5)

    --snip--
```

Como pequeña de seguridad, el script tiene una pausa de cinco segundos ❶ que da al usuario la oportunidad de pulsar CTRL-C (o mover el cursor del ratón a la esquina superior izquierda de la pantalla para lanzar la excepción FailSafeException) para cerrar el programa en caso de esté haciendo algo inesperado. A continuación, el programa espera hasta que el botón Submit
el color del botón es visible❷, haciendo saber al programa que la página del formulario tiene cargado. Recuerda que averiguaste la información de coordenadas y color de en el paso 2 y la almacenaste en las variables submitButton y submitButtonColor. Para usar pixelMatchesColor(), pasa las coordenadas submitButton[0] y submitButton[1], y el color submitButtonColor.

Después del código que espera hasta que el color del botón Enviar sea visible, añade lo siguiente:

```
#! python3
    # formFiller.py - Rellena automáticamente el formulario.

    --snip--

❶       print('Entrando %s info...' % (persona['nombre']))
❷       pyautogui.click(nombreCampo[0], nombreCampo[1])
```

```
        # Rellene el campo Nombre.
➲       pyautogui.typewrite(persona['nombre'] + '\t')

        # Rellene el campo Mayor(es) Miedo(s).
❹       pyautogui.typewrite(persona['miedo'] + '\t')

    --snip--
```

Añadimos una llamada ocasional a `print()` para mostrar el estado del programa en su ventana de Terminal para que el usuario sepa lo que está pasando ❶.

Dado que el programa sabe que el formulario está cargado, es el momento de llamar a `click()` para hacer clic en el campo Nombre ❷ y a `typewrite()` para introducir la cadena en `persona["nombre"]`

❸. El carácter `'\t'` se añade al final de la cadena pasada a `typewrite()` para simular la pulsación de TAB, que mueve el foco del teclado al siguiente campo, Mayor(es) Miedo(s). Otra llamada a `typewrite()` escribirá la cadena en `person['fear']` en este campo y luego tabulará al siguiente campo de la forma❹.

Paso 4: Manejar listas de selección y botones de radio

El menú desplegable de la pregunta "poderes del mago" y los botones de opción del campo RoboCop son más complicados de manejar que los campos de texto. Para hacer clic en estas opciones con el ratón, tendrías que averiguar las coordenadas x e y de cada opción posible. Es más fácil utilizar las flechas del teclado para hacer una selección.

Añade lo siguiente a tu programa:

```
#! python3
    # formFiller.py - Rellena automáticamente el formulario.

    --snip--

        # Rellene el campo Fuente de PODERES del asistente.
❶       si persona['fuente']== 'varita':
❷           pyautogui.typewrite(['down', '\t']) elif
        persona['fuente']== 'amuleto':
            pyautogui.typewrite(['abajo', 'abajo', '\t']) elif
        persona['fuente']== 'bola de cristal':
            pyautogui.typewrite(['abajo', 'abajo', 'abajo', '\t']) elif
        persona['fuente']== 'dinero':
            pyautogui.typewrite(['abajo',    'abajo',    'abajo',    'abajo',    '\t'])

        # Rellena el campo RoboCop.
➲       si persona['robocop']== 1:
❹           pyautogui.typewrite([' ', '\t']) elif
        persona['robocop']== 2:
            pyautogui.typewrite(['derecha', '\t'])
        elif persona['robocop']== 3:
            pyautogui.typewrite(['derecha', 'derecha', '\t'])
        elif persona['robocop']== 4:
            pyautogui.typewrite(['derecha', 'derecha', 'derecha',
        '\t']) elif persona['robocop']== 5:
            pyautogui.typewrite(['derecha', 'derecha', 'derecha', 'derecha', '\t'])

    --snip--
```

Una vez que el menú desplegable tiene el foco (recuerde que escribió código para simular que pulsa TAB después de rellenar el campo Mayor(es) Miedo), pulsando la tecla de flecha abajo se moverá al siguiente elemento de la lista de selección. Dependiendo del valor en persona['fuente'], su programa debería enviar un número de pulsaciones de flecha abajo antes de tabular al siguiente campo. Si el valor de la tecla 'fuente' en

el diccionario de este es 'varita'❶, simulamos pulsar la tecla de flecha abajo una vez (para seleccionar *Varita*) y pulsar TAB❷. Si el valor de la tecla 'fuente' es 'amuleto', simulamos pulsar la tecla de flecha abajo dos veces y pulsar TAB, y así sucesivamente para las demás respuestas posibles.

Los botones de opción de la pregunta sobre RoboCop pueden seleccionarse con las teclas de flecha derecha -o, si desea seleccionar la primera opción❸, con sólo pulsar la barra espaciadora-.

❹.

Paso 5: Envíe el formulario y espere

Puede rellenar el campo Comentarios adicionales con la función typewrite() pasando person["comments"] como argumento. Puedes escribir un '\t' adicional para mover el foco del teclado al siguiente campo o al Enviar. Una vez que el botón Enviar está en el foco, llamar a pyautogui.press('enter') simulará pulsar la tecla ENTER y enviar el formulario. Después de enviar el formulario, tu programa esperará cinco segundos a que se cargue la siguiente página.

Una vez cargada la nueva página, tendrá un enlace *Submit another response* dirigirá al navegador a una nueva página de formulario vacía. Usted almacenó las coordenadas de este enlace como una tupla en submitAnotherLink en el paso 2, así que pase estas coordenadas a pyautogui.click() para hacer clic en este enlace.

Con el nuevo formulario listo, el bucle for externo del script puede continuar con la siguiente iteración e introducir la información de la siguiente persona en el formulario.

Completa tu programa añadiendo el siguiente código:

```
#! python3
# formFiller.py - Rellena automáticamente el formulario.

--snip--

    # Rellena el campo COMENTARIOS ADICIONALES.
    pyautogui.typewrite(persona['COMENTARIOS']+ '\t')

    # Haz clic en Enviar.
    pyautogui.PRESS('enter')

    # Espera hasta que la página del
    formulario se HAYA cargado.
    print('Clicked Submit.')
    time.sleep(5)

    # Haz clic en el enlace Enviar otra respuesta.
    pyautogui.click(submitAnotherLink[0], submitAnotherLink[1])
```

Una vez finalizado el bucle for principal, el programa habrá introducido la información de cada persona. En este ejemplo, sólo hay que introducir los datos de cuatro personas. Pero si tuvieras *4.000* personas, ¡escribir un programa para hacer esto te ahorraría mucho tiempo y mecanografía!

Resumen

La automatización de GUI con el módulo pyautogui le permite interactuar con aplicaciones en su ordenador controlando el ratón y el teclado. Si bien este enfoque es lo suficientemente flexible como para hacer cualquier cosa que un usuario humano pueda hacer, el inconveniente es que estos programas son bastante ciegos a lo que están haciendo clic o escribiendo. Cuando escriba programas de automatización GUI, asegúrese de que se bloquean rápidamente si se les dan malas instrucciones. El bloqueo es molesto, pero es mucho mejor que el programa continúe por error.

Puedes mover el cursor del ratón por la pantalla y simular clics, pulsaciones y atajos de teclado con PyAutoGUI. El módulo pyautogui también puede comprobar los colores de la pantalla, lo que puede proporcionar a tu programa de automatización GUI una idea suficiente del contenido de la pantalla para saber si tiene
se ha desviado. Incluso puedes darle a PyAutoGUI una captura de pantalla y dejar que averigüe las coordenadas de la zona en la que quieres hacer clic.

Puedes combinar todas estas características de PyAutoGUI para automatizar cualquier tarea repetitiva en tu ordenador. De hecho, puede resultar hipnótico ver cómo el cursor del ratón se mueve por sí solo y cómo aparece texto en la pantalla automáticamente. ¿Por qué no emplear el tiempo que has ahorrado en sentarte y ver cómo tu programa hace todo el trabajo por ti? Hay una cierta satisfacción al ver cómo tu ingenio te ha salvado de las cosas aburridas.

Preguntas prácticas

P: 1. ¿Cómo puedes activar el fail safe de PyAutoGUI para detener un programa?

P: 2. ¿Qué función devuelve la resolución actual()?

P: 3. ¿Qué función devuelve las coordenadas de la posición actual del cursor del ratón?

P: 4. ¿Cuál es la diferencia entre pyautogui.moveTo() y pyautogui.moveRel()?

P: 5. ¿Qué funciones se pueden utilizar para arrastrar el ratón?

P: 6. ¿Qué llamada a función escribirá los caracteres de "¡Hola mundo!"?

P: 7. ¿Cómo se pueden realizar pulsaciones de teclas especiales, como la tecla de flecha izquierda del teclado?

P: 8. ¿Cómo se puede guardar el contenido actual de la pantalla en un archivo de imagen llamado *screenshot.png*?

P: 9. ¿Qué código establecería una pausa de dos segundos después de cada llamada a una función PyAutoGUI?

Proyectos prácticos

Para practicar, escribe programas que hagan lo siguiente.

Ocupado

Muchos programas de mensajería instantánea determinan si estás inactivo, o alejado del ordenador, detectando la falta de movimiento del ratón durante un cierto periodo de tiempo -digamos, diez minutos-. Tal vez te gustaría alejarte de tu escritorio durante un rato, pero no quieres que los demás vean que tu estado de mensajería instantánea pasa al modo inactivo. Escribe un script que mueva ligeramente el cursor del ratón cada diez segundos. El empujón debe ser lo suficientemente pequeño como para que no estorbe si necesitas utilizar el ordenador mientras se ejecuta el script.

Bot de mensajería instantánea

Google Talk, Skype, Yahoo Messenger, AIM y otras aplicaciones de mensajería instantánea a menudo utilizan protocolos propietarios que dificultan que otros escriban módulos Python que puedan interactuar con estos programas. Pero incluso estos protocolos propietarios no pueden escribir una herramienta de automatización GUI.

La aplicación Google Talk tiene una barra de búsqueda que te permite introducir un nombre de usuario de tu lista de amigos y abrir una ventana de mensajería al pulsar INTRO. El foco del teclado se mueve automáticamente a la nueva ventana. Otras aplicaciones de mensajería instantánea tienen formas similares de abrir nuevas ventanas de mensajes. Escribe un programa que envíe automáticamente un mensaje de notificación a un grupo selecto de personas de tu lista de amigos. Es posible que tu programa tenga que hacer frente a casos excepcionales, como que los amigos estén desconectados, que la ventana de chat aparezca en diferentes coordenadas de la pantalla o que haya cuadros de confirmación que interrumpan la mensajería. Tu programa tendrá que hacer capturas de pantalla para guiar su interacción GUI y adoptar formas de detectar cuándo no se están enviando sus pulsaciones virtuales.

NOTA

Es posible que quieras crear algunas cuentas de prueba falsas para no enviar spam accidentalmente a tus amigos reales mientras escribes este programa.

Tutorial de bot de juego

Hay un gran tutorial titulado "How to Build a Python Bot That Can Play Web Games" que puedes encontrar en http://nostarch.com/automatestuff/. Este tutorial explica cómo crear un programa de automatización GUI en Python que juega un juego Flash llamado Sushi Go Round. El juego consiste en pulsar los botones de los ingredientes correctos para rellenar los pedidos de sushi de los clientes. Cuanto más rápido se rellenan los pedidos sin errores, más puntos se obtienen. Esta es una tarea perfectamente adecuada para un programa de automatización GUI, ¡y una forma de hacer trampas para conseguir una puntuación alta! El tutorial cubre muchos de los mismos temas que este capítulo, pero también incluye descripciones de las características básicas de reconocimiento de imagen de PyAutoGUI.

Apéndice A. Instalación de módulos de terceros

Además de la biblioteca estándar de módulos incluidos en Python, otros desarrolladores han escrito sus propios módulos para ampliar aún más las capacidades de Python. La principal forma de instalar módulos de terceros es utilizar la herramienta pip de Python. Esta herramienta descarga e instala de forma segura módulos de Python en tu ordenador desde https://pypi.python.org/, el sitio web de la Python Software Foundation. PyPI, o Python Package Index, es una especie de tienda de aplicaciones gratuita para módulos de Python.

La herramienta pip

El archivo ejecutable de la herramienta pip se llama *pip* en Windows y *pip3* en OS X y Linux. En Windows, puede encontrar pip en *C:\Python34\Scripts\pip.exe*. En OS X, se encuentra en *LibraryFrameworks/Python.framework/Versions/3.4/bin/pip3*. En Linux, está en *usrbin/pip3*.

Mientras que pip viene instalado automáticamente con Python 3.4 en Windows y OS X, debes instalarlo por separado en Linux. Para instalar pip3 en Ubuntu o Debian Linux, abra una nueva ventana de Terminal e introduzca **sudo apt-get install python3-pip**. Para instalar pip3 en Fedora Linux, introduzca **sudo yum install python3 -pip** en una ventana de Terminal. Deberá introducir la contraseña de administrador de su ordenador para poder instalar este software.

Instalación de módulos de terceros

La herramienta pip está pensada para ejecutarse desde la línea de comandos: Le pasas el comando `install` seguido del nombre del módulo que quieres instalar. Por ejemplo, en Windows escribirías `pip install` *NombreMódulo*, donde *NombreMódulo* es el nombre del módulo. En OS X y Linux, tendrá que ejecutar pip3 con el prefijo `sudo` para conceder privilegios administrativos para instalar el . Tendría que escribir `sudo pip3 install` *NombreModulo*.

Si ya tiene instalado el módulo pero desea actualizarlo a la última versión disponible en PyPI, ejecute `pip install -U` *NombreMódulo* (o `pip3 install -U` *NombreMódulo* en OS X y Linux).

Después de instalar el módulo, puede comprobar que se ha instalado correctamente ejecutando `import` *NombreMódulo* en el intérprete de comandos interactivo. Si no ningún mensaje de error, puedes asumir que el módulo se ha instalado correctamente.

Puedes instalar todos los módulos cubiertos en este libro ejecutando los comandos listados a continuación. (Recuerda sustituir `pip` por `pip3` si estás en OS X o Linux).

- `pip install send2trash`
- `pip install solicitudes`
- `pip install beautifulsoup4`
- `pip install selenium`
- `pip install openpyxl`
- `pip install PyPDF2`
- `pip install python-docx` (instalar `python-docx`, no `docx`)
- `pip install imapclient`
- `pip install pyzmail`
- `pip install twilio`
- `pip install almohada`
- `pip install pyobjc-core` (sólo en OS X)
- `pip install pyobjc` (sólo en OS X)

- pip install python3-xlib (sólo en Linux)
- pip install pyautogui

NOTA

Para usuarios de OS X: El módulo pyobjc puede tardar 20 minutos o más en instalarse, así no te alarmes si tarda un poco. También deberías instalar primero el módulo pyobjc-core, que reducirá el tiempo total de instalación.

Apéndice B. Programas en ejecución

Si tienes un programa abierto en el editor de archivos de IDLE, ejecutarlo es una simple cuestión de pulsar F5 o seleccionar la opción de menú Ejecutar► Ejecutar Módulo. Esta es una manera fácil de ejecutar programas mientras los escribes, pero abrir IDLE para ejecutar tus programas terminados puede ser una carga. Hay formas más convenientes de ejecutar scripts Python.

Línea Shebang

La primera línea de todos tus programas Python debe ser una línea shebang, que indica a tu ordenador que quieres que Python ejecute este programa. La línea shebang comienza con **#!**, pero el resto depende de tu sistema operativo.

- En Windows, la línea shebang es **#! python3**.
- En OS , la línea shebang es **#! usrbin/env python3**.
- En Linux, la línea shebang es **#! usrbin/python3**.

Podrá ejecutar scripts Python desde IDLE sin la línea shebang, pero la línea es necesaria para ejecutarlos desde la línea de comandos.

Ejecutar programas Python en Windows

En Windows, el intérprete de Python 3.4 se encuentra en *C:\Python34\python.exe*. Alternativamente, el conveniente programa *py*.exe leerá la línea shebang en la parte superior del código fuente del archivo *.py* y ejecutará la versión apropiada de Python para ese script. El programa *py*.exe se asegurará de ejecutar el programa Python con la versión correcta de Python si hay varias versiones instaladas en tu ordenador.

Para que sea más cómodo ejecutar tu programa Python, crea un *archivo .bat por lotes* para ejecutar el programa Python con *py.exe*. Para crear un archivo por lotes, crea un nuevo archivo de texto que contenga una sola línea como la siguiente:

```
@py.exe C:\path\to\your\pythonScript.py %*
```

Sustituye esta ruta por la ruta absoluta a tu propio programa, y guarda este archivo con una extensión *.bat* (por ejemplo, *pythonScript.bat*). Este archivo por lotes evitará que tengas que escribir la ruta absoluta completa del Python cada vez que quieras ejecutarlo. Te recomiendo que coloques todos tus archivos batch y *.py* en una sola carpeta, como *C:\MyPythonScripts* o *C:\Users\YourName\PythonScripts*.

La carpeta *C:\MyPythonScripts* debe añadirse a la ruta del sistema en Windows para que pueda ejecutar los archivos por lotes en ella desde el cuadro de diálogo Ejecutar. Para ello, modifique la variable de entorno PATH. Haga clic en el botón **Inicio** y escriba **Editar variables de entorno para su cuenta**. Esta opción debería autocompletarse cuando empieces a . La ventana de Variables de Entorno que aparece
tendrá el aspecto de la Figura B-1.

En Variables del sistema, seleccione la variable Ruta y haga clic en **Editar**. En el campo de texto Valor, añade un punto y coma, escribe **C:\MyPythonScripts** y haz clic en **Aceptar**. Ahora puedes ejecutar cualquier script de Python en la carpeta *C:\MyPythonScripts* simplemente pulsando WIN-R e introduciendo el nombre del script. Ejecutando pythonScript, por ejemplo, ejecutará *pythonScript.bat*, que a su vez le ahorrará tener que ejecutar todo el comando py.exe C:\MyPythonScripts\pythonScript.py desde el diálogo Ejecutar.

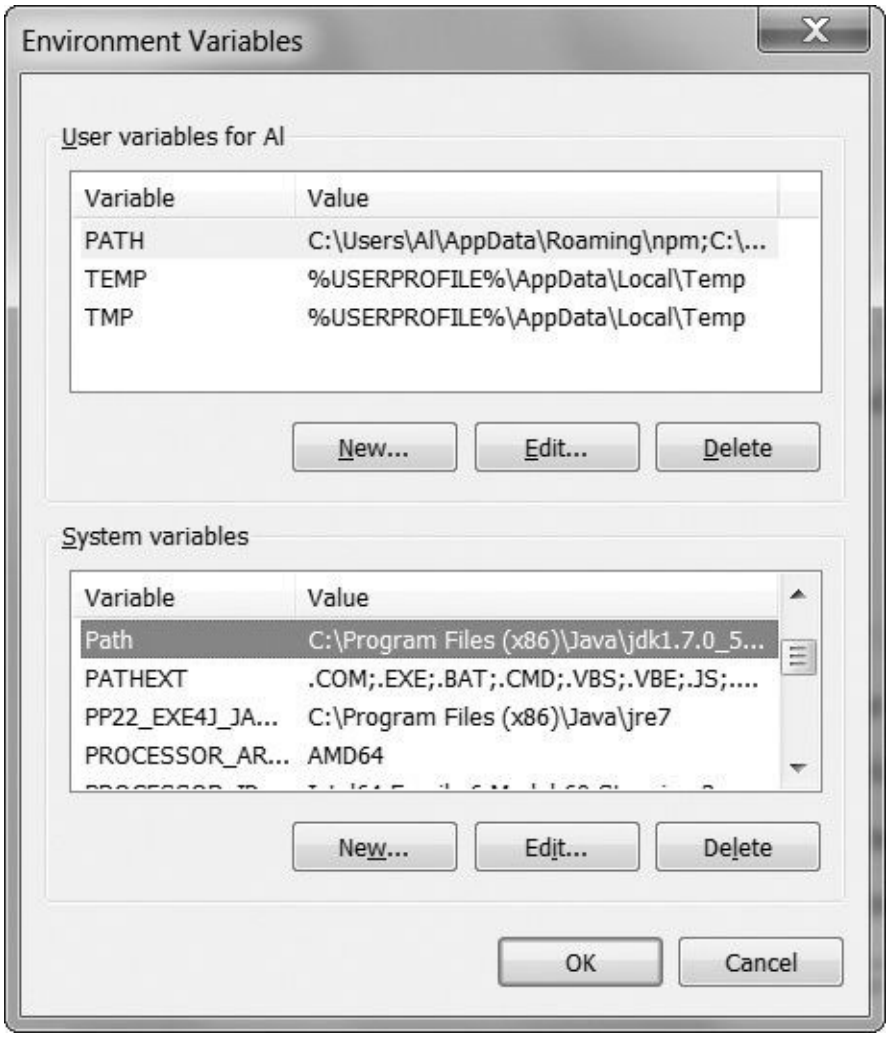


Figura B-1. La ventana de variables de entorno en Windows

Ejecutar programas Python en OS X y Linux

En OS X, seleccione Aplicaciones▸ Utilidades▸ Terminal para abrir una ventana *de Terminal*. Una ventana de Terminal es una forma de introducir comandos en tu ordenador utilizando sólo texto, en lugar de hacer clic a través de una interfaz gráfica. Para abrir la ventana Terminal en Ubuntu Linux, pulsa la tecla WIN (o SUPER) para abrir Dash y escribe **Terminal**.

La ventana de Terminal comenzará en la carpeta home de tu cuenta de usuario. Si mi nombre de usuario es *asweigart*, la carpeta de inicio será *Usersasweigart* en OS X y *homeasweigart* en Linux. El carácter tilde (~) es un atajo para su carpeta de inicio, por lo que puede introducir `cd~` para cambiar a su carpeta de inicio. También puede utilizar el comando `cd` para cambiar el directorio de trabajo actual a cualquier otro directorio. Tanto en OS X como en Linux, el comando `pwd` imprimirá el directorio de trabajo actual.

Para ejecutar tus programas Python, guarda tu archivo *.py* en tu carpeta de inicio. Luego, cambia los permisos del archivo *.py* para hacerlo ejecutable ejecutando `chmod +x pythonScript.py`. Los permisos de archivo están más allá del alcance de este libro, necesitarás ejecutar este comando en tu archivo Python si quieres ejecutar el programa desde la ventana Terminal. Una vez hecho esto, podrás ejecutar tu script siempre que quieras abriendo una ventana de Terminal e introduciendo `./pythonScript.py`. La línea shebang en la parte superior de la secuencia de comandos le dirá al sistema operativo dónde localizar el intérprete de Python.

Ejecutar programas Python con aserciones desactivadas

Puede desactivar las sentencias assert en sus programas Python para mejorar ligeramente el rendimiento. Cuando ejecute Python desde el terminal, incluya la sentencia -O después de python o python3 y antes del nombre del archivo *.py*. Esto ejecutará una versión optimizada de su programa que omite las comprobaciones de aserción.

Apéndice C. Respuestas a las preguntas prácticas

Este apéndice contiene las respuestas a los problemas prácticos que aparecen al final de cada capítulo. Le recomiendo encarecidamente que se tome su tiempo para resolver estos problemas. Programar es algo más que memorizar la sintaxis y una lista de nombres de funciones. Al igual que cuando aprendes un idioma extranjero, cuanta más práctica le dediques, más provecho le sacarás. También hay muchos sitios web con problemas de programación para practicar. Puedes encontrar una lista de ellos en http://nostarch.com/automatestuff/.

Capítulo 1

1. Los operadores son+ , -, * y /. Los valores son 'hola', -88,8 y 5.
2. La cadena es "spam"; la variable es spam. Las cadenas siempre empiezan y acaban entre comillas.
3. Los tres tipos de datos introducidos en este capítulo son enteros, números de coma flotante y cadenas.
4. Una expresión es una combinación de valores y operadores. Todas las expresiones se evalúan (es decir, se reducen) a un único valor.
5. Una expresión se evalúa con un único valor. Una expresión no.
6. La variable bacon se establece en 20. La expresión bacon+ 1 no reasigna el valor en bacon (eso necesitaría una sentencia de asignación: bacon= bacon+ 1).
7. Ambas expresiones se evalúan a la cadena "spamspamspam".
8. Los nombres de las variables no pueden empezar por un número.
9. Las funciones int(), float() y str() se evaluarán a las versiones de número entero, número de coma flotante y cadena del valor que se pase.
10. La expresión provoca un error porque 99 es un número entero, y sólo las cadenas pueden concatenarse a otras cadenas con el operador+ . La forma correcta es He comido '+ str(99)+ ' burritos.'.

Capítulo 2

1. Verdadero y falso, con *T* y *F* mayúsculas, y el resto de la palabra en minúsculas
2. y, o, y no
3. Verdadero y Verdadero es Verdadero. Verdadero y Falso es Falso. Falso y verdadero es falso. Falso y False es False. Verdadero o Verdadero es Verdadero. Verdadero o Falso es Verdadero. Falso o Verdadero es Verdadero. Falso o False es False. no True es False. no Falso es Verdadero.
4. Falso
   Falso
   Verd
   adero
   Falso
   Falso
   Verd
   adero
5. ==, !=, , ,<><= , y .>=
6. == es el operador igual a que compara dos valores y evalúa a un booleano, mientras que= es el operador de asignación que almacena un valor en una variable.
7. Una condición es una expresión utilizada en una sentencia de control de flujo que se evalúa con un valor booleano.
8. Los tres bloques son todo lo que hay dentro de la sentencia if y las líneas print('bacon') y print("ham").

```
print('huevos')
if spam> 5:
```

```
    print('bacon')
else:
    print("jamón")
print("spam")
```

9. El código:

```
si spam== 1:
    print('Hola')
elif spam== 2:
    print('Hola')
else:
    print("¡Saludos!")
```

10. Pulse CTRL-C para detener un programa atascado en un bucle infinito.

11. La sentencia break moverá la ejecución fuera y justo después de un . La sentencia continue moverá la ejecución al inicio del bucle.

12. Todos hacen mismo. La llamada a range(10) va desde 0 hasta (pero sin incluir) 10, range(0, 10) le dice explícitamente al bucle que empiece en 0, y range(0, 10, 1) le dice explícitamente al bucle que incremente la variable en 1 en cada iteración.

13. El código:

```
para i en rango(1, 11):
    print(i)
```

y:

```
i= 1
while i<= 10:
    print(i)
    i= i+ 1
```

14. Esta función se puede llamar con spam.bacon().

Capítulo 3

1. Las funciones reducen la necesidad de duplicar código. Esto hace que los programas sean más cortos, más fáciles de leer y más fáciles de actualizar.
2. El código de una función se ejecuta cuando se llama a la función, no cuando se define la función.
3. La sentencia def define (es decir, crea) una función.
4. Una función está formada por la sentencia def y el código de su cláusula def.

   Una llamada a función es lo que mueve la ejecución del programa a la función, y la llamada a función se evalúa al valor de retorno de la función.
5. Existe un ámbito global y se crea un ámbito local cada vez que se llama a una función.
6. Cuando una función regresa, el ámbito local se destruye y todas las variables que contiene se olvidan.
7. Un valor de retorno es el valor al que se evalúa una llamada a una función. Como cualquier otro valor, un valor de retorno puede utilizarse como parte de una expresión.
8. Si no hay declaración de retorno para una función, su valor de retorno es Ninguno.
9. Una sentencia global obligará a una variable de una función a referirse a la variable global.
10. El tipo de datos de None es NoneType.
11. Esa sentencia import importa un módulo llamado areallyourpetsnamederic. (Esto no es un módulo real de Python, por cierto.)
12. Esta función se puede llamar con spam.bacon().
13. Coloque la línea de código que podría causar un error en una cláusula try.
14. El código que podría causar un error va en la try. El código que se ejecuta si se produce un error va en la cláusula except.

Capítulo 4

1. El valor de lista vacío, que es un valor de lista que no contiene ningún elemento. Esto es similar a cómo `''` es el valor de cadena vacía.
2. `spam[2]= 'hola'` (Observe que el tercer valor de una lista está en el índice `2` porque el primer índice es `0`).
3. `'d'` (Observe que `'3' * 2` es la cadena `'33'`, que se pasa a `int()` antes de ser dividida por `11`. Esto finalmente se evalúa a `3`.) Esto finalmente se evalúa a `3`. Las expresiones pueden utilizarse dondequiera que se utilicen valores).
4. `'d'` (Los índices negativos cuentan desde el final.)
5. `['a', 'b']`
6. `1`
7. `[3.14, 'gato', 11, 'gato', Verdadero, 99]`
8. `[3.14, 11, 'cat', True]`
9. El operador para la concatenación de listas es `+` , mientras que el operador para la replicación es `*`. (Esto es igual que para las cadenas).
10. Mientras que `append()` añadirá valores sólo al final de una lista, `insert()` puede añadirlos en cualquier parte de la lista.
11. La sentencia `del` y el método de lista `remove()` son dos formas de eliminar valores de una lista.
12. Tanto las listas como las cadenas pueden pasarse a `len()`, tener índices y rebanadas, utilizarse en bucles `for`, concatenarse o replicarse, y utilizarse con los operadores `in` y `not in`.
13. Las listas son mutables; se les pueden añadir, eliminar o modificar valores. Las tuplas son inmutables; no pueden modificarse en absoluto. Además, las tuplas se escriben con paréntesis `(` y `)`, mientras que las listas utilizan `[` y `]`.
14. `(42,)` (La coma final es obligatoria.)
15. Las funciones `tuple()` y `list()`, respectivamente
16. Contienen referencias a valores de lista.

17. La función copy.copy() hará una copia superficial de una lista, mientras que la función copy.deepcopy() hará una copia profunda de una lista. Es decir, sólo copy.deepcopy() duplicará cualquier lista dentro de la lista.

Capítulo 5

1. Dos llaves: {}
2. `{'foo': 42}`
3. Los elementos almacenados en un diccionario están desordenados, mientras que los elementos de una lista están ordenados.
4. Aparece un error `KeyError`.
5. No hay ninguna diferencia. El operador `in` comprueba si un valor existe como clave en el diccionario.
6. `'cat' en spam` comprueba si existe una clave `'cat'` en el diccionario, mientras que `'cat' en spam.values()` comprueba si existe un valor `'cat'` para una de las claves de `spam`.
7. `spam.setdefault('color', 'negro')`
8. `pprint.pprint()`

Capítulo 6

1. Los caracteres de escape representan caracteres en valores de cadena que, de otro modo, serían difíciles o imposibles de escribir en código.
2. \n es una nueva línea; \t es un tabulador.
3. El carácter de escape \\ representará un carácter de barra invertida.
4. La comilla simple en Howl's está bien porque has utilizado comillas dobles para marcar el principio y el final de la cadena.
5. Las cadenas multilínea le permiten utilizar nuevas líneas en las cadenas sin el carácter de escape \n.
6. Las expresiones se evalúan de la siguiente manera:
   - 'e'
   - Hola.
   - Hola.
   - ¡Hola mundo!
7. Las expresiones se evalúan de la siguiente manera:
   - HOLA
   - Verdadero
   - Hola.
8. Las expresiones se evalúan de la siguiente manera:
   - ['Recuerda', 'recuerda', 'el', 'quinto', 'de', 'noviembre'].
   - "Sólo puede haber uno".
9. Los métodos de cadena rjust(), ljust() y center(), respectivamente
10. Los métodos lstrip() y rstrip() eliminan los espacios en blanco de los extremos izquierdo y derecho de una cadena, respectivamente.

Capítulo 7

1. La función re.compile() devuelve objetos Regex.
2. Se utilizan cadenas sin procesar para no tener que escapar las barras invertidas.
3. El método search() devuelve objetos Match.
4. El método group() devuelve cadenas del texto coincidente.
5. El grupo 0 es toda la coincidencia, el grupo 1 cubre el primer grupo de paréntesis y el grupo 2 cubre el segundo grupo de paréntesis.
6. Los puntos y paréntesis pueden escaparse con una barra invertida: \.,(, y).
7. Si la expresión regular no tiene grupos, se devuelve una lista de cadenas. Si la expresión regular tiene grupos, se devuelve una lista de tuplas de cadenas.
8. El carácter| significa la correspondencia "o, o" entre dos grupos.
9. El carácter ? puede significar "coincidir con cero o uno de los grupos precedentes" o utilizarse para indicar una coincidencia no libre.
10. El+ coincide con uno o más. El * coincide con cero o más.
11. El {3} coincide exactamente con tres instancias del grupo anterior. El {3,5} coincide con entre tres y cinco casos.
12. Las clases de caracteres abreviados \d, \w y \s corresponden a un solo dígito, palabra o espacio, respectivamente.
13. Las clases de caracteres abreviados \D, \W y \S coinciden con un único carácter que no sea un dígito, una palabra o un espacio, respectivamente.
14. Pasar re.I o re.IGNORECASE segundo argumento a re.compile() hará que las coincidencias no distingan entre mayúsculas y minúsculas.
15. El carácter . normalmente coincide con cualquier carácter excepto con el carácter de nueva línea. Si se pasa re.DOTALL como segundo argumento a re.compile(), el punto también coincidirá con caracteres de nueva línea.
16. El .* realiza una coincidencia codiciosa, y el .*? realiza una coincidencia no codiciosa.
17. O bien [0-9a-z] o bien [a-z0-9].
18. 'X tamborileros, X gaiteros, cinco anillos, X gallinas'

19. El argumento re.VERBOSE permite añadir espacios en blanco y comentarios a la cadena pasada a re.compile().
20. re.compile(r'^ \d{1,3}(,\d{3})*$ ') creará esta expresión regular, pero otras cadenas regulares pueden producir una expresión regular similar.
21. re.compile(r'[A-Z][a-z]*\sNakamoto')
22. re.compile(r'(Alice| Bob| Carol)\s(come| mascotas| lanza)\
s(manzanas|gatos|bolas base)\', re.IGNORECASE)

Capítulo 8

1. Las rutas relativas son relativas al directorio de trabajo actual.
2. Las rutas absolutas empiezan por la carpeta raíz, como */* o *C:³*.
3. La función os.getcwd() devuelve el directorio de trabajo actual. La dirección
   La función os.chdir() *cambia* el directorio de trabajo actual.
4. La carpeta . es la carpeta actual, y .. es la carpeta padre.
5. *C:\bacon\eggs* es el nombre del directorio, mientras que *spam.txt* es el nombre base.
6. La cadena 'r' para el modo lectura, 'w' para el modo escritura y 'a' para el modo append.
7. Un archivo existente abierto en modo de escritura se borra y se sobrescribe completamente.
8. El método read() devuelve todo el contenido del fichero como un único valor de cadena. El método readlines() devuelve una lista de cadenas, donde cada cadena es una línea del contenido del archivo.
9. Un valor de estantería se parece a un valor de diccionario; tiene claves y valores, junto con métodos keys() y values() que funcionan de forma similar a los métodos de diccionario de los mismos nombres.

Capítulo 9

1. La función shutil.copy() copiará un único archivo, mientras que shutil.copytree() copiará una carpeta entera, junto con todo su contenido.
2. La función shutil.move() se utiliza para renombrar archivos, así como para moverlos.
3. Las funciones send2trash moverán un archivo o carpeta a la papelera de reciclaje, mientras que las funciones shutil borrarán permanentemente archivos y carpetas.
4. La función zipfile.ZipFile() es equivalente a la función open(); el primer argumento es el nombre del archivo, y el segundo argumento es el modo en que se abrirá el archivo ZIP (lectura, escritura o anexión).

Capítulo 10

1. assert(spam>= 10, 'La variable spam es inferior a 10.')
2. assert(eggs.lower() != bacon.lower(), '¡Las variables eggs y bacon son iguales!') o assert(eggs.upper() != bacon.upper(), ¡Las variables eggs y bacon son iguales!")
3. assert(False, 'Esta aserción siempre se dispara.')
4. Para poder llamar a logging.debug(), debes tener estas dos líneas al principio de tu programa:

   ```
   importar registro
   logging.basicConfig(level=logging.DEBUG, format=' %(asctime)s -
   %(nombrenivel)s - %(mensaje)s")
   ```

5. Para poder enviar mensajes de registro a un archivo llamado *programLog.txt* con logging.debug(), debes tener estas dos líneas al comienzo de tu programa:

   ```
   importar registro
   >>> logging.basicConfig(filename='programLog.txt", level=logging.DEBUG, format='
   %(asctime)s - %(nombrenivel)s - %(mensaje)s")
   ```

6. DEBUG, INFO, ADVERTENCIA, ERROR y CRÍTICO
7. logging.disable(logging.CRITICAL)
8. Puede desactivar los mensajes de registro sin eliminar las llamadas a la función de registro. Puede desactivar selectivamente los mensajes de registro de nivel inferior. Puede crear mensajes de registro. Los mensajes de registro proporcionan una marca de tiempo.
9. El botón Paso moverá el depurador a una llamada de función. El botón Over ejecutará rápidamente la llamada a la función sin entrar en ella. El botón Out ejecutará rápidamente el resto del código hasta que salga de la función en la que se encuentra.
10. Después de hacer clic en Ir, el depurador se detendrá cuando haya llegado al final del programa o a una línea con un punto de interrupción.
11. Un punto de interrupción es un ajuste en una línea de código que hace que el depurador se detenga cuando la ejecución del programa llega a la línea.
12. Para establecer un punto de interrupción en IDLE, haga clic con el botón derecho en la línea y seleccione **Establecer punto de interrupción** del menú contextual.

Capítulo Xl

1. El módulo webbrowser tiene un método open() que lanzará un navegador web a una URL específica, y eso es todo. El módulo requests puede descargar archivos y páginas de la Web. El módulo BeautifulSoup analiza HTML. Finalmente, el módulo selenium puede lanzar y controlar un navegador.
2. La función requests.get() devuelve un objeto Response, que tiene un valor que contiene el contenido descargado en forma de cadena.
3. El método raise_for_status() lanza una excepción si la descarga tuvo problemas y no hace nada si la descarga tuvo éxito.
4. El atributo status_code del objeto Response contiene el código de estado HTTP.
5. Después de abrir el nuevo fichero en su ordenador en modo 'wb' "write binary", utilice un bucle for que itere sobre el método iter_content() del objeto Response para escribir trozos en el fichero. He aquí un ejemplo:

```
saveFile= open('nombrearchivo.html',
'wb') for chunk in
res.iter_content(100000):
    saveFile.write(chunk)
```

6. F12 abre las herramientas de desarrollo en Chrome. Si pulsas CTRL-Mayúsculas-C (en Windows y Linux) o⌘ -OPCIÓN-C (en OS X), aparecerán las herramientas de desarrollo de Firefox.
7. Haga clic con el botón derecho en el elemento de la página y seleccione **Inspeccionar elemento** en el menú.
8. #main
9. '.highlight'
10. div div
11. 'botón[valor="favorito"]'
12. spam.getText()
13. linkElem.attrs
14. El módulo selenium se importa con from selenium import

webdriver.

15. Los métodos find_element_* devuelven el primer elemento coincidente como objeto WebElement. Los métodos find_elements_* devuelven una lista de todos los elementos coincidentes como objetos WebElement.
16. Los métodos click() y send_keys() simulan los clics del ratón y las teclas del teclado, respectivamente.
17. Llamar al método submit() en cualquier elemento dentro de un formulario lo envía.
18. Los métodos de los objetos WebDriver forward(), back() y refresh() simulan estos botones del navegador.

Capítulo 12

1. La función openpyxl.load_workbook() devuelve un objeto Workbook.
2. El método get_sheet_names() devuelve un objeto Worksheet.
3. Llama a `wb.get_sheet_by_name("Hoja1")`.
4. Llama a wb.get_active_sheet().
5. hoja['C5'].valor or hoja.celda(fila=5, columna=3).valor
6. sheet['C5']= 'Hola' or sheet.cell(row=5, column=3).value= 'Hola'
7. celda.fila y celda.columna
8. Devuelven la columna y la fila con valores más altos de la hoja, respectivamente, como valores enteros.
9. `openpyxl.cell.column_index_from_string("M")`
10. openpyxl.cell.get_columna_letra(14)
11. hoja['A1':'F1']
12. `wb.save("ejemplo.xlsx")`
13. Una fórmula se establece de la misma manera que cualquier valor. Establezca el atributo de valor de la celda en una cadena con el texto de la fórmula. Recuerde que las fórmulas comienzan con el signo =.
14. Cuando llame a load_workbook(), pase True para el argumento de la palabra clave data_only.
15. sheet.row_dimensions[5].height= 100
16. `sheet.column_dimensions["C"].hidden = True`
17. OpenPyXL 2.0.5 no carga paneles de congelación, títulos de impresión, imágenes o gráficos.
18. Los paneles de congelación son filas y columnas que siempre aparecerán en la pantalla. Son útiles para las cabeceras.
19. openpyxl.charts.Reference(), openpyxl.charts.Series(), openpyxl.charts. BarChart(), chartObj.append(seriesObj), y add_chart()

Capítulo 13

1. Un objeto Archivo devuelto por open()
2. Leer-binario ('rb') para PdfFileReader() y escribir-binario ('wb') para PdfFileWriter()
3. Llamar a getPage(4) devolverá un objeto Page para Acerca de este libro, ya que la página 0 es la primera página.
4. La variable numPages almacena un número entero del número de páginas del Objeto PdfFileReader.
5. Llama a decrypt('swordfish').
6. Los métodos rotateClockwise() y rotateCounterClockwise(). Los grados a rotar se pasan como un argumento entero.
7. docx.Document('demo.docx')
8. Un documento contiene varios párrafos. Un párrafo comienza en una nueva línea y contiene varias secuencias. Las tiradas son grupos contiguos de caracteres dentro de un párrafo.
9. Utiliza doc.párrafos.
10. Un objeto Run tiene estas variables (*no* un Párrafo).
11. True hace que el objeto Run siempre esté en negrita y False hace que siempre no esté en negrita, sin importar cuál sea la configuración de negrita del estilo. Ninguno hará que el objeto Ejecutar sólo use la configuración de negrita del estilo.
12. Llama a la función docx.Document().
13. doc.add_paragraph('!')
14. Los números enteros 0, 1, 2, 3 y 4

Capítulo 14

1. En Excel, las hojas de cálculo pueden tener valores de tipos de datos distintos de las cadenas; las celdas pueden tener distintos tipos de letra, tamaños o configuraciones de color; las celdas pueden tener anchuras y alturas variables; las celdas adyacentes se pueden combinar; y se pueden incrustar imágenes y gráficos.
2. Se pasa un objeto `File`, obtenido de una llamada a `open()`.
3. Los objetos de archivo deben abrirse en lectura binaria (`'rb'`) para los objetos `Lector` y escritura binaria (`'wb'`) para los objetos `Escritor`.
4. El método `writerow()`
5. El argumento `delimitador` cambia la cadena utilizada para separar las celdas de una fila. El argumento `lineterminador` cambia la cadena utilizada para separar filas.
6. `json.loads()`
7. `json.dumps()`

Capítulo 15

1. Momento de referencia que utilizan muchos programas de fecha y hora. El momento es el 1 de enero de 1970, UTC.
2. `time.time()`
3. `time.sleep(5)`
4. Devuelve el número entero más cercano al argumento pasado. Por ejemplo, `round(2.4)` devuelve `2`.
5. Un objeto `datetime` representa un momento específico en el tiempo. Un `timedelta` representa una duración de tiempo.
6. `threadObj= threading.Thread(target=spam)`
7. `threadObj.start()`
8. Asegúrese de que el código que se ejecuta en un subproceso no lee ni escribe las mismas variables que el código que se ejecuta en otro subproceso.
9. `subprocess.Popen('c:\\Windows\System32\calc.exe')`

Capítulo 16

1. SMTP e IMAP, respectivamente
2. smtplib.SMTP(), smtpObj.ehlo(), smptObj.starttls(), y smtpObj.login()
3. imapclient.IMAPClient() y imapObj.login()
4. Una lista de cadenas de palabras clave IMAP, como 'BEFORE <fecha>', "FROM <cadena>", o "VISTO".
5. Asigne a la variable imaplib._MAXLINE un valor entero grande, como por ejemplo 10000000.
6. El módulo pyzmail lee los correos electrónicos descargados.
7. Necesitará el número SID de la cuenta Twilio, el número del token de autenticación y su número de teléfono Twilio.

Capítulo 17

1. Un valor RGBA es una tupla de 4 enteros, cada uno entre 0 y 255. Los cuatro enteros corresponden a la cantidad de rojo, verde, azul y alfa (transparencia) del color. Los cuatro enteros corresponden a la cantidad de rojo, verde, azul y alfa (transparencia) del color.
2. Una llamada a la función `ImageColor.getcolor('CornflowerBlue', 'RGBA')` devolverá `(100, 149, 237, 255)`, el valor RGBA para ese color.
3. Una tupla de caja es un valor de tupla de cuatro enteros: la coordenada x del borde izquierdo, la coordenada y del borde superior, la anchura y la altura, respectivamente.
4. `Image.open('zophie.png')`
5. `imageObj.size` es una tupla de dos enteros, la anchura y la altura.
6. `imageObj.crop((0, 50, 50, 50))`. Observe que está pasando una tupla de cajas a `crop()`, no cuatro argumentos enteros separados.
7. Llama al método `imageObj.save('nuevo_nombre_archivo.png')` de la `Imagen` objeto.
8. El módulo `ImageDraw` contiene código para dibujar sobre imágenes.
9. Los objetos ImageDraw tienen métodos para dibujar formas como `punto()`, `línea()`, o `rectángulo()`. Se devuelven pasando el objeto `Image` a la función `ImageDraw.Draw()`.

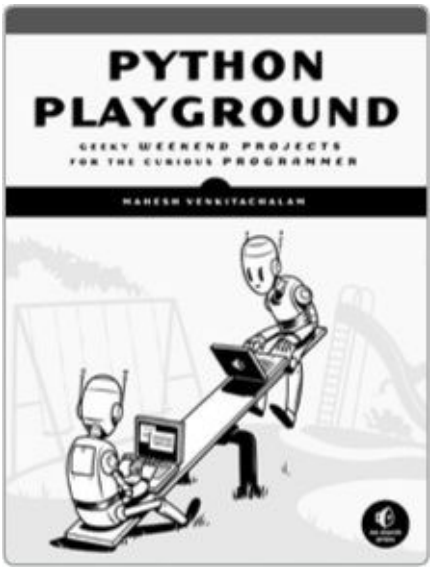
Capítulo 18

1. Mueve el ratón a la esquina superior izquierda de la pantalla, es decir, a las coordenadas (0, 0).
2. pyautogui.size() devuelve una tupla con dos enteros para la anchura y la altura de la pantalla.
3. pyautogui.position() devuelve una tupla con dos enteros para las coordenadas x-y del cursor del ratón.
4. La función moveTo() mueve el ratón a coordenadas absolutas en la pantalla, mientras que la función moveRel() mueve el ratón en relación a la posición actual del ratón.
5. pyautogui.dragTo() y pyautogui.dragRel()
6. pyautogui.typewrite('¡Hola mundo!')
7. Puede pasar una lista de cadenas de teclas de teclado a pyautogui.typewrite() (como 'izquierda') o pasar una única cadena de teclas de teclado a pyautogui.press().
8. pyautogui.screenshot('screenshot.png')
9. pyautogui.PAUSE= 2

Apéndice D. Recursos

Visite *http://nostarch.com/automatestuff/* para obtener recursos, erratas y más información.

Más libros sensatos de **NO STARCH PRESS**



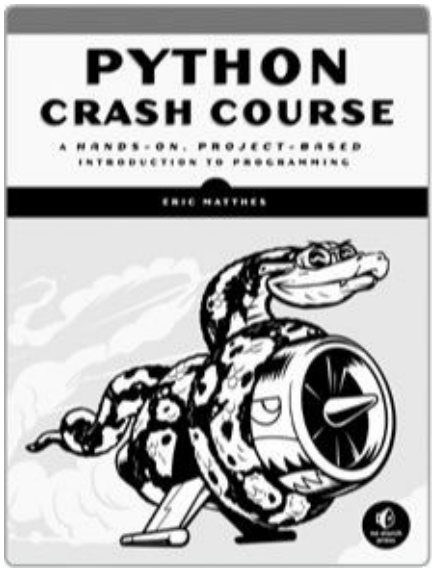
PATIO DE RECREO PYTHON

Proyectos frikis de fin de semana para programadores curiosos

por MAHESH VENKITACHALAM

OCTUBRE 2015, 352 PP., $29.95

ISBN 978-1-59327-604-1



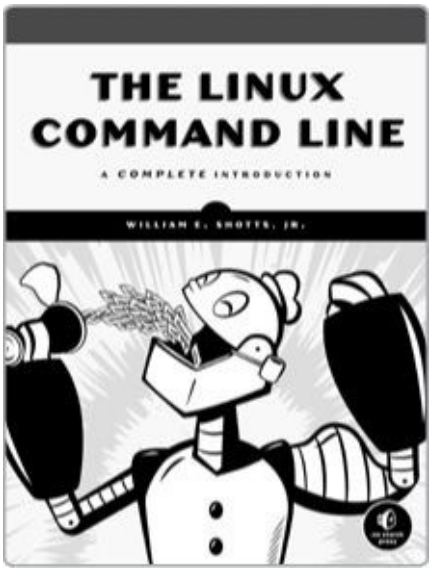
CURSO INTENSIVO DE PYTHON

Introducción práctica a la programación basada en proyectos

por ERIC MATTHES

NOVIEMBRE DE 2015, 560 PÁGS., 39,95 €.

ISBN 978-1-59327-603-4



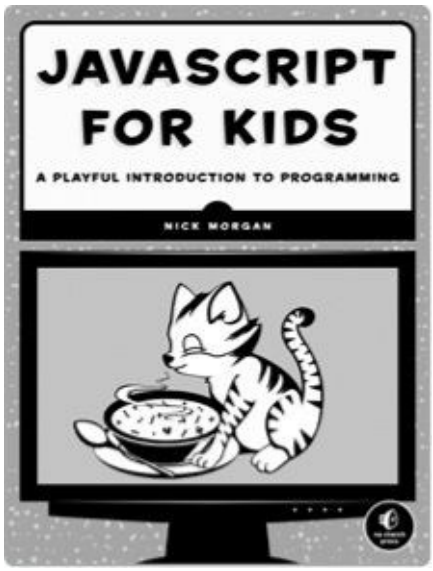
LA LÍNEA DE COMANDOS DE LINUX

Una introducción completa

por WILLIAM E. SHOTTS, JR. ENERO

2012, 480 PP., $39.95

ISBN 978-1-59327-389-7



JAVASCRIPT PARA NIÑOS

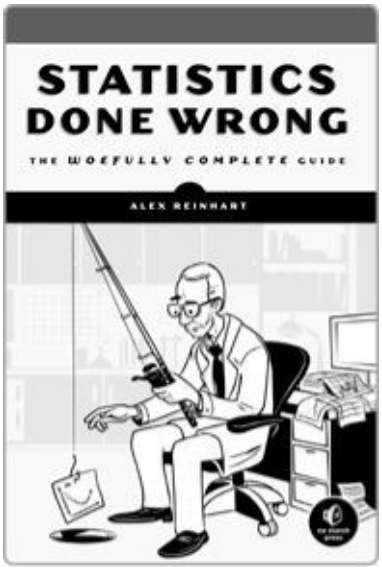
Introducción lúdica a la programación

por NICK MORGAN

DICIEMBRE DE 2014, 336 PÁGS., 34,95 $.

ISBN 978-1-59327-408-5

a todo color



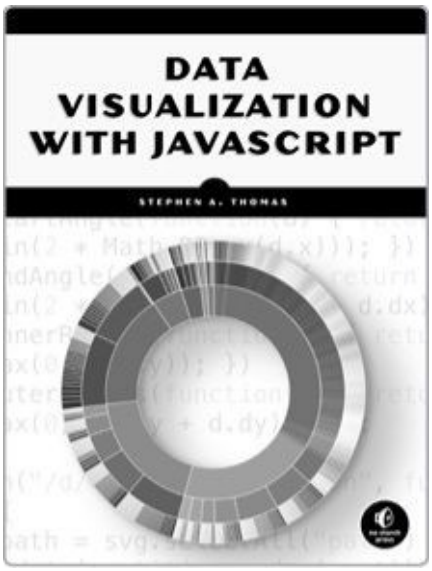
ESTADÍSTICAS MAL HECHAS

La guía lamentablemente completa

por ALEX REINHART

MARZO DE 2015, 176 PÁGS., 24,95 $.

ISBN 978-1-59327-620-1



VISUALIZACIÓN DE DATOS CON JAVASCRIPT

por STEPHEN A. THOMAS MARZO

2015, 384 PP., $39.95

ISBN 978-1-59327-605-8

a todo color

TELÉFONO:

800.420.7240 O 415.863.9900

CORREO ELECTRÓNICO:

SALES@NOSTARCH.COM **WEB:**

WWW.NOSTARCH.COM

Índice

NOTA SOBRE EL ÍNDICE DIGITAL

Un enlace en una entrada del índice se muestra como el título de la sección en la que aparece esa entrada. Como algunas secciones tienen varios marcadores de índice, no es raro que una entrada tenga varios enlaces a misma sección. Al hacer clic en cualquier enlace, accederá directamente al lugar del texto en el que aparece el marcador.

Símbolos

Operador = (asignación), Concatenación y replicación de cadenas, Operadores de comparación

$ (signo de dólar), Clases de caracteres, Coincidencia de nuevas líneas con el carácter de punto

. (carácter de punto), de intercalación y del signo de dólar, Los caracteres de intercalación y del signo de dólar, Los caracteres El directorio de trabajo actual

utilizando en rutas, El directorio de trabajo actual

Coincidencias con comodines, Los caracteres Caret y Dollar Sign

"(comillas dobles), Literales de cadena (exponente), Introducción de expresiones en la (igual a), consola interactiva== en la consola interactiva, Valores , booleanosOperadores de comparación/ (barra oblicua), Introducción de expresiones El truco de la asignación múltiple, Archivos y rutas de archivos**

operador de división, Introducción de expresiones en el intérprete de comandos interactivo, El truco de la asignación múltiple

\ (barra invertida), Programa de ejemplo: Bola 8 Mágica con una Lista, Literales de Cadena, Creación de objetos Regex, comparación de nuevas líneas con el carácter punto, archivos y rutas de archivos

carácter de continuación de línea, Programa de ejemplo: Bola 8 Mágica con una Lista

> (mayor que), Valores booleanos>= (mayor o igual que), Valores booleanos# (carácter hash), Cadenas multilínea con comillas triples// (división entera/cociente alícuota), Introducción de expresiones en el intérprete de comandos interactivo< (menor que), Valores booleanos<= (menor o igual que), Valores booleanos% (módulo/suma),

¡Introducción de expresiones en el intérprete de comandos interactivointérprete de comandos interactivo, , El truco de la asignación El truco de la asignación múltiple!= múltiple* (operador de multiplicación), Introducción de expresiones en el Obtención de la longitud de una lista con len(), (no igual a), Valores booleanos() (paréntesis), Tipos de datos mutables e inmutables, Revisión de la concordancia de expresiones regulares| (carácter de tubo), Agrupación con paréntesis, Gestión de expresiones regulares complejas+ (signo), másIntroducción de expresiones en la pantalla interactiva, enteros, de ante Tipos de datos coma floty de cadena, Obtención de la longitud de una lista con len(), El truco de la asignación múltiple, Concordancia opcional con el interrogaciónsigno de,
Coincidencia de nuevas líneas con el carácter de punto

operador de adición, Introducción de expresiones en el intérprete de comandos interactivo, Tipos de datos enteros, de coma flotante y de cadena, Obtención de la de una lista longitudcon len(), El truco de la asignación múltiple

? (signo de interrogación), Hacer coincidir varios grupos con la tubería, Hacer coincidir Newlines with the Dot Character' (comilla simple), String Literals[] (corchetes), The List Data Type, Matching Newlines with the Dot Character* (asterisco), Optional Matching with the Question Mark, The Wildcard Character, Matching Newlines with the Dot Character

utilizando con carácter comodín, El carácter comodín

cero o más coincidencias con, Coincidencia opcional con el signo de interrogación
- (resta), Introducción de expresiones en el intérprete de comandos interactivo, El truco de la asignación múltiple^ (símbolo de intercalación), Clases de caracteres, Clases de caracteres, Coincidencia de nuevas líneas con el carácter de punto

Coincidencia con el principio de la cadena, Clases de caracteres

clases de caracteres Clases de caracteresnegativos,
''' (comillas triples), Caracteres de escape, Gestión de Regexes complejas (guión bajo), Nombres de variables: (dos puntos), Bloques de código, bucle whileSentencias de, Bucles for y la función range(), Índices negativos, Indexación y troceado de cadenas{} (llaves), Diccionarios y estructuración de datos, Coincidencia de uno o más con el signo más, Coincidencia de uno o más con el signo más,
Coincidencia de nuevas líneas con el carácter de punto

Coincidencia Coincidencia codiciosa frente a no codiciosa, de una o más con el

más Coincidencia Coincidencia de una o más con el más de repeticiones

específicas con, A

Directiva %A, Pausa hasta una fecha determinada

Directiva %a, Pausa hasta una fecha específica rutas

absolutas, El directorio de trabajo actual función

abspath(), El módulo os.path

operador de adición (+), Introducción de expresiones en la consola interactiva, Tipos de datos enteros, de coma flotante y de cadena, Obtención de la longitud de una lista con len(), El truco de la asignación múltiple

modelo aditivo de color, Colores y valores RGBA método

add_heading(), Escribir documentos Word método addPage(),

Crear PDFs

método add_paragraph(), Escribir documentos Word método

add_picture(), Añadir encabezados

método add_run(), Escritura de documentos Word

notación algebraica ajedrez, Impresión bonita

atributo all_caps, Atributos de ejecución

Tecla de búsqueda ALL, Selección de una carpeta

alfa, definido, Fundamentos de la imagen por

ordenador y operador, Operadores de comparación

clave de búsqueda ANSWERED, Realización de la

búsqueda

API (interfaz de programación de aplicaciones), Paso 3: Escribir el archivo CSV sin la primera fila

método append(), Métodos

contraseñas específicas de la aplicación, Inicio de sesión en el servidor SMTP

palabra clave args, Pasando Argumentos a la Función Destino del Thread

argumentos, función, Comentarios, sentencias def con parámetros, retorno valores de y sentencias return, multihilo, lanzamiento otros de programas desde Python

argumentos de palabras clave, valores de retorno y declaraciones de

retorno pasadas a procesos, lanzamiento de otros programas desde

Python

pasar a hilos, Multithreading
aserciones, aserciones, ejecución de programas Python en OS X y Linux

desactivación, Ejecución de programas Python en OS X y Linux operador de asignación (=), Concatenación y replicación de cadenas, Operadores de comparaciónAT&T mail, Conexión a un servidor SMTP,
Recuperación y eliminación de mensajes de correo electrónico con IMAPattributes, HTML, A Quick
Repaso, Obtención de datos de los de atributos de un elementooperadores de asignación , El truco de la asignación múltiple

B

\b carácter de escape de retroceso, Paso 3: Obtener e imprimir las coordenadas del ratón

Directiva %B, Pausa hasta una fecha específica

directiva %b, Pausa hasta una fecha específica

método back(), Envío de claves especiales

Barra invertida (\), Literales de cadena, Creación de objetos Regex, Coincidencia de nuevas líneas con el carácter punto, Archivos y rutas de archivos

Función BarChart(), Gráficos

Función basename(), Manejo de rutas absolutas y relativas Tecla de

búsqueda BCC, Realización de la búsqueda

Beautiful Soup, análisis sintáctico de HTML con el módulo

BeautifulSoup (véase también el módulo bs4)

Objetos BeautifulSoup, Análisis de HTML con el módulo BeautifulSoupTecla de búsqueda BEFORE, Selección de una carpetaArchivos binarios, Búsqueda de archivos tamaños de y contenidos de carpetas, Escritura en archivosOperadores binarios, Operadores de comparaciónOperador bitwise or, Gestión de expresiones regulares complejasCadenas en blanco, Tipos de datos de enteros, coma flotante y cadenasBloqueo de la ejecución, Función time.time() Functionbloques de código, Mezcla de operadores booleanos y de comparaciónTecla de búsqueda BODY, Selección de una carpetaAtributo negrita, Atributos de ejecuciónTipo de datos booleano, Control de flujo, Operadores de comparación, binariosOperadores booleanos , Sentencias continue, Los operadores in y not in, Los operadores in y not in

operadores binarios, Operadores de

comparación control de flujo y, Control de

flujo

operador in, Los operadores in y no in Los

operadores in y no inoperador not in,

valores verdadero" y "falso", continúe Declaraciones

uso conjunto de operadores binarios y de comparación, Operadores booleanos binarios tuplas de caja, Coordenadas y tuplas de caja puntos de rupturaDepuración de un programa de suma de , depuración mediante, números sentencias de ruptura, Un molesto bucle while, bucles for y la función range()

resumen, Un molesto bucle while

utilizando en bucle for, bucles for y la función range()

navegador, apertura mediante el módulo webbrowser, módulo Web Scrapingbs4, Parseo de el módulo BeautifulSoup, Parseo de HTML con el módulo BeautifulSoup, HTML con Creación de un objeto BeautifulSoup a partir de HTML, Obtener datos de los atributos de un elemento

Creación de objetos a partir de HTML, Análisis de HTML con el módulo BeautifulSoup

encontrar un elemento con el método select(), Crear un objeto BeautifulSoup a partir de HTML

obtener atributo, Obtener datos de los atributos de un elemento

Visión general, Análisis de HTML con el módulo BeautifulSoup
Funciones incorporadas, Importar viñetasMódulosLista de, Crear en el marcado , WikiProyecto: Añadir viñetas a Wiki Markup, Proyecto: Añadir viñetas a Wiki Markup, Proyecto: Añadir viñetas a Wiki Markup, Paso 1: Copiar y pegar desde el portapapeles, Paso 3: Unir las líneas modificadas

copiar y pegar portapapeles, Proyecto: Añadir viñetas a Wiki Markup uniendo las líneas modificadas Paso 3: Unir las líneas modificadas,

visión general, Proyecto: Añadir viñetas al marcado wiki

separar líneas de texto, Paso 1: Copiar y pegar desde el portapapeles

C

funciones de llamada, Comentarios

pila de llamadas, definida, Raising Exceptions

camelcase, Nombres de variables

Símbolo de intercalación (^), Clases de caracteres, Clases de caracteres, Coincidencia de nuevas líneas con el carácter de punto

Coincidencia con el principio de la cadena, Clases de caracteres

clases de caracteres Clases de caracteresnegativos,
Hojas de estilo en cascada (CSS), Creación de un objeto BeautifulSoup a partir de HTML, Búsqueda de elementos en la página

emparejamiento con el módulo selenium, Búsqueda de elementos en la página

selectores, Creación de un objeto BeautifulSoup a partir de HTML
Distinción entre mayúsculas y minúsculas, Nombres de variables, Coincidencia entre mayúsculas y CCminúsculasTecla de búsqueda , Realización de la celdabúsquedaObjetos de, Obtención de hojas de las celdas del libro de trabajo, en hojas de cálculo Excel, Trabajo con hojas de cálculo Excel, Obtención de hojas del libro de trabajo, Creación y eliminación de hojas, Configuración fila del alto dey el ancho de columna

Accediendo al objeto Celda por su nombre, Obteniendo Hojas del Libro de Trabajo

fusionando y desagrupando, Configurando el Alto de Fila y el Ancho de Columna

escribir valores en, crear y eliminar hojas
método center(), Los métodos de cadena join() y split(), Reconocimiento de imágenesllamadas a métodos de encadenamiento, Girar y voltear imágenesclases , de , Coincidencia de nuevas líneas con el carácter de puntoestilos de caracteres, caracteresEl método findall()Estilización de gráficos de párrafo y de ejecuciónobjetos, Excel, Congelar chdirpanelesfunción(), El trabajo directorio de actualChrome, herramientas de desarrollador en, Ver el fuente HTML de una página webmétodo clear(), Encontrar elementos en la páginafunción click(), clic con Hacerel ratón, Revisión de las funciones de PyAutoGUI, Proyecto: Automatic Form Fillerclicking mouse, Haciendo clic en el método Mouseclick(), Encontrando Elementos en el Pageclipboard, usando string from, Paso 3: Manejar el del Portapapeles Contenido e Iniciar el NavegadorCMYK color model, Colores y Valores RGBAcolon (:), Bloques de Código, Declaraciones de Bucle while, Bucles for y la Función range(), Índices Negativos, Indexando y Rebanando Cadenascolor values, Fundamentos de la Imagen por Computadora, Colores y Valores RGBA

Modelos de color CMYK frente a RGB, Colores y valores RGBA

Valores RGBA, Fundamentos de imagen por ordenador función column_index_from_string(), Conversión columna entre letras de letras de y números, y númeroscolumnas, en hojas de cálculo Excel, Conversión entre columna Fórmulas

ajuste de la altura y la anchura de, Fórmulas

Rebanar objetos de la hoja de cálculo para obtener objetos de celda, Conversión entre letras y números de columna
Comcast mail, Conexión a un servidor SMTP, Recuperación y eliminación Correo electrónico con IMAPelementos delimitados por comas, El tipo de datos de la comandoslistaargumentos de la línea de , Paso 1: Descubra la función URLcommentAfterDelay(), Pulsar y soltar el tecladocomentarios, Comentarios, Cadenas de varias líneas con comillas triples

multilínea, cadenas multilínea con comillas triples

visión general, Comentarios
operadores de comparación, valores valores booleanosbooleanos, resumen de

operadores booleanos binarios,

uso de operadores binarios con, Operadores booleanos binarios

función compile(), Creación de Regexobjetos, Revisión de la expresiones regularesconcordancia de , Gestión de Regexescomprimidos complejos, Recorrido por un árbol de directorios, Compresión de archivos con el módulo zipfileExtracción , archivos de ZIP, Extracción de archivos ZIP, Paso 3: Formar el nuevo nombre de archivo y renombrar los archivos

carpeta de copia de seguridad en, Paso 3: Formar el nuevo nombre de archivo y renombrar los archivos

creación de archivos ZIP, Extracción de archivos

ZIP extracción de archivos ZIP, Extracción de

archivos ZIP descripción general, Recorrer un árbol

de directorios

lectura de archivos ZIP, Compresión de archivos con el módulo zipfile pantalla del ordenador, Pausas y FailSafes, Control del movimiento del ratón

coordenadas de, pausas y FailSafes

resolución de, Control del movimiento del ratón

concatenación, Tipos de datos enteros, de coma flotante y de cadena, Obtención de la longitud de una lista con len()

de listas, Obtención de la longitud de una lista con len()

string, Los tipos de datos Integer, Floating-Point, y String problemas de concurrencia, Pasando argumentos a la destino del función de threadcondiciones, definidas, Mezclando booleanos y de operadores comparacióncontinue statements, continue Statements, for Loops and the range() Función

resumen, continuar Declaraciones

utilizando en bucle for, bucles for y la función range()

Tiempo Universal Coordinado (UTC), La hora Modulecoordinates, Colores y valores RGBA, Pausas y FailSafes

de la pantalla del ordenador, Pausas y FailSafes

de una imagen, Colores y Valores RGBA

función copy(), Pasar referencias, Eliminar espacios en blanco con strip(), rstrip() y lstrip(), Organizar archivos, Copiar y pegar imágenes en

Otras ImágenesFuncióncopytree(), Organizar Cuenta AtrásArchivosProyecto , Proyecto: Programa Cuenta Atrás Simple, Proyecto: Simple Countdown Program, Proyecto: Simple Countdown Program, Proyecto: SimplePrograma Cuenta Atrás

cuenta atrás, Project: Simple Countdown Program descripción

general, Project: Programa Cuenta Atrás Simple

reproduciendo archivo de sonido, Proyecto: Programa simple de cuenta atrás Función cProfile.run(), La función time.time()crashes, programa, Introducción de expresiones en el Interactive Shellcreate_sheetmétodo (), Creación y eliminación de hojasNivelCRÍTICO, Niveles de registrocron, Lanzamiento de otros programas desde PythonCaptura de imágenes, Trabajo con el tipo de datos de imagenCSS (Cascading Style Sheets), Creación de un objeto BeautifulSoup a partir de HTML, Búsqueda de elementos en la página

emparejamiento con el módulo selenium, Búsqueda de elementos en la página

selectores, Creación de un objeto BeautifulSoup a partir de HTML
Archivos CSV, Trabajando con Archivos CSV y Datos JSON, Archivos y Datos JSON, Trabajando con CSV Objetos Lectores, Leyendo Datos de Lectores Objetosen un Bucle for, Leyendo Datos de Objetos Lectores en un Bucle for, Los Argumentos , , Los Argumentos , de , de delimiter lineterminatordelimiter y Palabra Clave delimiter lineterminatorArgumentos Palabra Clave Los Argumentos de Palabra Clave de Palabra Clave delimiter y lineterminatory Los y lineterminatorProyecto: Eliminación del encabezado de los archivos CSV, Proyecto: Eliminación de la cabecera de los archivos CSV, Paso 2: Lectura del archivo CSV

definida, trabajar con archivos CSV y datos JSON

delimiter for, El delimitador y el lineterminador Argumentos de palabras clave

resumen del formato, Trabajar con archivos CSV y datos JSON

terminador de línea para, El delimitador y el lineterminador Palabra clave

Argumentos Objetos de lectura, Objetos de lectura

lectura de datos en bucle, Lectura de datos de objetos de lectura en un bucle for

eliminar cabecera de, ,Los argumentos de palabra clave delimiter y lineterminator Los de palabra clave delimiter y lineterminatorargumentos,

Proyecto: Eliminación del encabezado de los archivos CSV, Proyecto: Eliminación del encabezado de los archivos CSV, Paso 2: Lectura del archivo CSV

bucle a través de archivos CSV, Proyecto: Eliminación del encabezado de los archivos CSV, Argumentos de las palabras clave delimiter y lineterminator

lectura en archivo CSV, Proyecto: Eliminación de la cabecera de los archivos CSV

escribir el archivo CSV, Paso 2: Leer el archivo CSV
Objetos de escritura, Lectura de datos de objetos de lectura en un bucle for
Paréntesis({})rizados , Coincidencia de uno Diccionarios y estructuración de datos, , o más con el signo másCoincidencia de uno o más con el signo más, Coincidencia de nuevas líneas con el carácter punto

Emparejamiento codicioso frente a no codicioso, Emparejamiento de uno o más con el Plus

Coincidencia de repeticiones específicas con, Coincidencia de una o más con el Plus
directorio de El directorio de trabajo actualtrabajo actual,

D

\D clase de caracteres, El método findall()

\d clase de caracteres, El método findall()

Directiva %d, Pausa hasta una fecha específica

estructuras de datos, Pretty Printing, Uso de las estructuras de datos para modelar la realidad. Cosas del mundo

notación algebraica de ajedrez, Pretty Printing

tablero de tres en raya, Uso de las estructuras de datos para modelar los del mundo realtipos de datos , Introducción de expresiones en el intérprete de comandos interactivo, de datos entero, y de entero, y de Los tipos , Los tipos de datos cadena, de coma flotante cadenade coma flotante Los tipos de datos entero, de coma flotante y de cadena,
Control de flujo, Valores de retorno y sentencias return, El tipo de datos lista, Tipos tipo lista: Cadenas y tuplas, Tipos de datos mutables e inmutables, El tipo de datos tupla, , Diccionarios y estructuración de datosEl tipo de datos tupla

Booleanos, control de flujo

definidos, Introducción de expresiones en los la shell

interactivadiccionarios de, Diccionarios y estructuración de datos

números en coma flotante, Tipos de datos , de coma flotante y de cadenas Tipos de

datos enteros, de coma flotante y de cadenasenterosnúmeros enteros,

función list(), El tipo de datos tupla listas,

El tipo de datos lista

mutable vs. immutable, Tipos tipo lista: Cadenas y tuplas Valor nulo,

Valores de retorno y sentencias return

strings, Tipos de datos enteros, de coma flotante y de cadenas función

tuple(), Tipo de datos tupla

tuplas, tipos de datos mutables e inmutables

módulo datetime, El El El módulo datetime, módulo , El módulo datetime, El módulo datetime, datetimetipo de datos timedelta, Pausa hasta una fecha específicaPausa hasta una , fecha , específicaConversión de objetos datetime en cadenas, Revisión de las de tiempo de Pythonfunciones

aritmética utilizando, El tipo de datos timedelta

conversión de objetos en cadenas, Pausa hasta una fecha determinada

conversión de cadenas a objetos, Conversión de objetos datetime en cadenas función

fromtimestamp(), El módulo datetime

Función now(), Módulo datetime

visión general, El módulo datetime, Revisión de las funciones de tiempo de Python

pausar el programa hasta la hora, Pausar hasta una fecha específica

tipo de datos timedelta, El módulo datetime

método total_seconds(), El módulo datetime

Objetos datetime, El módulo datetime, Pausar hasta una fecha específica,

Conversión de objetos datetime en cadenas conversión a

cadenas, Pausa hasta una fecha específica

conversión de cadenas a, Conversión de objetos datetime en cadenas función debug(), Uso del módulo depuraciónlogging Uso del módulo logging, , ¿Qué es Python?, Depuración, Levantamiento de excepciones, Aserciones, Uso de una aserción en una simulación de semáforo, Uso del módulo logging, Niveles de registro, Desactivación del registro, , Desactivación del registroCambio, Depuración de un programa de suma de números

aserciones, Definición de

aserciones, ¿Qué es Python?

Obtención del rastreo como cadena de caracteres, Salto de excepciones

in IDLE, Desactivación del registroDesactivación del registro, Over, Depuración de

un número Añadir programadescripción general del ,

paso a través del programa, Over

Uso de puntos de interrupción, Depuración de un programa de suma de números logging, Uso de una aserción en una simulación de semáforo, Uso del loggingmódulo , Uso del módulo logging, Niveles de logging, Desactivación del logging

deshabilitar, Niveles de

registro en archivo,

Deshabilitar registro

niveles de, Uso del módulo de registro

módulo loggingUso del módulo logging, Uso de una aserción en una de

simulación de semáforo función print() y,
Cómo generar excepciones, Nivel de , Utilización del módulo de registroNúmeros decimales, depuraciónDEBUGFunciones str(), int() y float() (consulte Números coma flot)Método decode(), Obtención de direcciones de correo electrónico de un mensaje sin cifrarencriptación de archivos PDF, de anteExtracción de texto de archivos PDFcómo duplicar código, Funcionesfunciónndeepcopy(), Paso de ,referenciasdeclaracionesdef
Funciones, def Sentencias con Parámetros

con parámetros, def Declaraciones con parámetros

Tecla de búsqueda DELETED, Ejecución del Searchdelete_messagesmétodo (), Obtención del cuerpo de un mensaje sin procesareliminación de archivos/carpetas, Movimiento y cambio de nombre de archivos y carpetas, Eliminación permanente de archivos y carpetas

permanentemente, mover y renombrar archivos y carpetas

usando el módulo send2trash, Borrado permanente de archivos y carpetas sentencias del, Eliminación de valores de listas con sentencias deldiccionarios, Paso referencias, de Paso de referencias, Diccionarios y estructuración de datos, El tipo de datos diccionario, Diccionarios vs. Listas, Diccionarios vs. Listas, Diccionarios vs. Listas, Comprobación de la una existencia de clave o un valor en un diccionarioComprobación una clave o un valor , de la existencia de en un diccionarioComprobación de la existencia de una clave o un valor en un , diccionario, El método setdefault(), Un tablero de tres en raya

función copy(), pasar referencias pasar

referenciasfunción deepcopy(),

método get(), Comprobación la existencia de valor en un de una clave o un

diccionario Comprobación de la existencia de una clave o un valor en un

diccionario en operador, método items(), Diccionarios frente a listas

método keys(), Diccionarios vs. Listas

listas vs., El del tipo de datos

diccionarioanidamiento , Un tablero de

tres en raya

operador not in, Comprobación la existencia de una clave o un valor en un

diccionario, Diccionarios y estructuración de datos

método setdefaultEl método setdefault(),

método values(), Diccionarios vs. Listas
directorios, Lectura y escritura de archivos, Archivos y rutas de archivos, El actualdirectorio de trabajo , El directorio de trabajo actualabsolutas vs. , Rutas relativas, El módulo os.path, El módulo os.path, El módulo os.path, Manejo de rutas absolutas y relativas, Manejo de absolutas y relativasrutas , Búsqueda de tamaños de archivos y contenidos de carpetas, Organización de archivos, de Copiaarchivos y carpetas, Copia de archivos y carpetas, Movimiento y cambio de nombre de archivos

y carpetas, Borrado permanente de archivos y carpetas, Borrado seguro con el módulo send2trash

rutas absolutas vs. relativas, El directorio de trabajo actual barra invertida

vs. barra diagonal, Archivos y rutas de archivos

copiar, Organizar archivos

creación, trayectorias absolutas frente a trayectorias relativas

directorio de trabajo actualdel directorio de trabajo actual, Definición ,

Lectura y escritura de archivos

eliminar permanentemente, mover y renombrar archivos y carpetas

borrando usando el módulo send2trash, borrando permanentemente archivos y

carpetas en movimiento, copiando archivos y carpetas

Módulo os.pathEl . El os.path, , módulo path,módulo , El módulo ospath.osManejo de rutas absolutas y Manejo de absolutas , rutas y relativas, Búsqueda de tamaños de archivos y contenidos de carpetas

rutas absolutas en, El módulo os.path

tamaño de los archivos, manejo de rutas absolutas y relativas

contenido de carpetas, Manejo de rutas absolutas y relativas, El

módulo os.path

validez de la ruta, búsqueda de tamaños de archivos y contenidos de carpetas

rutas relativas en, El módulo os.path
renombrar, Copiar archivos y carpetaswalking, Borrados seguros con la función send2trash Moduledirname(), Manejo de absolutas y rutas relativasfunción disable(), Niveles de registrooperador division (/), Introducir expresiones en el intérprete de comandos interactivo, El las asignaciones truco de múltiplesObjetos de documento, Documentos de Wordsigno de dólar ($), Clases de caracteres, Coincidencia de nuevas líneas con el carácter de de punto puntocarácter(.)y , Los caracteres de signo de de dólarintercalación , Los caracteres de signo de intercalación y de dólar, El directorio de trabajo actual

utilizando en rutas, El directorio de trabajo actual

Coincidencias con comodines, Los caracteres Caret y Dollar Sign
carácter punto-estrella (.*), El carácter doubleClickcomodínfunción(), Hacer clic con el ratón, Revisión de las funciones de dobles PyAutoGUIcomillas("), Literales de double strikecadenaatributo, Atributos de ejecucióndescarga, Descarga de archivos de la Web con el módulo requests, archivos descargados en el disco duro, Paso 3: Abrir navegadores Web Guardar para cada resultado, Proyecto: Multithreaded XKCD Downloader

archivos desde la web, Guardar archivos descargados en el disco duro

páginas web, Descarga de archivos de la Web con el módulo de solicitudes

Cómics XKCD, Paso 3: Abrir navegadores web para cada resultado, Proyecto: Descargador XKCD Multihilo
Tecla de búsqueda DRAFT, arrastrando el ratón, Pulsando la teclaRealizando la búsqueda
Función MousedragRel(), Hacer clic con el ratón, Arrastrar el ratón, Repaso Hacer clic con el ratón, Repaso de las PyAutoGUIFuncióndragTode las funciones
PyAutoGUIdibujar sobre imágenes, Ideas para funciones (), programas similaresIdeas para programas similares, , Ideas para similaresprogramas , Líneas, Líneas, Líneas, Líneas, Líneas, Dibujo Ejemplo

elipses, Líneas

programa de ejemplo, Líneas

Módulo ImageDraw, Ideas para similares Ideas para

programas similareslíneas de programas ,

puntos, Ideas para programas similares

polígonos, Líneas

rectángulos, Líneas

texto, Ejemplo de dibujo
Función Lectura de JSON con la función loads() duracióndumps(), Argumentos de la palabra clave , Control del movimiento del ratón

E

ehlo(), Conexión a un servidor SMTP, Paso 3: Enviar recordatorios personalizados por correo electrónico

elementos, HTML, Guardar archivos descargados en el disco duro

sentencias elif, sentencias else

método ellipse(), Líneas

declaraciones else, declaraciones if

direcciones de correo electrónico, extraer, , Combinar re.IGNORECASE, re.DOTALLy Combinar re.IGNORECASE, re.DOTALL, y re.VERBOSE, re.VERBOSE, Proyecto: de Números de Teléfono y Direcciones de Correo ElectrónicoExtractor, Paso 2: Crear una Regex para Direcciones de Correo Electrónico, Paso 3: Buscar Todas las Coincidencias en el Texto del Portapapeles

crear regex, Proyecto: Extractor de números de teléfono y direcciones de correo electrónico

encontrar coincidencias en el portapapeles, Paso 2: Crear una Regex para direcciones de correo electrónico

unir coincidencias en una cadena, Paso 3: Buscar todas las coincidencias en el texto del portapapeles

resumen, Combinación de re.IGNORECASE, re.DOTALL y re.VERBOSEcorreos electrónicos , SMTP, , SMTPConexión a un servidor SMTP, servidor SMTPConexión a un , Conexión a un SMTPservidor , Inicio de sesión en el servidor SMTP, Desconexión del servidor SMTP Desconectarse SMTPDesconectarse del del servidor IMAPservidor , del servidor SMTP, Conectarse a un servidor IMAPConectarse a un , , Conectarse a un servidor IMAP, servidor IMAPLímites de tamaño, Límites de tamañotamaño, Límites de , Obtener un correo electrónico y marcarlo como leído, Obtener el cuerpo de un mensaje sin procesar Obtener el cuerpo de un mensaje sin procesar, Desconectarse ,

borrar, Obtener el cuerpo de un mensaje sin procesar

Desconectarse del servidor, Obtener el cuerpo de un mensaje sin , Desconectarse del servidor SMTP, Conectarse a un IMAP Conexión a un servidor IMAP, Límites de tamaño, Obtención de un correo electrónico y

Marcar como leído

carpetas, Conexión a un servidor IMAP

obtención del contenido de los mensajes,

Límites de tamaño

Inicio de sesión en el servidor, Descripción general de la conexión

a un servidor IMAP, Desconexión del servidor SMTP

mensajes sin procesar, Obtener un correo electrónico y marcarlo como leído método gmail_search(), Límites de tamañoIMAP, Desconexión del servidor SMTPmarcar un mensaje como leído, Límites de tamañobuscar, Conectarse a un Servidores IMAPEnviar, SMTP, Conectarse a un servidor SMTP, Conectarse a un servidor SMTP, Conexión a un Conexión a un servidor SMTP, servidor SMTP, Inicio de sesión en el servidor SMTP, Desconexión del servidor SMTP, Desconexión del servidor IMAP

conectarse al servidor SMTP, Conectarse a SMTP un servidor Conectarse a

un servidor SMTPdesconectarse del servidor, Desconectarse del servidor

SMTP iniciar sesión en el servidor,

visión general, SMTP

recordatorio, Desconexión del servidor IMAP

envío de mensaje "hola", Conexión a un servidor SMTP envío de

mensaje, Inicio de sesión en el servidor SMTP

Encriptación TLS, Conexión a un servidor SMTP

SMTP, Atributo SMTPemboss, de ejecuciónEncriptaciónAtributos, de archivos PDF, Superposición de endswithpáginasMétodo(), El isX métododе cadenasSellos de tiempo, El módulo time, El módulo datetime, Revisión de la de función time PythonOperador secuencial a (==), Valores booleanos, Operadores de comparaciónNivel de ERROR, Niveles de registroerrores, Inicio de IDLE, Introducción de expresiones en el intérprete de comandos interactivo

choques y, Introducción de expresiones en el intérprete de comandos interactivo

ayuda para, Iniciar IDLE

caracteres de escape, String Literalsevaluation, definido, Introducir expresiones

en el Shell InteractivoHojas de cálculo Excel, Trabajar con Excelhojas de cálculo , Trabajar con hojas cálculo de Excelcálculo , Trabajar con hojas de Excel, Leer Exceldocumentos, Obtener hojas del libro de trabajo, Obtener hojas del libro de trabajo, Convertir entre letras y números de columna, Convertir entre letras y números de columna, Obtener filas y columnas de las hojas, Proyecto: Leer Datos de una Hoja de Cálculo, Paso 1: Leer los Datos de la Hoja de Cálculo, Paso 2: Rellenar la Estructura de Datos, Ideas para ilares, Programas SimIdeas para Programas Similares, Crear y Eliminar Hojas, Crear y Eliminar Hojas, Crear y Eliminar Hojas, Escribir Valores en Celdas, , Escribir Valores en CeldasProyecto: Actualizar una Hoja de Cálculo, Configurar el Fuente Estilo de de las Celdas, Objetos de Fuente, Fórmulas, Fórmulas, Configurar el Alto de Fila y el Ancho de Columna, Fusionar y Deshacer Celdas, Congelar Paneles, del Servidor IMAP

soporte de aplicaciones, Trabajar con hojas de cálculo Excel

gráficos de en, Congelar paneles

ancho de columna, Fórmulas

Conversión entre letras de columna y números, Conversión entre letras de columna y números

crear documentos, Ideas para programas similares crear

hojas de cálculo, Crear y eliminar hojas eliminar hojas de

cálculo, Crear y eliminar hojas estilos de fuente, Configurar

el estilo de fuente de las celdas

fórmulas en, Font Objects

congelación de paneles, fusión y separación de celdas

Obtención de Obtención de hojas del libro de trabajovalores de celda,

Obtención de filas y columnas, Conversión entre letras y números de columna

Obtención de Obtención de hojas del libro de trabajonombres de hojas de cálculo,

fusionar y desunir celdas, Configurar el Alto de Fila y el Ancho de Columna

abriendo documentos, Leer documentos Excel

openpyxl module, Trabajar con hojas de cálculo Excel Trabajar con hojas de cálculo Excel overview,

Leyendo Archivos, Obteniendo Filas y Columnas de las Hojas, Proyecto: Leyendo Datos de una Hoja de Cálculo, Paso 1: Leer los Datos de la Hoja de Cálculo, Paso 2. Leer los Datos de la Hoja de Cálculo, Paso 3. Leer los Datos de la Hoja de Cálculo Rellenar la estructura de datos

Visión General, Obteniendo Filas y Columnas de las Hojas Poblando la Estructura de Datos, Paso 1: Leer los Datos de la Hoja de Cálculo

leyendo datos, Proyecto: Lectura de Datos de una Hoja de Cálculo

escritura de resultados en un fichero, Paso 2: Poblar la Estructura de Datos

y correos electrónicos recordatorios proyecto, Desconexión del servidor IMAP filaaltura de la , Formulassalvando libros de trabajo, Ideas para programas similaresactualización, Escritura de valores en celdas, , Proyecto: Actualización de una Hoja de CálculoEscritura de valores en celdas

visión general, escribir valores en celdas

configuración, Proyecto: Actualización de una hoja de cálculo

libros de trabajo vs., Trabajar con hojas de cálculo Excelescribir valores en celdas, Crear y eliminar SheetsExceptionobjetos , Raising Exceptionsexceptions, La declaración global, Depuración, Raising Exceptions, Aserciones

aserciones y, Aserciones

Obtención del rastreo como cadena, Manejo de excepciones, La declaración global

crianza, depuración

ejecución, programa, control de flujo, bloques de código, importación de módulos, Pausa hasta una fecha específica

definida, Control de flujo

visión general, Bloques de código

pausar hasta una hora específica, Pausar hasta una fecha

específica terminar el programa con sys.exit(), Importar módulos
Función exists(), Búsqueda del tamaño de los archivos y del contenido de las salidacarpetasCódigos de , Lanzamiento de otros programas desde Pythonexpand palabra clave, Rotación y volteo de imágenesOperador de exponente (**), Introducción de expresiones en el intérprete , Introducción de expresiones en el de comandos interactivoExpresionesintérprete de comandos interactivo, Mezcla de operadores booleanos y de comparación

condiciones y, Mezcla de operadores booleanos y de comparación

en el intérprete de comandos interactivo, Introducción de expresiones en el intérprete de comandos interactivo método expunge()método filesextract(), Obtención del cuerpo de un mensaje sin procesarmétodo (), Extracción de archivos ZIP, de archivos ZIP filesextractallExtracción,
Extracción de archivos ZIP

F

Excepción FailSafeException, Paso 2: Establecer Coordenadas

valores "falsey", continuar Declaraciones

método fetch(), Realización de la búsqueda, Editor de

archivos con límites de tamaño, Nombres de variables

gestión de archivos, Lectura y escritura de archivos, Lectura y escritura de archivos, Archivos y rutas de archivos, El directorio de trabajo actual, El directorio de trabajo actual, Rutas absolutas vs. relativas, El os.pathmódulo, El módulo os.path, El módulo os.path, Manejo de Absolutas y RelativasRutas , Manejo de Rutas Absolutas y Relativas, Búsqueda de Tamaños de Archivos y Contenidos de Carpetas, Búsqueda de Tamaños de Archivos y Contenidos de Carpetas, El de ArchivosProceso de , Lectura/EscrituraApertura de Archivos con la Función open(), Lectura de Contenidos de Archivos, Escritura en Archivos, Almacenamiento de Variables con el Módulo shelve, Paso 4: Escribir Contenido en los Archivos de Cuestionario y Clave de Respuestas, Organización de Archivos, de CopiaArchivos y Carpetas, Copia de Archivos y Carpetas, Movimiento y Renombrado de Archivos

y carpetas, Borrado permanente de archivos y carpetas, módulo send2trash, Borrado seguro con el Recorrido por un árbol de directorios, Compresión de archivos con el módulo zipfile, Extracción de archivos ZIP, Extracción de archivos ZIP, Creación y adición a archivos ZIP, Paso 3: Formar el nuevo nombre de archivo y renombrar los archivos

rutas absolutas vs. relativas, El directorio de trabajo actual barra invertida

vs. barra diagonal, Archivos y rutas de archivos

Archivos comprimidos, Recorrer un árbol de directorios, módulo zipfileComprimir archivos con el , Extraer de archivos ZIP, Extraer de archivos ZIP, Paso 3: Formar el nuevo nombre de archivo y renombrar los archivos

crear copias de seguridad, Paso 3: Formar el nuevo nombre de archivo y renombrar

los archivos crear archivos ZIP, Extracción de archivos ZIP

Extracción de archivos ZIP, Descripción general de

la extracción de archivos ZIP, Recorrido por un

árbol de directorios

lectura de archivos ZIP, compresión de archivos con el módulo zipfile

creación de directorios, Rutas absolutas vs. actualrelativasdirectorio de trabajo , El directorio de trabajo del tablero multiclipactualproyecto , Paso 4: Escribir contenido en los archivos del cuestionario y de la clave de respuestasapertura de archivos, El proceso de lectura/escritura de archivosmódulo os.path, El ospathmódulo., El módulo os.path, Manejo de rutas absolutas y relativas, Manejo de rutas absolutas y relativas, Búsqueda de tamaños de archivos y contenidos de carpetas

rutas absolutas en, El módulo os.path

tamaño de los archivos, manejo de rutas absolutas y relativas

contenido de carpetas, Manejo de rutas absolutas y relativas, El

módulo os.path

validez de la ruta, Cómo encontrar el tamaño de los archivos y el

contenido de las carpetas en rutas relativas, El módulo os.path

Visión general, Lectura y escritura de rutas de archivos, Lectura y escritura de archivosarchivos de texto plano frente a archivos binarios, Búsqueda de tamaños de archivos y contenidos de carpetaslectura de archivos, Apertura de archivos con la open(función)renombramiento de archivos, estilos de fecha, Creación y adición a archivos ZIPguardado de Guardado de variables con el variables con la función pformat(), módulo shelvemódulo, end2trashBorrado permanente de archivos y shelvecarpetasmódulo, Escritura en archivosmódulo shelve, Organización de archivos, Copia de archivos y carpetas, Copia de archivos y carpetas, Movimiento y cambio de nombre de archivos y carpetas

copiar archivos/carpetas, Organizar archivos

eliminar archivos/carpetas, mover y renombrar archivos y carpetas mover

archivos/carpetas, copiar archivos y carpetas

renombrar archivos/carpetas, Copiar archivos y carpetas

Recorrer árboles de directorios, Borrados seguros con el send2trashEscribir móduloarchivos, Leer el contenido de nombres de archivosarchivosDefinir, Leer y escribir archivosObjetos de archivo, Abrir archivos con la función open()Método findall(), Correspondencia codiciosa y no codiciosaMétodos find_element_by_*navegador controlado por SeleniumMétodos find_elements_by_*, Inicio de un , Iniciar un navegador controlado por SeleniumFirefox, desarrollo, Apertura herramientas de de las herramientas de desarrollo del navegadorTecla de búsqueda FLAGGED, Realización de la búsquedaInversión de imágenes, Rotación e inversión de float()imágenesFunción, Función len()Números de coma flotante, entero, coma flotante y Tipos de datos cadena, Funciones str(), int() y float(), La hora.sleep() Función

equivalencia de enteros, Visión general de las funciones str(), int() y

float(), Tipos de datos enteros, de coma flotante y de cadena redondeo,

Función time.sleep()

Control de flujo, Control de flujo, , Control de flujoValores booleanos, de comparaciónOperadores , Operadores booleanos , Mezcla de booleanos y de binariosoperadores , comparaciónMezcla de operadores booleanos y de comparación, Bloques de código, Sentencias , ifSentencias else, Sentencias while Loop, Un molesto whilebucle , Sentencias continue

operadores binarios, operadores de comparación

bloques de , Mezcla de operadores booleanos y de comparación

Valores booleanos y, Control de flujo

sentencias break, un molesto bucle while operadores

de comparación, valores booleanos

condiciones, mezcla de operadores booleanos y de comparación

sentencias continue, sentencias continue

sentencias elif, sentencias else

sentencias else, sentencias if if,

Bloques de código, Control de

flujo

utilización conjunta de operadores binarios y de comparación, Operadores booleanos binarios

bucles while, sentencias del bucle while

carpetas, Lectura y escritura de archivos, Archivos y rutas de archivos, El actualdirectorio de trabajo , El directorio de trabajo actual, absolutas vs. Rutas relativas, El módulo os.path, El módulo os.path, El módulo os.path, Manejo de rutas absolutas y relativas, rutas, Manejo de absolutas y relativasBúsqueda de tamaños de archivos y contenidos de carpetas, Organización de archivos, de Copiaarchivos y carpetas, Copia de archivos y carpetas, Movimiento y cambio de nombre de archivos y carpetas, Borrado permanente de archivos y carpetas, Borrado seguro con el módulo send2trash, Paso 3: Formar el nuevo nombre de archivo y cambiar el nombre de los archivos, Proyecto: Copia de seguridad de una carpeta en un archivo ZIP, Averiguar del archivo ZIP, Paso 1: el nombre Paso 1: Averiguar el nombre del archivo ZIP

rutas absolutas vs. relativas, El directorio de trabajo actual

Cómo Proyecto: Copia de seguridad de una carpeta en un archivo ZIPAveriguar
hacer una copia de seguridad en un archivo ZIP, Paso 3: Formar el nuevo nombre del archivo y renombrar los archivos, , archivo ZIP, Paso 1: el nombre del Paso 1: Averiguar el nombre del archivo ZIP

crear un nuevo archivo ZIP, Paso 1: Averiguar el nombre del archivo ZIP

averiguar el nombre del archivo ZIP, Proyecto: Hacer una copia de seguridad de

una carpeta en un archivos ZIPárbol de directorios andante de , Paso 1: Averiguar

el nombre del archivo ZIP

barra invertida vs. barra diagonal, Archivos y rutas de archivoscopiar, Organizar archivoscrear, Rutas absolutas vs. relativasdirectorio de trabajo actual, El directorio de trabajo actualdefinir, Leer y escribir archivosborrar permanentementeBorrar permanentemente archivos y carpetasmover, Mover y renombrar archivos y carpetasborrar usando el send2trash, , Copiar archivos y carpetasmódulo os.path, , El módulo os.path, El módulo os.path, El módulo os.pathManejo de rutas absolutas y relativas, Manejo de rutas absolutas y relativas, Encontrar tamaños de ficheros y contenidos de carpetas

rutas absolutas en, El módulo os.path

tamaño de los archivos, manejo de rutas absolutas y relativas

contenido de carpetas, Manejo de rutas absolutas y relativas, El

módulo os.path

validez de la ruta, búsqueda de tamaños de archivos y contenidos de carpetas

rutas relativas en, El módulo os.path
renombrar, Copiar archivos y carpetasrecorrer árboles de directorios, Borrados seguros con el módulo send2trashObjetos font, Establecer el estilo de fuente de las celdasestilos font, en hojas de cálculo Excel, Establecer el estilo de fuente de las celdasfor loops, sentencias continue, Uso de bucles for con listas, Los métodos keys(), values() e items()

resumen, continuar Declaraciones

uso de elementos de diccionario en, Métodos keys(), values() y items() uso de

listas con, Uso de bucles for con listas
format, Trabajar con de los datos de imagenel atributo , Typeformat_descriptionTrabajar con la lista TypeformData de los datos de imagen, Paso 2: Configurar coordenadasproyecto de relleno automático de formularios, Repaso de las funciones de PyAutoGUI Repaso de las funciones de PyAutoGUI,, Proyecto: Rellenador Automático de Formularios, Paso 1: Descifrar los Pasos, Paso 2: Configurar Coordenadas, Paso 3: Comenzar a Digitar Datos, Paso 3: Comenzar a Digitar Datos, Paso 4: Manejar Listas de Selección y Botones de Radio

visión general, Revisión de las funciones de PyAutoGUI

botones de opción, Paso 3: Empezar a escribir

datos Paso 3: Empezar a escribir datoslistas

de selección,

Establecer coordenadas, Paso 1: Averiguar los del proceso,

Proyecto: Rellenador automático de formularios

formulario de envío, Paso 4: Manejo de listas de selección y botones de radio

teclear datos, Paso 2: Establecer coordenadas

fórmulas, en hojas de cálculo Excel, método (), Envío de claves especialesbarra diagonal (/), Font ObjectsforwardArchivos y rutas de archivosclave de búsqueda FROM, Realización de la datetimebúsquedafunción(), El módulo datetime, Repaso deRepaso de las funciones de PyAutoGUI las funciones temporales de Pythonfunciones, Comentarios, Importación de módulos, Funciones, def Statements with Parameters, def Statements with Parameters, def Statements with Parameters, def Statements with Parameters, Valores de retorno y return Statements, retorno Valores dey return Statements, The global Statement, The global Statement,

(véanse también los nombres de las distintas funciones)

argumentos, Comentarios, def Declaraciones con parámetros

como "caja negra", La declaración global

incorporado, Importar módulos

def statements def Statements with Parameters ,manejo de

excepciones, The global Statement

argumentos de palabras clave, Valores de retorno y sentencias retorno

de valores de retorno y sentencias de retornoNingún valor y,

Descripción general de , Funciones

parameters, def def Statements with Parameters

Statements with Parameters return values, G

método get_active_sheet(), Obtener hojas del libro de trabajo método

get_addresses(), Obtener direcciones de correo electrónico de un mensaje sin

procesar método get_attribute(), Encontrar elementos en la página

Función getcolor(), Fundamentos de la imagen por ordenador, Trabajar con el tipo de datos de imagen

Función get_column_letter(), Conversión entre letras de columna y números

Función getcwd(), El directorio de trabajo actual

función get(), Comprobación de la existencia de una clave o un valor en un diccionario, Descarga de archivos de la Web con el módulo de solicitudes

Resumen, Comprobación de la existencia de una clave o un valor en un diccionario

módulo requests, Descarga de archivos de la Web con el módulo requests

método get_highest_column(), Obtención de celdas de las hojas, Paso 1: Abrir el archivo de Excelel métodoget_highest_row(), Obtención de celdas de las hojasel métodoget_payload(), Obtención de direcciones de correo electrónico de un mensaje (), sin procesarla Cambio de píxeles individuales, Desplazamiento del ratón, Análisis de la captura de pantallael métodoget_sheet_by_name(), Obtención de hojas del libro de trabajoget_sheet_names(), Obtención de hojas del libro de trabajogetsize(), Manejo de absolutas y rutas relativasget_subject(), Obtención de direcciones de correo electrónico de un sin mensaje procesaregetText(), Lectura de documentos de GIFWordFormato, Trabajo con el tipo de datos de imagenámbito global, Variables locales y globales con el mismo nombreGmail, Conexión a un servidor SMTP, Inicio de sesión en el SMTPservidor , Recuperación y eliminación de correos electrónicos con el método IMAPgmail_search(), Límites de tamañoGoogle Maps, Web gráfica ScrapingAutomatización de la interfaz de usuario, Revisión de las funciones PyAutoGUI (véase Automatización de la GUI (interfaz gráfica de usuario))operador greater than (>), BooleanosOperador mayor o igual que (>=), Valores Valores rápidaBooleanosCoincidencia, Coincidencia de uno o más con el más, El carácter comodín

dot-star for, El personaje comodín

en expresiones regulares, Coincidencia de uno o más con el signo más
método group(), Creación de objetos Regex, Revisión de la coincidencia de coincidencia de , expresiones regularesgrupos, expresión regular, Revisión de la expresiones regularesAgrupación con paréntesisCoincidencia de varios grupos , tuberíacon la , Coincidencia opcional con el signo de interrogación, opcional con Coincidencia Coincidencia de uno o más con el signo más, Coincidencia de uno el signo de interrogación, o más con el signo más, Coincidencia codiciosa y no codiciosa

Coincidencia, Coincidencia de varios grupos el tubocon Coincidencia de uno más con el signo más, Coincidencia opcional ,con el signo de interrogación Coincidencia opcional con el signo de interrogación, o Coincidencia de uno o más con el signo más, , Emparejamiento codicioso y no codicioso

greedy, Emparejamiento Emparejamiento greedy y

nongreedyde uno o más con el plus nongreedy,

uno o más, Coincidencia con el signo de interrogación signo

opcional opcional, Coincidencia de varios grupos con las la

tuberíarepticiones específicas de , Coincidencia de uno o más

con el más

cero o más, Coincidencia opcional con el signo de interrogación
usando paréntesis, Revisión de la de Expresiones Regularesusando Coincidenciael carácter pipe en, Agrupando con Adivina ParéntesisPrograma, el NúmeroManejo de la GUI ExcepcionesAutomatización de(interfaz gráfica de usuario), Controlando el Teclado y Ratón con la GUIControlando ella con la Automatización de la GUIControlando el Teclado y el Ratón con la la GUIAutomatización el Teclado y el Ratón de , , Automatización de , Pausas y FailSafes, Moviendo el Ratón, Paso 3: Obtener e Imprimir las Coordenadas del Ratón, Haciendo Clic con el Ratón, Haciendo Clic con el Ratón, Arrastrando el Ratón, Desplazando el Ratón, Proyecto: Extendiendo el Programa mouseNow, Reconocimiento de Imágenes, Reconocimiento de Imágenes, Enviando una Cadena desde el Teclado, Nombres de Teclas, Presionando y Soltando el Teclado, Revisión de las Funciones PyAutoGUI

(véase también proyecto de relleno de formularios)

teclado de control, Reconocimiento de imágenesReconocimiento de imágenes, , Envío de un

Cadena del teclado, nombres de las teclas, pulsar y soltar el teclado

Combinaciones de teclas de acceso rápido, Pulsar y soltar los

tecladonombres de las teclas del , Enviar una cadena desde el

teclado

pulsar y soltar, Nombres de las teclas

enviar cadena desde el teclado, Reconocimiento de imágenes
Controlar el ratón, Pausas y FailSafes, Paso 3: Obtener e imprimir las del coordenadas ratón, Hacer clic con el ratón, Hacer clic con el ratón, Arrastrar el ratón

hacer clic con el ratón, hacer clic con el

ratón arrastrar el ratón, hacer clic con el

ratón

Desplazamiento del ratón, Arrastrar el ratón
determinar la posición del ratón, Mover el imágenesratónreconocimiento de , Proyecto: Ampliación del programa mouseNowinstalación del módulo pyautogui, Controlar teclado ratón interfaz gráfica de usuariofinalización del programa, el Desplazar el y el Controlar el teclado y el ratón con la automatización de la interfaz gráfica de de pantallacon la automatización de la Controlar el teclado y el con la automatización de la interfaz gráfica ratónparar el programa, Controlar el teclado y el ratón con la automatización de la interfaz gráfica de usuarioratón de general usuariovisión, usuariocapturas,

H

Directiva %H, Pausa hasta una fecha específica

Carácter almohadilla (#), Cadenas multilínea con entre comillas

triplesencabezados , Documento Word, Ayuda para escribir

documentos Word, Inicio de IDLE, Cómo encontrar ayuda

preguntar en línea, Cómo encontrar ayuda

para los mensajes de error, Starting IDLE
Combinaciones de teclas de acceso rápido, Pulsar y soltar el hotkeytecladoFunción , ()Pulsar y soltar el teclado, Revisión de las de funciones PyAutoGUIHotmail.com, Conectarse a un servidor SMTP, Recuperar y eliminar correos electrónicos con IMAPHTML (Hypertext Markup Language), Guardar

Archivos descargados en el disco duro, Cómo guardar archivos descargados en el durodisco, repaso rápido, Cómo ver el código HTML fuente de una página web, Cómo utilizar las herramientas de desarrollo para buscar elementos HTML

herramientas de desarrollo del navegador y, Visualización del HTML fuente de una página web búsqueda de elementos, Uso de las herramientas de desarrollo para buscar elementos HTML recursos de aprendizaje, Guardar archivos en el disco duro descargados Guardar archivos descargados en el disco durodescripción general,

ver página fuente, Un rápido repaso

I

directiva %I, Pausa hasta una fecha específica

atributo id, Un repaso rápido

IDLE (entorno de desarrollo interactivo), Descarga e instalación de Python, Inicio de IDLE, Introducción de expresiones en el intérprete de comandos interactivo, Nombres de variables, Copia y pegado de cadenas con el módulo pyperclip, Desactivación del registro, Más de, Depuración de un programa de adición de números creación de programas, Nombres de variables

depurando en, Desactivando el registro, Over, Depurando un programa de suma de números

visión general, Desactivación del registro paso a paso del programa,

Sobre

Uso de puntos de interrupción, Depuración de un programa de suma de números
expresiones en, Introducir expresiones en el Shelloverview interactivo, Inicio de IDLEejecutar scripts fuera de, Copiar y pegar cadenas con el pyperclip Modulestarting, Descarga e instalación de Pythonif declaraciones, Bloques de código, mientras que las declaraciones de bucle

visión general, Bloques de código

Uso en el bucle while, sentencias del bucle while

Módulo imageDraw, Ideas para Programas SimilaresObjetos imageDraw, Ideas para Programas SimilaresObjetos imageFont, Dibujando Objetos ExampleImage, Manipulando Imágenes con Pillowimages, Fundamentos de la Imagen por Ordenador, Fundamentos de la Imagen por Ordenador, Colores y Valores RGBA, Coordenadas y , Tuplas BoxCoordenadas y Tuplas Box, Trabajando con el ImagenTipo de Datos de , Trabajando con el Tipo de Datos de Imagen, Copiando y Pegando Copiando y Pegando Imágenes en Otras Imágenes en Otras Imágenes, , , Rotando y Volteando ImágenesCopiando y Pegando Imágenes en Otras ImágenesImágenes, Rotando y Volteando Imágenes, Cambiando Píxeles Individuales, Proyecto: Añadir un Logotipo, Ideas para Programas Similares, Ideas para Programas Similares, Ideas para Programas Similares, Líneas, Líneas, Líneas, Ejemplo de Dibujo,
Proyecto: Ampliación del Programa mouseNow

añadiendo logotipo a, Proyecto: Añadir un logotipo

atributos para, Trabajar con tipo de datos de

imagentuplas de cuadro de Coordenadas y tuplas de

cuadro,

valores de color en, Fundamentos de la imagen por ordenador

coordenadas en, Colores y valores RGBA

copiar y pegar, Copiar y pegar imágenes en otras imágenes recortar, Trabajar con el tipo de datos de imagen

dibujar Ejemplo de dibujoen, para programas similares, Ideas para programas similares, Ideas para programas similares, Líneas, Líneas, Líneas,

programa de ejemplo, Líneas

elipses, Líneas

Módulo ImageDraw, Ideas para similares Ideas para

programas similareslíneas de programas ,

puntos, Ideas para programas similares

polígonos, Líneas

rectángulos, Líneas

texto, Ejemplo de dibujo
Voltear, Girar y Voltear Imágenesabrir con Almohadilla, Coordenadas y Tuplas de Cajamanipulación de píxeles, Cambiar individualesreconocimiento píxelesde, Proyecto: Extender el ratónProgramar ahoradimensionar, Fundamentos deordenadorrotar, Copiar y pegar imágenes en otras imágenesvalores RGBA, la imagen por Rotar y voltear transparentesimágenespíxeles, Copiar y pegar imágenes en otras imágenesIMAP (mensajes de InternetProtocolo de acceso a), Desconexión del servidor SMTP, Conexión un a servidor , Conexión a un servidor IMAPConexión a un IMAPIMAP, servidor , Límites de tamaño, Obtener el cuerpo de un mensaje , Obtener el sin procesarcuerpo de un mensaje sin procesar

definido, Desconexión del servidor SMTP

borrado de mensajes, Obtención del cuerpo de un mensaje sin procesar

Obtención del cuerpo de un mensaje sin procesar desconexión del servidor,

obtención de mensajes, Límites de tamaño

carpetas, Conexión a un servidor IMAP

iniciar sesión en el servidor, Conectarse a un servidor IMAP

Conectarse a un servidor IMAPbuscar mensajes,
Módulo imapclient, Desconexión del imapclientservidorSMTPObjetos, Recuperación y borrado de correos electrónicos con inmutablesIMAPTipos de datos , Tipos tipo lista: Strings and Tuplesimporting modules, Importar módulos,
Mover el ratón

visión general, Importar módulos módulo

pyautogui, Mover el ratón
Atributo imprint, Variable , Desplazamiento del Run AttributesimMouseindentation, Programa de Ejemplo: Bola 8 Mágica con una Listaíndices, El Tipo de Datos Lista, Índices Negativos, Obtención de la Longitud de una Lista con len(), Eliminación de Valores de Listas con Sentencias del, Diccionarios y Estructuración de Datos, Cadenas Multilínea con Comillas Triples

para diccionarios, Diccionarios y estructuración de datos (véase claves, diccionario)

para listas, El tipo de datos lista, Índices negativos, Obtención de la longitud de una

con

len(), Eliminación de Valores de Listas con Sentencias del cambio

de valores usando, Obtención de la Longitud de una Lista con

len() obtención de valor usando, El Tipo de Datos Lista

negativo, Índices negativos

eliminar valores de una lista mediante, Eliminar valores de listas con sentencias del
para cadenas, Cadenas de varias líneas con comillas triplesIndexError, El diccionario
Método Data Typeindex(), Métodos bucles infinitos, Un molesto bucle while,
Sentencias continue, Paso 1: Importar el nivel ModuleINFO, Utilización del operador
Modulein de registro, Los operadores in y not in, Comprobación de la existencia de
una clave o un valor en un diccionario, Indexación y segmentación de cadenas.

con diccionarios, Comprobación de la existencia de una clave o un valor en un
diccionario

usando con listas, Los Operadores in y not in

Uso de cadenas, indexación y segmentación de cadenas
Función input(), ComentariosComentarios, MétodosMétodos, Visión general del

inicio de sesión en el servidor SMTP, ,

utilizando para información sensible, Inicio de sesión en el servidor SMTP
instalación, Acerca de este libro, Paso 4: Guardar la imagen y buscar el
anteriorcómic , Trabajar con hojas de cálculo Excel, Controlar el teclado y el ratón
con la automatización de la interfaz gráfica de usuario, Instalar módulos de terceros

Módulo openpyxl, Trabajar con hojas de cálculo Excel

Módulo pyautogui, Controlar el teclado y el ratón con automatización GUI

Python, Acerca de este libro

módulo selenium, Paso 4: Guardar la imagen y buscar los Previous

Comicmódulos de terceros , Instalar módulos de terceros
int, Tipos de datos enteros, de coma flotante y de cadena

(véase también enteros)

división de enteros/operador de cociente (//), Introducción de expresiones en la Shellintegers interactiva, datos enteros, de coma flotante y de cadenaTipos de , Funciones str(), int() y float()

equivalencia en coma flotante, Funciones str(), int() y float()

Entorno de desarrollo interactivo de tipos de datos enteros, de coma flotante y de cadenas, Nombres de variables (consulte IDLE (entorno de desarrollo interactivo))shell interactivo, Nombres de variables (véase IDLE)Internet Explorer, herramientas para desarrolladores, Visualización del código HTML fuente de una página WebProtocolo de acceso a mensajes de Internet, Desconexión del servidor SMTP (véase IMAP (Protocolo de acceso a mensajes de Internet))intérprete, Python, Descarga e instalación de Pythonfunciónint(), len(Función (), Módulo)funciónisabsos.path Moduleisalnum(), Los métodos , lower(), isupper(), e islowerMethodsisalpha()upper()Los métodos upper(), lower(), isupper(), and islower() String Methodsfunción isdir() String Los métodos upper(), lower(), isupper(), e islower() String Methodsisdecimal, (), (), Encontrar tamaños de archivos y contenidos de carpetasmétodo is_displayed(), Encontrar elementos en la is_enabledpáginamétodo (), Encontrar elementos en la páginafunción isfile(), Método isis_selected(), Búsqueda de elementos en la ispacepáginaMétodo , ()Método isX_isistitleStringMétodo(), Método isX StringMétodo Método Método StringMétodoitalic, liter_contentisisupper(), isupper(), isupper() y islower())))) Atributos de ejecuciónMétodo items(), (), Método liter_content(), Método liter_content(), Método liter_content(, Método liter_content(, Método liter_content(, Método liter_content(Diccionarios frente a listasMétodo Método Listsiter_content(), Guardar archivos descargados en el disco duro

J

Directiva %j, Pausa hasta una fecha específica

método join(), Los métodos de cadena isX, Archivos y rutas de archivos, El módulo os.path, Paso 2: Crear e iniciar subprocesos

Formato JPEG, Trabajar con el tipo de datos de imagen

Archivos JSON, Trabajar con archivos CSV y datos JSON, Paso 3: Escribir el archivo CSV sin la Paso 3: Escribir el archivo CSV sin primera fila, la primera fila, JSON y las API, Leer JSON con la función loads(),

Lectura de JSON con la función loads()

API para, Paso 3: Escribir el archivo CSV sin la primera fila definida,

Trabajar con archivos CSV y datos JSON

resumen del formato, Paso 3: Escribir el archivo CSV sin la primera filaleer , JSON y APIs

y proyecto de datos meteorológicos, Lectura de JSON con la función loads()

escritura, Lectura de JSON con la función loads()
justificación del texto, Los métodos de cadena join() y split()

K

Teclado, Reconocimiento de imágenes, , Nombres de las teclasEnviar una cadena desde el teclado, Pulsar y soltar el teclado

controlar, con PyAutoGUI, el reconocimiento de imágenes, los nombres de las teclas, pulsar y soltar el teclado

combinaciones de teclas de acceso rápido, Pulsar el teclado y soltar

pulsar y soltar teclas, Nombres de las teclas

enviar cadena desde el teclado, Reconocimiento de imágenes
nombres de las teclas, Envío de una cadena desde el tecladoExcepción, KeyboardInterruptPaso 2: Seguimiento e impresión de los tiempos de vuelta, Mover el ratón, Paso 1: Importar la función ModulekeyDown(), KeyDownNombres de las teclas, Pulsar y soltar el teclado, Repaso de las funciones de PyAutoGUIkeys, dictionary, Diccionarios y estructuración de datosMétodo (), Diccionarios vs. Diccionarios.
ListasfunciónkeyUp(), Nombres de las teclas, Pulsar y soltar el teclado, Revisión de las funciones de las teclasPyAutoGUIargumentos de , Valores de retorno y sentencias return.

L

Tecla de búsqueda LARGER, Realización de la búsqueda

launchd, Lanzamiento de otros programas desde Python

Lanzamiento de programas, Paso 2: Crear e iniciar hilos, Lanzamiento de otros programas desde Python, , PythonLanzamiento de otros programas desde PythonLanzamiento de otros programas desde Python, Lanzamiento de otros programas desde , cron, , cron, cron, Programador de tareas, launchd Programador de tareas, launchd Programador de tareas, launchd y Programador de tareas, launchd y y y cronProyecto: Programa simple de cuenta atrás

y proyecto de cuenta atrás, Project: Programa simple de cuenta atrás

abrir archivos con aplicaciones predeterminadas, Programador de tareas, launchd

cron Programador de tareas, launchd y crony abrir sitios web,

resumen, Paso 2: Crear e iniciar subprocesos

pasar argumentos de línea de comandos a procesos, Lanzar otros programas desde Python

método poll(), Lanzamiento de otros programas Python

Lanzamiento de otros programas desde Python desde ejecución de

scripts Python, Programación de tareas, launchd y

cronProgramador de tareas, launchd y cron, función sleep(),

método wait(), Lanzamiento de otros programas desde Python
Función len(), La función input(), Obtención de la longitud de una lista con len(), Documentos Word

encontrar el número de valores en la lista, Obtener la longitud de una lista

con len() visión general, La función input()
Operador menor que (<), Valores booleanosOperador menor o igual que (<=), Valores booleanosLibreOffice, Trabajar con hojas de cálculo Excel, Paso 4: Guardar los resultadosSaltos de línea, Documento Word, Añadir LineChart()encabezadosFunción, Carácter de continuación de (\), Programa de ejemplo: Magic 8 Ball con un método Listline(), línea Ideas para programas similaresestilos enlazados, Estilización de objetos de párrafo y ejecuciónLinux, ¿Qué es Python?, Descarga e instalación de Python, Inicio de IDLE, Archivos y rutas de archivosLanzamiento , de otros programas desde Python, Lanzamiento de otros programas desde Python,

Programador de tareas, launchd y cron, Abrir archivos con aplicaciones predeterminadas, Controlar el teclado y el ratón con la la interfaz gráfica de usuarioautomatización de , Instalar módulos de terceros, La herramienta pip, Ejecutar programas Python en OS X y Linux

barra invertida vs. barra diagonal, Archivos y rutas de

archivos cron, Ejecutar otros programas desde Python

instalar Python, Descargar e instalar Python instalar módulos

de terceros, La herramienta pip

lanzar procesos desde PythonLanzar otros programas desde Python,

salir del programa de automatización, Controlar el teclado y el ratón con la automatización GUI

apertura de archivos con aplicaciones predeterminadas, Programador de tareas,

launchd y cron herramienta pip activada, Instalación de módulos de terceros

Soporte de Python, ¿Qué es Python?

Ejecutar programas Python en, Ejecutar programas Python en OS X y Linux

Iniciar IDLEIniciar IDLE,

Filosofía Unix, Abrir archivos con aplicaciones predeterminadas

Función listdir(), Manejo de absolutas y list foldersrutas relativasMétodo (), Conexión a un servidor listIMAPFunción(), Objetos de lectura, Listas de reconocimiento de imágenes, El tipo de datos de lista, El tipo de datos de lista, negativosÍndices , Índices negativos, Obtención de la longitud de una lista con len()Obtención una , la longitud de con len)de lista (, Obtención de la longitud de una lista con len(), Obtención de la una longitud de lista con len(), Eliminación de valores de listas con sentencias del, Eliminación de Valores de Listas con Sentencias del, Uso de Bucles for con Listas, Los Operadores in y not in, Los Operadores in y not inLos , Operadores in y not in, El Truco de la Asignación Múltiple, Métodos, Métodos, Métodos, Métodos, Adición de Valores a Listas con los Métodos append() e insert(), Eliminación de Valores de Listas con remove(), Ordenación de los Valores de una Lista con el Método sort(), Tipos de Listas: Cadenas y Tuplas, la TuplaLos Datos de

TipoEl tipo de datos diccionario, Pasar referencias Pasar referencias,, , Un tablero de tres en raya

método append(), Métodos

operadores de asignación aumentada, El truco de la asignación múltiple

cambiar valores usando índice, Obtener de lenla longitud una lista con ()

Obtener la longitud de una lista con len()concatenación de,

función copy(), Paso de referencias Paso de

referencias función deepcopy(), diccionarios vs., El

tipo de datos diccionario

encontrar el número de valores usando len(), Obtener la longitud de una lista con

len() obtener sublistas con slices, Índices negativos

obtención de valor mediante índice, El del tipo de

datos listamétodo index() , Métodos

operador in, Los operadores in y not in método

insert(), Métodos

función list(), El tipo de datos tupla

Magic 8 Ball programa de ejemplo utilizando, Ordenar los valores de una lista con el método sort()

truco de asignación múltiple, Los operadores in y not in

Tipos de datos mutables vs. inmutables, Tipos tipo lista: Strings and Tuples índices

negativos, Índices negativos

anidación, Un tablero de tres en raya

operador not in, Visión general de los operadores in

y not in, El tipo de datos lista

método remove(), Añadir valores a listas con append() e insert()
Métodos

eliminación de valores de, Eliminación de valores de listas con del sentencias

Replicación de de, Obtención de la longitud de una lista con len()

método sort(), Eliminación de valores de las listas con remove()

almacenamiento de variables as, Eliminación de valores de listas con sentencias

del uso Uso de bucles for con listascon bucles for,
Método ljust(), Los métodos de cadena join() y split(load_workbook)Función ,
()Lectura de documentos de loadsExcelFunción (), JSON y APIs, Paso 2: Descargar el Datalocal scope, Local and Global ScopelocateAllOnScreen(), Función Image RecognitionlocateOnScreen(), Proyecto: Ampliación del mouseNow Programlocationatributo , Búsqueda de elementos en la páginaelogging, Utilización del módulo logging, Utilización del módulo logging, Niveles de logging, Desactivación del logging

deshabilitar, Niveles de

registro en archivo,

Deshabilitar registro

niveles de, Uso del módulo de registro

print() y, Utilización del módulo logging
módulo logging, Uso de una aserción en una simulación de semáforologging out, de programa de automatización, Control del teclado y el ratón con GUI Automationmétodo login(), Conexión a un servidor SMTP, Conexión a un servidor IMAP, Paso 3: Enviar personalizados por correo recordatorios electrónicologogo, añadir a una imagen, Proyecto: Añadir un Logotipo, Proyecto: Añadir un Logotipo, Paso 1: Abrir la Imagen del Logotipo, Paso 2: Recorrer todos los Archivos y Abrir las Imágenes, Paso 3: Redimensionar las Imágenes

bucle sobre archivos, Paso 1: la imagen del Abrir logotipo

abrir la imagen del logotipo, Proyecto: Añadir logotipouna

vista general del , Paso 3: Redimensionar las imágenes

redimensionar imagen, Paso 2: Recorrer todos los archivos y abrir las imágenes

método logout(), Obtención del cuerpo de un mensaje sin log recordprocesarObjetos, Uso de una aserción en una simulación de semáforo bucles, whilesentencias , un molesto bucle while, sentencias continue, sentencias , continueun bucle while equivalente, Uso de bucles for con listas, Lectura de datos de objetos de lectura en un bucle for

sentencias break, Un molesto bucle while

sentencias continue, sentencias continue bucle

for, sentencias continue

función range() para, Un bucle while equivalente

lectura de datos de un archivo CSV, Lectura de datos de objetos de lectura en un bucle for

Uso de conUso de bucles for con listaslistas ,

bucle while, sentencias del bucle while
Método lower(), Los métodos de cadena upper(), lower(), isupper() e islower()Método strip(), Justificar texto con rjust(), ljust() y center()

M

Directiva %M, Pausa hasta una fecha específica

Directiva %m, Pausa hasta una fecha específica

Mac OS X, Archivos y rutas de archivos (consulte

OS X)

Programa de ejemplo Magic 8 Ball, Ordenar los valores de una lista con sort() Método

Función makedirs(), Rutas absolutas vs. relativas, Paso 2: Recorrer todos los archivos e imágenes abiertas

mapas, abrir cuando se copia la ubicación, Web Scraping, Web Scraping, Scraping, Web Paso 1: Averiguar la URL, portapapeles Paso 3: Manejar el del , Paso 3: Manejar el del portapapeles contenidoe iniciar el navegadorcontenidoe iniciar el navegador

averiguar URL, Web Scraping

Manejar el contenido del portapapeles, Paso 3: Manejar el contenido del portapapeles e iniciar el navegador

manejo del argumento de la línea de comandos, Paso 1: Averiguar la URL

lanzar el navegador, Paso 3: Manejar el contenido del portapapeles y lanzar el navegador

visión general, Web Scraping
Objetos coincidentes, Creación de objetos Regexmath, ¿Qué es Python?, Introducción de expresiones en el intérprete de comandos interactivo

operadores para, Introducción de expresiones en el intérprete de comandos interactivo

programación y, ¿Qué es Python?
método mergePage(), método , Los métodos de cadena upper(), lower(), isupper() e islower(método Superposición de PagesMessageobjetos , Envío de mensajes de textométodos, Métodos, Métodos, ,Métodos Métodos, Añadir valores a las listas con los métodos append() e insert(), Eliminar valores de las listas Comprobación setdefault()con remove(), , El), El Diccionarios frente a listas, Diccionarios frente a listas, Diccionarios frente a listas, de la existencia de una clave o un valor en el diccionarioLists, Comprobación de la existencia de una clave o un valor en un diccionario, El método setdefault(), Los métodos upper(), lower(), isupper() e islower() para cadenas() para Los métodos upper(), lower(), isupper() e islower() para cadenas, upper, ,, , Los métodos upper(), lower(), isupper() e islowercadenas, Los métodos para cadenas Los métodos cadenas Los métodos para cadenas Los métodos para cadenas isX, ((isupper(), e islowere islower)), isupper(), e islower(upper()lower(), isupperlowerupper() lower(), ())(), para , , (), Los Los métodos Los métodos join() y split() String, Los join() y split(métodos métodos métodos StringisX isX String, Los , Los métodos join() y split() String, Los métodos join() y split() String, Justificar texto con isX String, Los métodos isX String, Los métodos isX String, String,) rjust(), ljust() y (),center Justificar con rjustJustificar texto con rjust(), ljust() y center()texto (), ljust() y center(), , Eliminar espacios en blanco con strip(), rstrip() y lstrip()Eliminar espacios en blanco con strip(), rstrip() y , lstrip(), Girar y voltear imágenes

encadenamiento de llamadas, definición de rotar y

voltear imágenes, métodos

diccionario, Diccionarios vs. Listas, Diccionarios vs. Diccionarios vs. Listas, Listas, Comprobación de la existencia de una clave o un valor en un diccionario, El método setdefault()

método get(), Comprobación de la existencia de una clave o un valor en un diccionario método items(), Diccionarios frente a listas

método keys(), Diccionarios vs. Listas método

setdefault(), El método setdefault()

método values(), Diccionarios vs. Listas
list, Métodos, Métodos, , Añadir valores a las listas con los métodos append() e insert(), MétodosEliminar valores de las listas con remove()

método append(), Métodos

método index(), Métodos

método insert(), Métodos

método remove(), Añadir valores a listas con append() e insert()
Métodos

método sort(), Eliminación de valores de las listas con remove()
string, Métodos isX para cadenas, Métodos join() y split() para cadenas, Eliminación de espacios en blanco con strip(), rstrip() y lstrip()

método center(), Los métodos de cadena join() y split()

método copy(), Eliminación de espacios en blanco con los métodos strip(), rstrip() y lstrip() método endswith(), Los métodos isX String
isalnum(, método upper(), lower(), isupper(), e islower(String)Methodsisalpha(), método upper(), lower(), isupper(), e islower)islower() String Methodsislower(), método upper(), lower(), () String Methodsisdecimal(), método upper(), lower(), isupper(), e método , método isupperisistitleisX String Methodsmétodo isupper()(), método isupper() y método islower String Methodsmétodo , upper(), método lower(), método isupper() y método islower() String Methodsmétodo join(), Methods()isX método isX String Methodsmétodo ljust(), método join() y split() String ispace()método String Methodsmétodo (), método

Método)Método paste(), Eliminación de espacios en blanco con strip(), rstrip() rstrip()rjust(), Justificación de texto con rjust(), ljust() center(de texto con rjust(), ljust() y center()y y lstrip()Método , Justificación Método split(), Methodsstartswith(), El método)upper(), El método upper(), lower(), isupper(), e islower() String MethodsMicrosoft isX String El método isX String Methodsstrip(), Justificar texto con rjust(), ljust(), y center(Windows, Archivos y rutas de archivos (ver SO Windows)middleClick() function, Hacer clic con el ratón, Revisión de los módulos PyAutoGUI Functions,
Importar módulos, La herramienta pip

importar, Importar módulos

terceros, instalar, La herramienta pip

operador módulo/resumente (%), Introducir expresiones en la shell interactiva, El truco de la asignación múltipleMonty Python, ¿Qué es Python? ratón, Pausas y FailSafes, Mover el ratón, Mover el ratón Mover el ratón,, Paso, 3: Obtener e imprimir las coordenadas del ratón, 1: Importar el módulo, Paso : Importar el móduloPaso , 1: Importar el móduloPaso Hacer clic con el ratón, Hacer clic con el ratón,
Arrastrar el ratón, analizar la captura de pantalla

controlando, Pausas y FailSafes, Paso 3: Obtener e Imprimir las Coordenadas del Ratón, Clicando el Ratón, Clicando el Ratón, Arrastrando el Ratón

hacer clic con el ratón, hacer clic con el

ratón arrastrar el ratón, hacer clic con el

ratón

Desplazamiento del ratón, Arrastrar el ratón

Determinar la posición de, Desplazar el ratón, Desplazar el ratón, Desplazar el ratón, Paso 1: Importar el móduloPaso 1: Importar el módulo, ,
Paso Paso 1: Importar el módulo 1: Importar el móduloPaso 1:

Importar el módulo, obtener coordenadas,

handling KeyboardInterrupt exception, Paso 1: Importar el módulo Paso 1:

Importar el móduloimporting pyautogui module,

bucle infinito, Paso 1: Importar el módulo

visión general, Mover el ratón
y píxeles, identificando los colores de, Analizando la Captura de PantallafunciónmouseDown(), Haciendo Clic con el Ratón, Revisión de las Funciones PyAutoGUIfunciónmouse.position(), Paso Archivos y Carpetasmulticlipboard project, Paso 4. Escribir el Contenido de los Archivos del Cuestionario y de la Clave de Respuestas, Paso 4. Escribir el Contenido de los Archivos del Cuestionario y de la Clave de Respuestas, : Escribir Contenido en los 1: Importar el MódulomouseUp() function, Repaso Archivos Repaso de las Funciones PyAutoGUImover archivos/carpetas, Repaso de las Funciones PyAutoGUIde las Funciones PyAutoGUImove() function, Copiar y CarpetasmoveRel() function, Controlar el Movimiento del Ratón, Mover el Ratón, Hacer Clic con el Ratón, Repaso de las Funciones PyAutoGUImoveTo() function, Controlar el Movimiento del Ratón, Hacer Clic con el Ratón, Copiar Archivos y Clave de Respuestas Paso 4: Escribir Contenido en los Cuestionario y Clave de Respuestasde Cuestionario ,Archivos de , Paso 1: Configuración de Comentarios y Estantes, Paso 1: Comentarios y EstantesConfiguración de , Paso 2: Guardar Portapapeles , Paso 2: Guardar Contenido del con una Palabra ClaveContenido del Portapapeles con una Palabra Clave

Listado de palabras clave, Paso 2: Guardar el contenido del portapapeles con una palabra clave

cargar el contenido de las palabras claveGuardar el contenido del portapapeles con

un las palabras clave, Paso 2: Paso 4: Escribir el contenido en los archivos del

cuestionario y de las claves de respuestaresumen de ,

guardar el contenido del portapapeles, Paso 1: Comentarios y configuración de la estantería

configuración del archivo de estanterías, Paso 1: Comentarios y configuración de estanterías
comentarios multilínea, Cadenas multilínea con comillas multilíneatriplescadenas, Caracteres de múltipleescapetruco , Los in y not de la asignación truco de la asignación múltiplemultihilo, operadores inoperador de multiplicación (*), Introducción de expresiones en el intérprete de comandos interactivo, Obtención de la longitud de una lista con len(), El Multihilo, Multihilo, Multihilo Multihilo,, Pasar argumentos a la de destino del hilofunción, Pasar argumentos a la función de destino del hilo, Proyecto: Multithreaded XKCD Downloader, Proyecto: Multithreaded XKCD Downloader, Paso 1: Modificar el Programa para Usar una Función, Paso 2: Crear Threads, e Iniciar Paso 2: Crear e Iniciar Threads

problemas de concurrencia, paso de argumentos a la función de destino del subproceso

descarga de múltiples imágenes,, Proyecto: Multithreaded XKCD Downloader, Proyecto: Multithreaded XKCD Downloader, Paso 1: Modificar el Programa para Usar una Función, Paso 2: Crear e Iniciar Threads

crear e iniciar hilos, Paso 1: Modificar el programa para utilizar una función

usando la función downloadXkcd(), Proyecto: Multithreaded XKCD Downloader

Esperar a que finalicen los hilos, Paso 2: Crear e iniciar Paso 2: Crear e iniciar hilosvisión generalhilos método join(), , Multithreadingpasar argumentos a los hilos, Multithreadingstart() Multithreading, Pasar Argumentos a la Destino del Función de ThreadFunción Thread(), MultithreadingTipos de datos mutables, listaTipos tipo: Cadenas y Tuplas

N

NameError, Eliminación de valores de listas con del Statements el

método namelist(), Compresión de archivos con el módulo zipfile

clases , de Clases de caracterescaracteres negativos

índices negativosÍndices negativos,

listas anidadas y diccionarios, Un tablero de tres en raya

argumento de la palabra clave newline, Lectura de datos de objetos de lectura en un bucle for

Valor nulo, valores de retorno y declaraciones de retorno

concordancia no codiciosa, , el carácter comodínconcordancia codiciosa y no codiciosa

punto, estrella y signo de para, interrogación El carácter comodín

en expresiones regulares, Greedy y Nongreedy Matching

operador not equal to (!=), Operador in, Los operadores in y not in, Boolean Valuesnot Comprobación de la existencia de una clave o un valor en un diccionario, Indexación y segmentación de cadenas

utilizando con diccionarios, Comprobación de la existencia de una clave o un valor en un

Diccionario

uso de listas, operadores in y not in uso de cadenas,

indexación y segmentación de cadenas
operador not, Operadores booleanos NOTbinariosTecla de búsqueda , Ejecución de la función Searchnow(), El módulo datetime, Revisión de las temporales de Pythonfunciones

O

Tecla de búsqueda ON, Selección de una carpeta

función open(), El proceso de lectura/escritura de archivos, , Programador de tareas, launchd y cron, Web ScrapingManipulación de imágenes con Pillow

abrir archivos, El proceso de lectura/escritura de archivos

OpenOffice, Trabajar con hojas de cálculo Excel, Paso 4: Guardar el

Resultadosprograma abierto , Programador de tareas, launchd y cron

módulo openpyxl, instalación, Trabajar con hojas de cálculo Excel

operadores, de Introducción de Introducción intérprete expresiones en el de comandos interactivo, expresiones en el intérprete de comandos interactivo, Valores booleanos, comparaciónOperadores de, Operadores booleanos binarios, El de la asignación múltipletruco

asignación aumentada, El truco de la asignación múltiplebinario ,

Operadores de comparación

comparación, valores booleanos

definido, Introducción de expresiones en el intérprete de

comandos interactivo Introducción de expresiones en el intérprete

de comandos interactivomath,

Uso conjunto de operadores binarios y de comparación, Orden de operaciones de los operadores booleanos binarios, Introducción de expresiones en la Shellinteractiva OperadorOR Operadores booleanos ,binariosTecla de búsquedaOR, Realización de la búsquedaOS X, ¿Qué es Python?, Descarga e instalación de Python, Inicio de IDLE, Archivos y rutas de archivos, Inicio de otros programas desde

Python, Programador de tareas, launchd y cron, Abrir archivos con predeterminadasaplicaciones , Abrir archivos con aplicaciones predeterminadas, Controlar el teclado y el ratón con automatización GUI, Instalar de tercerosmódulos , La herramienta pip, Ejecutar programas Python en OS X y Linux

barra invertida vs. barra diagonal, Archivos y rutas de

archivos instalación de Python, Descarga e instalación de

Python instalación de módulos de terceros, La herramienta

pip

launchd, Lanzamiento de otros programas desde Python

lanzar procesos desde Python, Abrir archivos con aplicaciones por defecto

salir del programa de automatización, Controlar el teclado y el ratón con la automatización GUI

apertura de archivos con aplicaciones predeterminadas, Programador de tareas,

launchd y cron herramienta pip activada, Instalación de módulos de terceros

Soporte de Python, ¿Qué es Python?

Ejecutar programas Python en, Ejecutar programas Python en OS X y Linux

Iniciar IDLEIniciar IDLE,

Filosofía Unix, Abrir archivos con aplicaciones predeterminadas
Atributo de contorno, Atributos de ejecuciónOutlook.com, Conexión a un servidor SMTP, Recuperación y eliminación de mensajes de correo electrónico con IMAP

P

Directiva %p, Pausar hasta una fecha específica saltos

de página, Documento Word, Añadir encabezados

Objetos de página, Extraer texto de PDF Objetos de

párrafo, Documentos Word

párrafos, documento Word, Obtener el texto completo de un archivo .docx

parámetros, función, def Declaraciones con parámetros

paréntesis (), Tipos de datos mutables e inmutables, Revisión de la concordancia de expresiones regulares

análisis sintáctico, definido, Análisis sintáctico de HTML con el módulo

BeautifulSoup pasando argumentos, Comentarios

pasar referenciasPasar referencias,

contraseñas, Copiar y pegar cadenas Copiar y pegar cadenas con el módulo pyperclipcon el módulo pyperclip, pegar cadenas con el módulo pyperclip, Paso 1: Diseño del programa , Copiar y y estructuras de datos, Paso 1: Diseño del programa y estructuras de datos, Inicio de sesión en el servidor SMTP

específico de la aplicación, Inicio de sesión en el servidor SMTP

proyecto de gestión, Copiar y Copiar y pegar cadenas con el módulo pyperclippegar cadenas con el módulo pyperclip, pegar cadenas con el módulo pyperclip, , Copiar y Paso 1: Diseño del programa y estructuras de datos Paso 1: Diseño del programa y estructuras de datos,

argumentos de línea de comandos, Paso 1: Diseño de programas y estructuras de

datos Paso 1: Diseño de programas y estructuras de datoscopiar contraseña,

estructuras de datos, Copiar y pegar cadenas con el módulo pyperclip

descripción general, Copiar y pegar cadenas con el módulo pyperclip pastebin.com, método (), Eliminar espacios en blanco con strip(), rstrip() y lstrip(), SummarypasteCopiar pegar imágenes en otras imágenesy Copiar y pegar imágenes en otras rutas de imágenes, , Leer y escribir Archivos, Archivos y rutas de archivos, El directorio de trabajo actualEl actual, directorio de trabajo , El módulo os.path, ,módulo , El módulo os. El os.pathpathManejo de rutas absolutas y , Manejo de absolutas y relativasrutas , Búsqueda de tamaños de archivos y contenidos de carpetas.

absoluto vs. relativo, El directorio de trabajo actual

barra invertida vs. barra diagonal, Archivos y rutas de archivos

directorio de trabajo actualdel directorio de trabajo actual, Visión general ,

Lectura y escritura de archivos

Módulo os.pathEl . El os.path, , módulo path,módulo , El módulo ospath.osManejo de rutas absolutas y Manejo de absolutas , rutas y relativas, Búsqueda de tamaños de archivos y contenido de carpetas

rutas absolutas en, El módulo os.path

tamaño de los archivos, manejo de rutas absolutas y relativas

contenido de carpetas, Manejo de rutas absolutas y relativas, El

módulo os.path

validez de la ruta, búsqueda de tamaños de archivos y contenidos de carpetas

rutas relativas en, El módulo os.path

Variable PAUSE, Pausas y FailSafes, Paso 2: Configurar coordenadasObjetos PDFFileReader, Extraer texto , Extraer texto de PDFPDFsArchivos, Trabajar con PDF y documentos word, Documentos PDFde PDFs, Descifrar PDFs, Crear PDFs, Copiar páginas, Superponer , Superponer páginas, páginasCifrar PDFs, Cifrar PDFsCifrar , PDFs, Paso 1: Buscar todos los archivos PDF, Paso 1: Buscar todos los archivos PDF, Paso 2: Abrir cada PDF

Combinar páginas de varios archivos, Cifrar PDFs, Cifrar PDFs, Cifrar PDFs, Paso 1: Encontrar todos los archivos PDF Paso 1: Encontrar todos los archivos PDF,, Paso 2: Abrir cada PDF

Añadir páginas, Cifrar PDF

encontrar archivos PDF, Paso 1: Encontrar todos

los archivos Paso 1: Encontrar todos los archivos

PDF PDF abrir archivos PDF, descripción general,

Cifrar archivos PDF

guardar resultados, Paso 2: Abrir cada PDF

crear, Descifrar PDFdescifrar, Extraer texto de PDFdescifrar, Superponer páginasextraer texto de, Documentos PDFversión general del formato, Trabajar con PDF y documentos wordpáginas en, Crear PDF, Copiar páginas, Superponer páginas

copiar, Crear PDF superponer,

Superponer páginas

rotar, copiar páginas
Objetos PdfFileWriter, La función (), La función setdefault()PDFspformat Método, guardar variables con el módulo shelve

Visión general, El método setdefault(), Guardar variables con el módulo shelve

guardando variables en archivos de texto usando, Guardando Variables con el módulo shelve números de teléfono, extrayendo, Combinando re.IGNORECASE, re.DOTALL, y Combinando re.IGNORECASE, re.DOTALL, y re.VERBOSE, re.VERBOSE, Proyecto: de números de teléfono y direcciones de correo electrónicoExtractor, Paso 2: Crear una Regex para direcciones de correo electrónico, Paso 3: Buscar todas las coincidencias en el texto del portapapeles

crear regex, Proyecto: Extractor de Números de Teléfono y Direcciones de Correo

Electrónico encontrando coincidencias en el portapapeles, Paso 2: Crear una Regex

para Direcciones de Correo Electrónico

unir coincidencias en una cadena, Paso 3: Buscar todas las coincidencias en el texto del portapapeles

Visión general, Combinación de re.IGNORECASE, re.DOTALL y re.VERBOSEla almohadilla , Fundamentos de la imagen por ordenador, Coordenadas y tuplas de caja, Trabajar con el tipo de datos de imagen Trabajar con el tipo de datos de imagen,, , Copiar y pegarCopiar y pegar imágenes en otras imágenes Imágenes en otras imágenes, Copiar y pegar imágenes en otras imágenes, Girar y voltear imágenes, Girar y voltear imágenes, Cambiar píxeles individuales, Ideas para programas similares, Ideas para similaresprogramas, Ideas para programas similares, Líneas, Líneas, Líneas, Ejemplo de dibujo

copiar y pegar en imágenes, Copiar y pegar imágenes en otras imágenes

Recortar imágenes, Trabajar con el tipo de datos de imagen

dibujo sobre imágenes, similaresIdeas para programas , Ideas para programas similares, Ideas para programas similares, Líneas, Líneas, Líneas, Ejemplo de dibujo

elipses, Líneas

programa de ejemplo, Líneas

Módulo ImageDraw, Ideas para similares Ideas para programas similareslíneas de programas ,

puntos, Ideas para programas similares

polígonos, Líneas

rectángulos, Líneas

texto, Ejemplo de dibujo

voltear imágenes, Rotar y voltear imagenimágenesatributos de , Trabajar con el tipo de datos de imagenmódulo, Fundamentos de la imagen por ordenadorabrir imágenes, Coordenadas y de píxelestuplascajamanipulación de , Cambiar píxeles individualesdimensionar imágenes, Copiar y pegar imágenes en otras imágenesrotar imágenes, Rotación y volteo de imágenespíxeles transparentes, Copiar y pegar imágenes en otras imágenescarácter pip (|), Agrupación con paréntesis, Gestión de expresiones Regexes complejasherramienta pip, Instalación de Datapíxelesmódulos de tercerosfunción pixelMatchesColor(), Análisis de la captura de pantalla, , Paso 3. Empezar a escribir Paso 4: Empezar a escribir Datapixels, Fundamentos de la imagen por ordenador, Cambiar píxeles individualesarchivos plaintext, Encontrar tamaños de archivos y contenidos de más carpetassigno de (+), Coincidencia opcional con el signo de interrogación, Coincidencia de nuevas líneas con el carácter de puntoformato PNG, Trabajar con el de datos de imagenmétodo Typepoint() , Ideas para programas similaresmétodo Poll(), Ejecutar otros programas desde Pythonmétodo Polygon(), Función LinesPopen(), Paso 2: Crear e iniciar hilos, Ejecutar otros programas desde Python,
Programador de tareas, launchd y cron

abrir archivos con aplicaciones predeterminadas, Programador de tareas, launchd y cron pasar argumentos de línea de comandos a, Iniciar otros programas desde

Python
Función position(), Mover el ratón, Paso 3: Obtener e imprimir las coordenadas del ratónFunción print(), El método setdefault()Precedencia de los operadores matemáticos, Introducir expresiones en la shell pressinteractivaFunción(), Pulsar y soltar el teclado, Repaso de las funciones de PyAutoGUI, Paso 4: Manejar listas de selección y botones de printradioFunción (), Comentarios, La función input(), Argumentos de palabra clave e print(), Usar el módulo de registro, Paso 3: Empezar a escribir datos

logging y, Uso del módulo logging Visión general,

Comentarios

pasar varios argumentos a, Argumentos de palabra clave y print()

uso de variables con, La función input()
procesos, Paso :e iniciar Paso 2: Crear e iniciar 2 Crear hilos, hilos, Lanzamiento de otros programas desde Lanzamiento de otros Pythonprogramas desde Python, , Lanzamiento de otros programas desde Python, Programador de tareas, launchd y cron, Programador de tareas, launchd y cron, Proyecto:
Programa de cuenta atrás simple

y proyecto de cuenta atrás, Proyecto: Programa de cuenta atrás simple

definido, Paso 2: Crear e iniciar hilos

abrir archivos con aplicaciones predeterminadas, Programador de tareas, launchd

cron Programador de tareas, launchd y crony abrir sitios web,

pasar argumentos de línea de comandos a, Ejecutar otros programas desde Python

método poll(), Lanzamiento de otros programas desde la de

Pythonfunción Popen() , Paso 2: Crear e iniciar subprocesos

método wait(), Lanzamiento de otros programas desde Python
Perfilar código, El tiempo Móduloprogramación, Convenciones, ¿Qué es Python?, ¿Qué es Python?, Programar es una actividad creativa, Comentarios, Mezclar operadores booleanos y de comparación, Bloques de código, Importar módulos, Funciones, Ámbito local y global, locales y globalesVariables

con el Mismo Nombre, La Sentencia global La Sentencia global,, Programa de Ejemplo: Bola 8 Mágica con una Lista

bloques de código, Mezcla de operadores booleanos y de comparación

comentarios, Comentarios

creatividad necesaria para, La programación es una actividad creativa deduplicar el código, Funciones

definidos, Convenios

manejo de excepciones, La global

Statementejecución , programa, Bloques de código

funciones como "cajas negras", La declaración global

ámbito global, Variables locales y globales con el mismo nombre sangría,

Programa de ejemplo: Bola 8 Mágica con una Lista

Ámbito local, Matemáticas de ámbito

local y global y, ¿Qué es Python?

Python, ¿Qué es Python?

terminando el programa con sys.exit(), Importando Módulos

proyectos, Copiar y pegar cadenas con el módulo pyperclip, Proyecto: Añadir Viñetas al Wiki Markup, Combinando re.IGNORECASE, re.DOTALL, y re.VERBOSE, Guardando Variables con la Función pprint.pformat(), Paso Paso 3: Abrir navegadores web para cada resultado, Obtener 4: Escribir contenido en los archivos de preguntas y respuestas, Crear y añadir a archivos ZIP, Paso 3: Formar el nuevo nombre de archivo y renombrar los archivos, Web Scraping, Obtener datos de de un elementolos atributos , filas y columnas de las hojas, Escribir valores en celdas, Cifrar PDFs, Los argumentos de palabra clave delimiter y lineterminator, Leer JSON con la función loads(), La función time.sleep(), Proyecto: Multithreaded XKCD Downloader, Proyecto: Programa simple de cuenta atrásDesconexión , del servidor IMAP, Proyecto: Módulo "Just Text Me", Proyecto: Añadiendo un Logo, Moviendo el Ratón, , Revisión de las Funciones de PyAutoGUIAnalizando la Captura de Pantalla

Añadir viñetas al Wiki Markup, Proyecto: Añadir viñetas a Wiki Markup Añadir un logotipo, Proyecto: Añadir un logotipo

Relleno automático de formularios, revisión de las funciones de PyAutoGUI

Copia de seguridad de una carpeta en un archivo ZIP, paso 3: formar el nuevo nombre de archivo y cambiar el nombre de los archivos

Combinar páginas seleccionadas de varios archivos PDF, Cifrar archivos PDF

Descarga de todos los cómics XKCD, paso 3: abrir los navegadores web para cada resultado

Ampliación del programa mouseNow, análisis de la captura de pantalla

Obtención de datos meteorológicos actuales, lectura de JSON con la función loads()

Generación de archivos de prueba aleatorios, guardar variables con pprint.pformat() Función

Búsqueda en Google "Me siento afortunado", obtención de datos a partir de los atributos de un elemento

Módulo "Just Text Me", Proyecto: Módulo "Just Text Me"

mapIt.py con el Módulo webbrowser, Web Scraping

Multiclipboard, Paso 4: Escribir el contenido de los archivos del cuestionario y de la clave de respuestas

Multithreaded XKCD Downloader, Proyecto: Multithreaded XKCD Downloader

Password Locker, copiar y pegar cadenas con el módulo pyperclip

Extractor de números de teléfono y direcciones de correo electrónico, combinando re.IGNORECASE, re.DOTALL y re.VERBOSE

Lectura de datos de una hoja de cálculo, obtención de filas y columnas de las hojas

Eliminación de la cabecera de los archivos CSV, El delimitador y el lineterminador

Argumentos de las palabras clave

Renombrar archivos con fechas de estilo americano a fechas de estilo europeo, crear y añadir a archivos ZIP

Envío de correos electrónicos de recordatorio de cuotas, desconexión del servidor IMAP

Programa Cuenta Atrás Simple, Proyecto: Programa Cuenta Atrás Simple

Super Cronómetro, La Función time.sleep()

Actualización de una hoja de cálculo, escritura de valores en celdas

"¿Dónde está el ratón Mover el ratónahora mismo?", putpixel() method, Cambiar píxeles individualespyautogui.click() function, Proyecto: Rellenado automático de formulariosmétodopyautogui.click(), Pulsar el ratónfunciónpyautogui.doubleClick(), Pulsar el ratónfunciónpyautogui.dragTo(), Pulsar el ratónexcepciónpyautogui.FailSafeException, Pausas y FailSafesfunciónpyautogui.hotkey(), Pulsar y soltar el .keyDown(), tecladofunciónpyautoguiNombres de las teclasfunciónpyautogui.keyUp(), Nombres de las teclasmódulopyautogui.middleClick(), Pulsando el Ratónpyautogui module, Controlando Teclado y el Ratón el con Automatización Controlando el Teclado y el Ratón con Automatización GUIGUI, , Pausas y FailSafes, Pausas Safesy FailPausas y FailSafes, , Moviendo el Ratón, Paso 3: Obtener e Imprimir las Coordenadas del Ratón, Pulsando el Ratón, Pulsando el Ratón, Arrastrando el Ratón, Desplazando el Ratón, Proyecto: Extendiendo el Programa mouseNow, Reconocimiento de Imágenes, Reconocimiento de , ImágenesEnviando una Cadena desde el Teclado, Nombres de las Teclas, Presionando y Soltando el Teclado, Revisión de , las Funciones de PyAutoGUIRevisión de las Funciones de PyAutoGUI

proyecto de relleno de formularios, Revisión de las funciones de PyAutoGUI

teclado controlador, Reconocimiento de imágenes Reconocimiento de imágenes,, Envío de una cadena desde el teclado, Nombres de las teclas, Pulsar y soltar el teclado

combinaciones de teclas de acceso rápido, Pulsar y soltar el teclado

nombres de las Nombres de las teclasteclas, Envío de una

cadena desde el teclado pulsando y soltando teclas,

enviar cadena desde el teclado, Reconocimiento de imágenes
Controlar el ratón, Pausas y FailSafes, Paso 3: Obtener e imprimir las coordenadas del ratón, Hacer clic con el ratón, Hacer clic con el ratón, Arrastrar el ratón

hacer clic con el ratón, hacer clic con el

ratón arrastrar el ratón, hacer clic con el

ratón

Desplazamiento del ratón, Arrastrar el ratón
documentación para, Controlando el Teclado y el Ratón con Automatización GUIfunción a prueba de fallos, Pausas y FailSafesfunciones, Revisión de las Funciones de PyAutoGUIreconocimiento de imágenes, Proyecto: Extendiendo el Programa mouseNowimportando, Moviendo el Ratóninstalando, Controlando el Teclado y el Ratón con la Automatización GUIllamadas a funciones de pausa, Pausas y FailSafescapturas de pantalla, Desplazando el .mouseDown(Ratónfunciónpyautogui), Haciendo click con el Ratónfunciónpyautogui.moveRel(), Controlando del Ratónel Movimiento , Moviendo el Ratónfunciónpyautogui.moveTo(), Controlandoel Movimiento Ratónvariablepyautogui.PAUSE variable, Pausas y del FailSafespyautogui.position() function, 3: Obtener e Paso Coordenadas del Ratónpyautogui.press() function, Paso 4: Manejar Listas de Selección y Imprimir las Botones de Radiopyautogui.rightClick(), Hacer clic con el ratónfunciónpyautogui.screenshot(), Desplazar el ratónfunciónpyautogui.size(), Controlar el movimiento del ratónfunciónpyautogui.typewrite(), Reconocimiento de imágenes, Enviar una cadena desde el teclado, Proyecto: Automatic Form Fillerpy.exe program, Shebang Linepyobjc module, The pip ToolPyPDF2 module, Working with PDF and word Documents, PDFDocumentos , Extracción de texto de PDFs, Descifrado de PDFs, Creación de PDFs, Copia de páginas, Superposición de páginas, Superposición de páginas, Cifrado de PDFs

combinar páginas de varios PDF, cifrar PDF crear PDF, descifrar

PDF

descifrar PDF, extraer texto de PDF cifrar PDF,

superponer páginas

extraer texto de PDFs, Documentos PDF

Visión general del formato, Trabajar con PDF y documentos word

páginas en PDF, Crear PDF, Copiar páginas, Superponer páginas

copiando, Crear PDF

superposición, superposición de páginas

rotar, copiar páginas
Módulo pyperclip, Eliminación de espacios en blanco con strip(), rstrip() y lstrip()Python, Convenciones, ¿Qué es Python?, ¿Qué es Python?, Acerca de ,este libro Acerca de este libro, Descarga e instalación de Python, Descarga e instalación de Python, Inicio IDLEde Inicio de IDLE, ,
Introducción de expresiones en el intérprete de comandos

interactivoIntroducción de expresiones en la intérprete de comandos

interactivo, Tipos de datos de nombres de variables, descarga del ,

Acerca de este libro

programa de ejemplo, Ayuda para

nombres de variables, Inicio de IDLE

instalación, Acerca de este libro shell

interactivo, Inicio de IDLE

intérprete, definido, Descargar e instalar Python matemáticas y,

¿Qué es Python?

visión general, ¿Qué es Python? visión

general de la programación,

Convenciones

inicio de IDLE, descarga e instalación de Python
módulo python-docx, Paso 4: Guardar los python-docxresultadosmódulo ,
Desconexión del servidor SMTP, Obtener un correo electrónico marcarlo
leídoObjetos python-docxy como Obtenerun correo electrónico y marcarlo como leído,

Q

signo de interrogación (?), Coincidencia varios grupos con la tuberíade , Coincidencia de nuevas líneas con el carácter de punto

método quit(), Envío de claves especiales, Desconexión del servidor SMTP, Paso 3: Enviar recordatorios personalizados por correo electrónico

quiz generator, Guardando Guardando Variables (con la Función pprint.pformatVariables con la Función pprint.pformat()), , Paso 1: Almacenar del Quiz los Datos en un Diccionario, Paso 2: Crear el Quiz Crear el Archivo del y Barajar el Orden de las Preguntas, Paso 2: Crear el Archivo del Quiz y Barajar el Orden de las Preguntas, Paso 3: Archivo del Quiz y Barajar el Orden de las Preguntas: Crear las opciones de respuestaPaso 3: Crear las opciones de respuesta,

crear el archivo quiz, Paso 2: Crear el archivo quiz y barajar el orden de las preguntas

Creación de opciones de respuesta, Paso 3: Creación del opciones de respuestaresumen de , Almacenamiento de variables con la función pprint.pformat()

barajar el orden de las preguntas, Paso 2: Crear el archivo del cuestionario y barajar el orden de las preguntas

almacenar los datos del cuestionario en un diccionario, Paso 1: Almacenar los datos del cuestionario en un diccionario escribir el contenido en archivos, Paso 3: Crear las opciones de respuesta

R

botones de opción, Paso 3: Empezar a introducir datos

método raise_for_status(), Descarga de una página web con el método requests.get() Función

palabra clave raise, Depuración

función range(), un bucle while equivalente

cadenas en bruto, Caracteres de escape, Creación de objetos Regex Objetos Reader Objetos Reader,

lectura de archivos, apertura de archivos con la función open(), compresión de archivos con el módulo zipfile

Método readlines(), Abrir la función openarchivos con Abrir archivos

con la función open() () Método read(), Método rectangle(), Líneas

Reddit, Cómo encontrar

ayudaobjetos de referencia de ,

Gráficos

referencias, El tipo de datos tuplaEl tipo de datos tupla, Pasar

referencias visión general,

Pasar, Pasar Referencias

método refresh(), Envío de claves RegexespecialesObjetos , Búsqueda de patrones de texto sin expresiones regulares, Creación de objetos Regex

crear, encontrar patrones de texto sin expresiones regulares

matching, Creación de objetos Regex

Expresiones Regulares, Coincidencia de Patrones con Expresiones Regulares, Coincidencia con Expresiones Regulares, Búsqueda de Patrones de Texto sin de Patrones Expresiones RegularesBúsqueda de Patrones de Texto sin , Expresiones Regulares, Creación de Objetos Regex, Revisión de la RegularesCoincidencia de , ExpresionesAgrupación con ParéntesisCoincidencia de Múltiples Grupos , Tubocon el , Coincidencia Opcional con el Signo de Interrogación, Coincidencia Opcional con el Signo de Interrogación, Coincidencia de uno o más con el signo másCoincidencia de uno , más con el signo más, o Coincidencia , Coincidencia , codiciosa y no codiciosacodiciosa y no codiciosaEl método findall(), Clases de caracteres, Clases de caracteres, Los caracteres Caret y Dollar Sign, Coincidencia de nuevas líneas con el carácter punto, Coincidencia sensible a mayúsculas y minúsculas, Coincidencia sensible a mayúsculas y minúsculas, Manejo de expresiones regulares complejas, Manejo de expresiones regulares complejas, Combinación de re.IGNORECASE, re.DOTALL y re.VERBOSE, Cómo abrir las herramientas de desarrollo del navegador

inicio de las coincidencias de cadena, Clases de

caracteres sensibilidad a mayúsculas , Coincidencia

sensible a mayúsculas y minúsculasy minúsculas

clases de caracteres, El método findall()

Creación de objetos Regex, Búsqueda de patrones de texto sin expresiones regulares

definidas, Coincidencia de patrones con expresiones regulares

coincidencias de final de cadena, Clases de caracteres

extraer números de teléfono y direcciones de correo electrónico, Combinando re.IGNORECASE, re.DOTALL, y re.VERBOSE

método findall(), concordancia Greedy y Nongreedy

encontrar texto sin, Coincidencia de patrones con expresiones regulares

coincidencia codiciosa, Coincidencia de uno o más con el signo más

agrupación, Revisión de la coincidencia de expresiones regulares, Agrupación con paréntesis, Coincidencia de varios grupos con la tubería, Coincidencia con el signo ,opcional de interrogación Coincidencia opcional con el signo de interrogación, Combinar uno o más con el Plus

Coincidencia de uno o más con el signo más de repeticiones específicas,

Coincidencia una o más coincidencias, Coincidencia opcional con el signo de

interrogación Coincidencia opcional, Coincidencia de varios grupos con el

signo pipa

utilizando paréntesis, Revisión de la concordancia de expresiones

regulares utilizando el carácter pipa en, Agrupación con paréntesis

cero o más coincidencias, Coincidencias opcionales con el signo de interrogación HTML y, Apertura de las herramientas de desarrollo del navegadorcoincidencias con, Creación de objetos regexmúltiples argumentos para la función (), compileGestión de regexes complejascoincidencias no regreedy, Greedy y Nongreedy Coincidencia de patrones para, Búsqueda de patrones de texto sin expresiones regularesextensión a varias líneas, Gestión de expresiones regulares complejassustitución de cadenas mediante, Coincidencia sensible a mayúsculas y minúsculasreferencia a símbolos, Coincidencia de nuevas líneas con el carácter comodínpuntocarácter , Los signo de intercalación y signo de relativascaracteres dólarrutas , El trabajo actual

Directoryrelpath(), El módulo os.path Module, Manejo de y americanoamericanoestilo americanorutas relativasoperador remainder/modulus (%), Introducción de expresiones fechas de estilo fechas de estilo fechas de absolutasen la shell interactiva, El truco de las asignaciones múltiplesmétodo remove(), una una Adición de valores a listas con los métodos append() e insert()método remove_sheet(), para Creación y eliminación de hojasrenombrar Regex para fechas de estilo americanoRegex Regex Regex archivos/carpetas, Copia de archivos y carpetas, Creación y adición a archivos ZIP, Creación y adición a archivos ZIPCrear , Paso , Paso 3, Paso , Paso , Paso , Paso 7, Paso 14568: Crear una Regex para fechas de estilo americano: Formar el nuevo nombre de archivo y renombrar los archivos: Crear una Regex para fechas de estilo americano: Crear : : Crear una para : Crear una para : Crear una Regex para Fechas de Estilo Americano, Paso 3: Formar el Nuevo Nombre de Archivo y Renombrar los Archivos

estilos de fecha, y añadir a ZIPañadir a ZIPCrear archivos , Crear y , Crear y añadir a archivos ZIP, Paso 1: Crear una Regex para archivos fechas de estilo americano, Paso 3: Formar el nuevo nombre de archivo y renombrar los archivos

crear regex para fechas, Crear y añadir a archivos ZIP

identificar fechas en nombres de archivo, Paso 1: Crear una Regex para fechas de estilo americano

visión general, creación y adición a archivos ZIP

renombrar archivos, Paso 3: Formar el nuevo nombre de archivo y renombrar los archivos replicación, Concatenación de cadenas y replicación, Obtener la longitud de una lista con len()

de listas, Obtención de la longitud de una lista con len()

cadena, concatenación de cadenas y replicación
módulo requests, Descarga de archivos de la Web con el módulo requests, Guardar archivos descargados en el disco duro

descargar archivos, guardar archivos descargados en el disco duro

descarga de páginas, Descarga de archivos de la Web con el módulo de solicitudes
resolución de la pantalla del ordenador, Control del movimiento del ratónRespuesta objetos, Descarga de archivos de la Web con el módulo requestsereturn valores, función, def Declaraciones con parámetrospalabra clavereverse, Eliminación de valores de listas con remove()Valores RGBA, Fundamentos de la imagen por ordenadorModelo de color RGB, Colores y valores RGBArightClick()

hacer clic con el ratón, Revisión de las funciones de rjustPyAutoGUImétodo, ()Métodos join() y split(), Paso 3: Obtener e imprimir las coordenadas ratónfunción rmdir(), Mover y renombrar archivos y del carpetasfunciónrmtree(), Mover y renombrar archivos y carpetasmétodo rotateClockwise(), Copiar rotateCounterClockwisepáginasmétodo, ()Copiar páginasrotating images, Rotar y voltear imágenesrounding numbers, La función time.sleep() Functionrows, en hojas de cálculo Excel, Conversión entre letras de columna y números, Fórmulas

ajuste de la altura y la anchura de, Fórmulas

Rebanar objetos de la hoja de cálculo para obtener objetos de celda, Conversión entre letras y números de columna

Método rstrip(), Justificar texto con rjust(), ljust() y center()rtl atributo, Ejecutar atributosEjecutar objetos, Estilizar párrafo y ejecutar objetos, Ejecutar atributosejecutar programas, Ejecutar programas, Ejecutar programas, Línea Shebang, Ejecutar programas Python y Linuxen OS X , Ejecutar programas Python en OS X y Linux

en Linux, Ejecutar programas Python en OS X y Linux Ejecutar programas Python en OS X y Linux en OS X, visión general, Ejecutar programas

en Windows, Shebang Line shebang

line, Ejecutar programas S

\S clase de caracteres, El método findall()

\s character class, El método findall()

Directiva %S, Pausa hasta una fecha específica

Safari, herramientas para Cómo abrir el de las herramientas para desarrolladores del desarrolladores en, método save() navegador, Manipulación de imágenes con Pillow

scope, Ámbito local y global, Variables locales y globales con el mismo nombre

global, Variables locales y globales con el mismo nombre

Ámbito local y mundiallocal,
función screenshot(), Desplazamiento del ratón, Revisión de las funciones de PyAutoGUIscreenshots, Desplazamiento del ratón, Análisis de la captura de pantalla

analizar, Analizar la captura de pantalla

Obtener, Desplazar el ratón
scripts, Copiar y pegar cadenas con el módulo pyperclip, Programador de tareas, launchd y cron

ejecución desde programa Python, Programador de tareas, launchd y cron

Ejecución fuera de IDLE, Copiar y pegar cadenas con el módulo pyperclip
scroll(), Arrastrar el ratón, Desplazar el ratón, Revisión de la función Funciones de PyAutoGUIdesplazar el ratón, Arrastrar el ratónBuscar, Obtener datos de los atributos un elementode , los atributos Obtener datos de de un elemento, Paso 1: Obtener los argumentos de la línea de comandos y Paso 1: Obtener los argumentos de la línea de comandos y solicitar la página de búsqueda, solicitar la página de búsqueda, Paso 2: Buscar todos los resultados, Conectarse a un servidor IMAP

correo electrónico, Conexión a un servidor IMAP

la Web, un Obtener datos de de un Obtener datos de los atributos elementode , los atributos 1elemento, comandos 1: Obtener los argumentos de la línea de comandos Paso Obtener los argumentos de la línea de y Paso solicitar la página de búsquedaPaso 1: Obtener los argumentos de la línea de comandos y : Paso 2: Buscar todos los resultadossolicitar la página de búsqueda, , solicitar la página de búsqueda, y

búsqueda de resultados, Paso 1: Obtener los argumentos de la línea de comandos y solicitar la página de búsqueda

Obtención de argumentos de la línea de comandos, Paso 1: Obtención de los argumentos de la línea de comandos y solicitud de la página de búsqueda

abrir el navegador web para ver los Paso 2: Buscar todos los resultadosresultados,

visión general, Obtener datos de de un los atributos elemento

solicitar la página de búsqueda, Paso 1: Obtener los argumentos de la línea de comandos y solicitar la página de búsqueda
search(), Creación de objetos RegexSEEN clave de búsqueda, Ejecución del método Programa Searchsee, Programador de tareas, launchd, y cronselect_folder()método , Seleccionar una carpetaelect Paso páginasmódulo selenium, Paso 4: listas, 3: Empezar a escribirDataselect() método, módulo bs4, Crear HTMLselectores, CSS, un objeto BeautifulSoup Crear un objeto BeautifulSoup a partir de a partir de HTML, Encontrar elementos en las Guardar la Imagen y Encontrar el Cómic Anterior, Paso 4: Guardar la Imagen y Encontrar el AnteriorCómic , Iniciando un Navegador Controlado por Selenium, Encontrando en Elementos , Encontrando Elementos en la Página, la Páginay Enviando Formularios,
Envío de claves especiales

pulsar botones, Enviar teclas especiales

encontrar elementos, Iniciar un navegador controlado por

Selenium siguiendo enlaces, Encontrar elementos en la página

instalar, Paso 4: Guardar la imagen y buscar el cómic anterior enviar

pulsaciones especiales, Rellenar y enviar formularios enviar formularios,

Buscar elementos en la página

usando Firefox con, Paso 4: Guardar la imagen y encontrar el de Previous Comicmódulo send2trash , Eliminar permanentemente archivos y carpetasenviar correos electrónicos de recordatorio, Desconectarse del IMAPservidor , servidor IMAPDesconectarse , Desconectarse del servidor IMAP, Paso 2: Buscar todo Miembros impagados, Paso 2: Encontrar todos los miembros

impagados Paso 2: Encontrar todos los miembros impagados

encontrar miembros impagados, abrir archivo Excel, Desconectarse

del servidor IMAP Desconectarse del servidor IMAPdescripción

general,

envío de correos electrónicos, Paso 2: Buscar a todos los socios impagados
método send_keys(), Encontrar elementos en las páginasmétodo send_keys(), Iniciar sesión en el servidor SMTP, Paso 3: Enviar correo electrónico personalizado

Recordatoriosnúmeros de secuencia, Obtención de un correo electrónico y marcado como lecturaSecuencias, Uso de bucles for con el método Listssetdefault(), El método setdefault()Atributo shadow, Atributos de ejecuciónlínea shebang, Ejecución de programasmóduloelve, Escritura en archivosServicio de mensajes cortos (SMS), Envío de mensajes de texto con Twilio, Envío de mensajes de texto

envío de mensajes, Envío de mensajes de texto

Servicio Twilio, Envío de mensajes de texto con Twilio

Módulo shutil, Copiar archivos carpetasy Copiar archivos y carpetas, , Mover y renombrar archivos y carpetas

eliminar archivos/carpetas, mover y renombrar archivos y carpetas mover

archivos/carpetas, copiar archivos y carpetas

renombrar archivos/carpetas, Copiar archivos y carpetas

SID (ID de cadena), Envío de mensajes de textoProtocolo simple de transferencia de correo, Conexión a un servidor SMTP (ver SMTP (Simple Mail Transfer Protocol))Tecla de búsqueda SINCE, Selección de una carpetaUna comilla ('), Literales de cadenaProgramas de un solo hilo, size(), Control de los movimientos del ratónFunción sleep(), La hora.time(), Pausa hasta una fecha específica, tareasRevisión de las funciones de tiempo de Python, Programador de , launchd y cronslices, Índices negativos, Cadenas multilínea con comillas triples

obtener sublistas con, Índices negativos

para cadenas, cadenas de varias líneas con comillas triples

Atributo small_caps, Ejecutar SMALLERatributosClave de búsqueda , Realización de la búsquedaSMS (Short Message Service), Envíomensajes de texto con Twilio, de Envío de mensajes de texto

enviar mensajes, Enviar mensajes de texto

Servicio Twilio, Envío de mensajes de texto con Twilio

SMTP (Protocolo simple de transferencia de correo), SMTP, Conexión a un SMTPservidor , SMTPConexión a servidor , servidor SMTPun un Inicio de sesión en el servidor SMTPConexión a Conexión a un servidor SMTP, , , Desconexión del servidor SMTP

Conectarse a un servidorConectarse a un servidor SMTP,

definido, SMTP

desconectarse del servidor, Desconectarse del servidor SMTP iniciar sesión

en el servidor, Conectarse a un servidor SMTP

envío de mensaje "hola", Conexión a un servidor SMTP envío de

mensaje, Inicio de sesión en el servidor SMTP

Encriptación TLS, Conexión a un servidor SMTP

Objetos SMTP, Conexión a un servidor SMTPmétodo sort(), Eliminación de valores de listas con remove()archivos de sonido, reproducción, Proyecto: Simple Countdown Programscódigo fuente, definido, Convencionesmétodo, Los split()métodos de cadena isX, Manejo de rutas absolutas y relativas, Trabajar con archivos CSV y hojas de datos JSONhojas de cálculo, Trabajar con hojas de cálculo Excel (ver hojas de cálculo Excel)corchetes [], El de datos de pilatipolistaDesbordamiento de , Cómo encontrar estándarayudabiblioteca , Importación de módulosestrella (*), Coincidencia opcional con el signo de interrogación, El carácter comodín, El carácter comodín, Coincidencia de nuevas líneas con el carácter punto

utilizando con carácter comodín, El carácter comodín

cero o más coincidencias con, Coincidencias opcionales con el signo de interrogación método start(), Multihilo, Pasar argumentos a la función destino del hilo, Paso 1: Modificar el programa para utilizar una funciónprograma start, Programador de tareas, launchd y cronstartswithmétodo (), La cadena isX Methodsmétodostarttls(), Conectarse a un servidor SMTP, Paso 3: Enviar recordatorios personalizados por correo electrónicoargumento stepp, Un while equivalenteproyecto Loopstopwatch , La función time.sleep()La time.sleep, función (), Proyecto: Super Cronómetro, Proyecto: Super Cronómetro

resumen, La función time.sleep() puesta

en marcha, Proyecto: Super Cronómetro

seguimiento de tiempos de vuelta, Proyecto: Super Cronómetro

Función strftime(), Pausa hasta una fecha específica, Revisión de las temporales de funciones PythonFunción strftime(), La función len(), El tipo de datos tupla, Paso 3: Obtener e imprimir las coordenadas del ratónAtributo strike, Atributos de ejecuciónCadena ID (SID), Envío de mensajes de textoCadenas, Los enteros, coma flotante y

Tipos de datos de cadenas, tipos de datos de enteros, de coma flotante y de cadenas, concatenación y replicación de cadenas, tipos de lista: Cadenas y tuplas, Literales de cadena, Literales de cadena, , Literales de cadenaCaracteres de escape, escapeCaracteres de , Cadenas de varias líneas con comillas triples, Cadenas de varias líneas con comillas triples, Indexación y división de cadenas, Indexación y división de cadenas, Métodos de cadena , lower(), isupper() e islower() Métodos de cadena upper(),upper(), lower)()and islower() String Methods, The (), lower(), (, upperupper(The upper(), lower(), isupper(), and islower() String isupper(String The upper(), lower(), isupper(), and islower() String , and islower() String , The upper(), lower(), isupper(), and islowerisupper(), and islower() String The), lower(), isupperMethods, MethodsMethods, Methods, (),) Methods, Los métodos métodos , Los métodos join() y split() String, Los métodos join() y split() String, Justificar texto con rjust(), ljust() y isX String, Los métodos isX String, , Los métodos isX String, Los métodos isX String, Los isX String, Los Los StringisX métodos métodos join() y split() String, Los métodos join() y split() Stringcentercon rjust, con rjust(), ljust() y center(), Eliminación de espacios en blanco con strip(), (), () y lstrip(), Justificación de texto (), ljust() y center()Justificación de texto rstripCoincidencia sensible a mayúsculas y minúsculas, Salto de excepciones, PDFDocumentos , Pausa hasta una fecha específica, Conversión de objetos datetime en cadenas de texto

método center(), Los métodos join() y split() cadenasconcatenación de , Los

tipos de datos entero, coma flotante y cadena conversión de objetos

datetime a, Pausa hasta una fecha específica

conversión a objetos datetimeConversión de objetos datetime en cadenas,

copiar y pegar, Eliminar espacios en blanco con strip(), rstrip() y lstrip()

comillas dobles para, El , Los los métodos de

cadena isXmétodo endswith() de los literales de

cadenacaracteres de escape de , Los literales de

cadena extraen el texto PDF como, Los

documentos PDF obtienen el rastreo como,

Planteamiento de excepciones

índices para, Cadenas de varias líneas con comillas triples en el operador, Indexación y segmentación de cadenas

Método isalnum(), Métodos de cadena upper(), lower(), isupper() e islower()

Método isalpha(), Métodos de cadena upper(), lower(), isupper() e islower()

Método isdecimal(), Métodos de cadena upper(), lower(), isupper() e islower()

Método islower(), Métodos de cadena upper(), lower(), isupper() e islower()

Método isspace(), Métodos de cadena isX Método istitle(), Métodos de cadena isX

Método isupper(), Los métodos de cadena upper(), lower(), isupper() e islower()

método join(), La cadena isX Métodos

literales, Literales de cadena

Método ljust(), Los métodos de cadena join() y split()

método lower(), Los métodos de cadena upper(), lower(), isupper() e islower()

Método lstrip(), Justificar texto con rjust(), ljust() y center() multilínea, Caracteres de escape

Tipos de datos mutables vs. inmutables, Tipos tipo lista: Cadenas y tuplas no en operador, Indexación y segmentación de cadenas

resumen, Tipos de datos enteros, de coma flotante y de cadena raw, Caracteres de escape

replicación de, Método String Concatenation and Replication

rjust(), Métodos String join() y split()

Método rstrip(), Justificar texto con rjust(), ljust() y center() slicing, Cadenas multilínea con comillas triples

método split(), Los métodos isX String Los métodos isX Stringmétodo startswith(),

método strip(), Justificación de texto con rjust(), ljust() y center() sustitución mediante expresiones regulares, Coincidencia sensible a mayúsculas y minúsculas

método upper(), Los métodos de cadena upper(), lower(), isupper() e islower()
método strip(), Justificar texto con rjust(), ljust() y center()strptime()
Convertir objetos de fecha y hora en cadenas, Revisión de funciones de tiempo de Python, las Tipos de datos enteros, de coma flotante y de cadena

(véase también cadenas)
Objetos de estilo, Configuración del estilo de fuente de las SUJETOceldasTecla de búsqueda Selección de , pasos una carpetaListas, obtener con rebanadas, Índices negativosMétodo SubSub(), mayúsculas y Coincidencia sensible a minúsculasVariable de color del botón SubmitButtonColor, Paso 1: Averiguar los Método ssubmit(), Rellenado y envío de ssubprocessformulariosMódulo, Control del tiempo, programación de tareas y ejecución de programas, Paso 2: Creación e inicio de hilosOperador de sustracción (-), Introducción de expresiones en el intérprete de comandos interactivo, Trucos de asignación color sustractivomúltipleModelo de, Colores y valores sudokuRGBAPuzles de , ¿Qué es Python?Función sys.exit(), Importación de módulos

T

atributo tag_name, Búsqueda de elementos en la página

Objetos etiqueta, Creación de un objeto BeautifulSoup a partir de etiquetas HTML, HTML, Guardar archivos descargados en el disco duro Programador de tareas, Ejecutar otros programas desde Python

terminación, programa, Su primer programa, Importar módulos

atributo texto, Leer documentos Word, Ejecutar atributos

mensajería de texto, Envío de mensajes con Twiliode texto Envío de mensajes de texto, , Proyecto "Módulo "Just Text Me

notificaciones automáticas, Proyecto: Módulo "Just Text Me"

envío de mensajes, Envío de mensajes de texto

Servicio Twilio, Envío de mensajes de texto con Twilio

método text(), Dibujar TEXTOejemploTecla de búsqueda , Seleccionar una carpetamétodo textsize(), Dibujar textomódulos de terceros, instalar, Instalar módulos de threadtercerosFunción (), Multithreading, Paso 1: Modificar el programa para utilizar una funciónmódulo threading, Controlar el tiempo, programar tareas e iniciar programas, MultithreadingObjetos, threadMultithreadingthreads, Multithreading, Multithreading, Pasar argumentos a la función de destino del thread, Proyecto: Multithreaded XKCD Downloader, Paso 2: Crear e Iniciar Threads, Paso 2: Crear e Iniciar Threads

problemas de concurrencia, Pasar argumentos al la función destino del hilométodo

join() de , Paso 2: Crear e iniciar hilos

multithreading, Multithreading, Proyecto: Multithreaded XKCD Downloader

descargador de imágenes, Proyecto: Descargador XKCD multihilo

pasar argumentos a, Multihiloprocesos vs., Paso 2: Crear e Iniciar HiloTablero de tres en raya, Usando Estructuras de Datos para Modelar del Mundo Cosas RealTipo de datos ,datetime El Módulo datetime, de Repaso de las Funciones de Tiempo Repaso de las Funciones de Tiempo de PythonPythonObjetos ,datetime El Módulo datetime, El timeMódulo , La Función time.time(), La Función time.sleep(), Pausando Hasta una Fecha Específica,

visión general, Revisión de las funciones temporales de Python

función sleep(), La función time.time(), Pausa hasta una fecha específica proyecto

cronómetro, La función time.sleep()

función time(), El módulo time
Encriptación TLS, Conectándose a un Servidor de nivel superiorSMTPDominios, Paso 2. Crear una Regex para Direcciones de Email Crear una Regex para direcciones de correo electrónicoTotal seconds(), El módulo datetime, Repaso de la función tiempo Pythonstraceback, salir de un error, lanzar de excepcionestransparenciapegar imágenes en otras imágenestranspose(), Rotación y , Fundamentos de la imagen por ordenador, Copiar y Volteo de Imágenescomillas triples ('''), Caracteres de Escape, Manejo de Regex complejasfunción struetype(), Dibujo de Textotablas de verdad, Operadores de Comparaciónvalores de "verdad", sentencias continuastipo , de datos tuplaEl tipo de datos tuplaTipos de datos mutables e inmutables,

Tipos de datos mutables e inmutables

Función tuple(), El tipo de datos tuple
módulo twilio, Envío TwilioTwilioRestClientde mensajes de texto con Envío de mensajes de texto con Twilioobjetos , Envío de mensajes de textoServicio Twilio, , Envío de mensajes de texto, Proyecto: "Just Text Me" Módulo

mensajes de texto automáticos, Proyecto: módulo "Just Text

Me"Visión general del , Envío de mensajes de texto con Twilio

enviar mensajes de textoEnviar mensajes de texto,
TypeError, tipo listaObtención de valores individuales en una lista con índices, Tipos : Strings y la función Tuplestypewrite(), Reconocimiento de Imágenes, de Envíouna Cadena desde el Teclado, Revisión de las Funciones de PyAutoGUI, Proyecto: Relleno Automático de Formularios, Paso 3: Empezar a Escribir Datos, Paso 4: Manejar Selección Listas dey Botones de Radio

U

Ubuntu, Descargar e Instalar Python, desde PythonEjecutar Otros Programas , Ejecutar Otros Programas desde Python, Programador de Tareas, launchd y cron, Abrir Archivos con Aplicaciones por Defecto

cron, lanzar otros programas desde python

lanzamiento de procesos desde Python, lanzamiento de otros programas desde

Python apertura de archivos con aplicaciones predeterminadas, programador de

tareas, launchd y cron

Filosofía Unix, Abrir archivos con aplicaciones predeterminadasTecla de búsqueda UNANSWERED, Realizar la variablesTecla de búsquedaUNDRAFT, Realizar la búsquedabúsquedaTecla de búsquedaUNDELETED, Realizar la subrayadobúsquedaAtributo de , Atributos de ejecuciónSubrayado (_), Nombres de UNFLAGGED, de búsqueda Realizar carpetasTecla de búsqueda , Realizar la la búsquedaMétodo upper(), Los upper(), lower(), isupper() e islower(búsquedaCodificaciones, Unicode Guardar archivos descargados en el disco UnixduroÉpoca , El módulo time, El módulo datetime, Revisión de las funciones de tiempo de UnixPythonFilosofía, Abrir archivos con aplicaciones unlinkpredeterminadasFunción(), Mover y renombrar archivos y UNSEENmétodos de cadena)UTC (Tiempo Universal Coordinado), El módulo time

V

ValueError, El truco de la asignación múltiple, Conversión de objetos datetime en cadenas de texto

valores, definidos, Introducción de expresiones en el intérprete de comandos interactivo, Búsqueda de patrones de texto sin expresiones regulares

método values(), Diccionarios vs. Listas

variables, Concatenación y replicación de cadenas, replicación de , Concatenación y cadenasSentencias , Eliminación de valores de listas con sentencias del, de asignación, Sentencias de asignación, variablesNombres de , Valores de retorno y sentencias return, Ámbito local y global, Variables locales y globales con el mismo nombreMétodos, El tipo de datos tupla, Escritura en archivos

(ver también listas)

sentencias de asignación, definición de concatenación y replicación de

cadenas concatenación y replicación de cadenas,

global, Variables locales y globales con el mismo nombre

inicialización, Sentencias de asignación

local, Nombres de Nombres de

variablesámbito local y global,

Valor nulo y, Valores de retorno y declaraciones de retorno

sobrescritura, Declaraciones de asignación

referencias, el tipo de datos tupla

guardar con el módulo shelve, escribir en archivos

Almacenamiento como lista, Eliminación de valores de listas con declaraciones del Correo de Verizon, Conexión a un servidor SMTP, Recuperación y eliminación de correos electrónicos con IMAPvolumes, definidos, Archivos y rutas de archivos

W

\Clase de caracteres W, Método findall()

\w clase de caracteres, El método findall()

Directiva %w, Pausa hasta una fecha específica

walk(), Borrados seguros con el módulo send2trash, Lanzamiento de otros programas desde Python

Nivel ADVERTENCIA, Uso del módulo de registro

datos meteorológicos, fetching, Lectura Lectura de de JSON (con la función loads), con la función loads(), JSON Paso 1: Obtener ubicación a partir del argumento de la línea de comandosPaso 1: Obtener ubicación a partir del argumento de la línea de comandos, ,
Paso 2: Descargar los datos JSON

Descarga de datos JSON, Paso 1: Obtención de la ubicación a partir del argumento de la línea de comandos

Obtención de la Paso 1: Obtención de la ubicación a partir del argumento de la línea de comandos ubicación, Carga de datos JSON, Paso 2: Descarga de los datos JSON

Visión general, Lectura de JSON con la función loads()
módulo webbrowser, Web Scraping, Programador de tareas, launchd y cron

Programador de tareas, launchd y cronfunción open(),

apertura del navegador mediante, Web Scraping

Objetos WebDriver, Iniciando un por Navegador ControladoSeleniumObjetos , WebElementIniciando un Navegador Controlado por Seleniumweb scraping, Web Scraping, Web Scraping, Descargando Archivos desde la Web con el Módulo requests, Descargando Archivos desde la Web con el Módulo , requestsGuardando Archivos Descargados en el , Guardando Disco Duroen el Disco Duro, Archivos Descargados Saving Downloaded Files to the Hard Drive, A Quick Refresher, Viewing the Source HTML of a Web Page, Using the Developer Tools to Find HTML Elements, Parsing HTML with the BeautifulSoup Module, Parsing HTML with the BeautifulSoup Module, Creating a BeautifulSoup Object from HTML, Getting Data from an Element's Attributes, Getting Data from an Element's Attributes, Step 3: Abrir Navegadores Web para Cada Resultado, Paso 4: Guardar la Imagen y Encontrar el Cómic Anterior, Paso 4: Guardar la Imagen y Encontrar el Cómic Anterior, Iniciar un Navegador Controlado por Selenium, Encontrar Elementos en la Página, , Encontrar Elementos en la PáginaRellenar y Enviar Formularios, Enviar Claves Especiales

Módulo bs4, Análisis sintáctico de Análisis sintáctico de HTML con el módulo BeautifulSoup, HTML con el módulo BeautifulSoup, Creación de un BeautifulSoup objetoa partir de HTML, Obtención de datos a partir de los atributos de un elemento

Creación de objetos a partir de HTML, Análisis de HTML con el módulo BeautifulSoup

encontrar un elemento con el método select(), Crear un objeto BeautifulSoup a partir de HTML

obtener atributo, Obtener datos de los atributos de un elemento

Visión general, Análisis de HTML con el módulo BeautifulSoup
descargar, Descargar archivos de la Web con el módulo de solicitudes, , Paso 3: Abrir los navegadores Web para cada resultadoGuardar los archivos descargados en el disco duro

archivos, Guardar los archivos descargados en el disco duro

imágenes, Paso 3: Abrir los navegadores web para cada

resultado

descargar archivos de la Web con el módulo de solicitudes
y Google Maps, Web Scraping y Google Search, Obtención de datos de los atributos de un elementoHTML, Almacenamiento de archivos descargados en el archivo

Disco , DuroDisco DuroGuardar Archivos Descargados en el , Repaso , Un RápidoVer el HTML Fuente de una Página Web, Usar las Herramientas de Desarrollador para Encontrar Elementos HTML

herramientas de desarrollo del navegador y, Visualización del HTML fuente de una página web búsqueda de elementos, Uso de las herramientas de desarrollo para buscar elementos HTML recursos de aprendizaje, Guardar archivos en el disco duro descargados Guardar archivos descargados en el disco durodescripción general,

ver página fuente, Un rápido repaso
descripción general, Módulo , Descarga de archivos de la Web Web Scrapingrequestscon el módulo requestsel módulo elenium botones de clic, Paso 4: Guardar la imagen y encontrar el cómic anterior, Paso 4: Guardar la imagen y encontrar el cómic anterior, Inicio de un navegador controlado por Selenium, Búsqueda de elementos en la página, Búsqueda de elementos en la página, Rellenado y envío de formularios, Envío de claves especiales

encontrar elementos, Iniciar un navegador controlado por Selenium siguiendo enlaces, Encontrar elementos en la página

instalar, Paso 4: Guardar la imagen y buscar el cómic anterior enviar

pulsaciones especiales, Rellenar y enviar formularios enviar formularios,

Buscar elementos en la página

usando Firefox con, Paso 4: Guardar la imagen y encontrar los cómics sitios web de anteriores, abrir desde script, Programador de tareas, launchd y bucles cronwhile, , Declaraciones de bucle whilePaso 1: Importar el módulo, Paso 1: Importar el módulo

obteniendo e imprimiendo las coordenadas del ratón usando, Paso 1: Importar el módulo Paso 1: Importar el módulo infinito,

resumen, sentencias de bucle while
Espacios en blanco, eliminación, Justificar texto con rjust(), ljust() y center()Carácter comodín (.), El signo de intercalación y el símbolo del dólarSistema operativo Windows, ¿Qué es Python?, Acerca de este libro, Descargar e instalar Python, Archivos y rutas de archivos, Iniciar otros programas

desde Python, , Programador de tareas, Ejecutar otros programas desde Pythonlaunchd y cron, Controlar el teclado y el ratón con GUIautomatización , Instalar módulos de terceros, La herramienta pip, Línea Shebang

Barra invertida vs. barra diagonal, Archivos y rutas de

archivos instalando Python, Acerca de este libro

instalación de módulos de terceros, La herramienta pip

lanzar procesos desde PythonLanzar otros programas desde Python,

salir del programa de automatización, Controlar el teclado y el ratón con la automatización GUI

apertura de archivos con aplicaciones predeterminadas, Programador de tareas,

launchd y cron herramienta pip activada, Instalación de módulos de terceros

Soporte de Python, ¿Qué es Python?

Ejecutar programas Python en, Línea Shebang iniciando

IDLE, Descargar e instalar Python

Programador de tareas, lanzar otros programas desde Python

Documentos de Word, Paso 4: Guardar los resultados Paso 4: Guardar los resultados,, WordDocumentos de , Leer documentos de Word, Obtener el texto completo de un .docxarchivo , Estilizar objetos de párrafo y de ejecución, Atributos ,de ejecución Atributos de ejecución,
Escribir documentos de WordEscribir documentos de Word, Añadir títulos,

Añadir títulos añadir títulos,

creación de documentos con estilos no predeterminados, Estilización de objetos de párrafo y de ejecución

Resumen del formato, Paso 4: Guardar los

resultados Obtención de texto de, Lectura de

documentos Word Saltos de línea/página, Añadir

encabezados

imágenes en, Añadir títulos

módulo python-docx, Paso 4: Guardar la

resultadoslectura de , Documentos Word

Ejecutar atributos deEjecutar atributos objetos,

estilizar párrafos, Obtener el texto completo de un archivo .docx

escribir en un archivo, Ejecutar atributos
Objetos de libros de trabajo, Lectura de documentos de Excellibros de trabajo, Excel, Trabajar con hojas de cálculo de Excel, Lectura de documentos de Excel, Ideas para programas similares, Creación eliminación de hojas, y Creación y eliminación de hojas

crear hojas de cálculo, Crear y Crear y eliminar hojas

eliminar hojas eliminar hojas de cálculo, abrir, Leer

documentos Excel

ahorro, Ideas para programas similares
Objetos hoja de cálculo, Obtención de hojas desde el Workbookwritemétodo (objetos , Lectura de datos desde objetos Reader en un método for Loopwriterow(),), Lectura del contenido de FilesWriterObjetos Writer

X

Cómics XKCD, Paso 3: Abrir los navegadores web para cada resultado, Paso 3: Abrir los navegadores web para cada resultado, Proyecto: Descargar todos los cómics de XKCD, Paso 1: Diseñar el programa, Paso 4: Guardar la imagen y buscar el anteriorcómic , Proyecto: Multithreaded XKCD Downloader, Proyecto: Multithreaded XKCD Downloader, Paso 1: Modificar el Programa para Usar una Función, Paso 2: Modificar el Programa para Usar una Función, Paso 3: Modificar el Programa para Usar una Función:
Crear e iniciar hilos

descargar proyecto, Paso 3: Paso 3: Abrir los navegadores web para cada resultado, Abrir los navegadores web para cada resultado, Proyecto: Descargar todos los de XKCDcómics, Paso 1: Diseñar el programa, Paso 4: Guardar la imagen y buscar el cómic anterior

programa de diseño, Proyecto: Descargar todos los cómics de XKCD

descargar página web, Paso 1: Diseñar el del programaresumen ,

Paso 3: Abrir navegadores web para cada resultado

guardar imagen, Paso 4: Guardar la imagen y buscar el cómic anterior proyecto de descarga multihilo, Proyecto: Multithreaded XKCD Downloader, Proyecto: Multithreaded XKCD Downloader, Paso 1: Modificar el Programa para Usar una Función, Paso 2: Crear e Iniciar Threads

crear e iniciar hilos, Paso 1: Modificar el programa para utilizar una función

usando la función downloadXkcd(), Proyecto: Multithreaded XKCD Downloader

Esperar a que terminen los hilosPaso 2: Crear e iniciar hilos,

Y

Directiva %Y, Pausa hasta una fecha específica

Directiva %y, Pausa hasta una fecha determinada

Yahoo! Mail, Conexión a un servidor SMTP, Recuperación y eliminación de correos electrónicos con IMAP

Z

Módulo , zipfileComprimiendo archivos con el zipfileRecorriendo un árbol de directorios, módulo , Extrayendo de archivos ZIP, Extrayendo de archivos ZIP, Paso 3:
Formar el nuevo nombre de archivo y renombrar

los archivos crear archivos ZIP, Extraer archivos

ZIP de Extraer de archivos ZIPextraer archivos

ZIP,

y carpetas, Paso 3: Formar el nuevo nombre de archivo y renombrar el

archivosresumen de , Caminar por un árbol de directorios

lectura de archivos ZIP, compresión de archivos con el módulo zipfile
Objetos ZipFile, Compresión de archivos con los zipfile objetos ModuleZipInfo,

Comprimir archivos con el módulo zipfile

Automatice las cosas aburridas con Python: Programación práctica para principiantes

Al Sweigart

Copyright© 2015

AUTOMATIZAR LAS COSAS ABURRIDAS CON PYTHON.

Reservados todos los derechos. Queda prohibida la reproducción o transmisión total o parcial de esta obra por cualquier medio, ya sea electrónico o mecánico, incluidas la fotocopia, la grabación o cualquier sistema de almacenamiento o recuperación de información, sin la autorización previa por escrito del titular de los derechos de autor y del editor.

19 18 17 16 15 1 2 3 4 5 6 7 8 9

ISBN-10: 1-59327-599-4

ISBN-13: 978-1-59327-599-0

Editor: William Pollock Editor de
producción: Laurel Chun Ilustración
de portada: Josh Ellingson Diseño de
interiores: Octopod Studios
Editores de desarrollo: Jennifer Griffith-Delgado, Greg Poulos y Leslie Shen Revisor
técnico: Ari Lacenski
Corrector: Kim Wimpsett Compositora:
Susan Glinert Stevens Correctora: Lisa
Devoto Farrell
Indexador: Servicios de indexación y corrección BIM

Para obtener información sobre distribución, traducciones o ventas al por mayor, póngase en contacto directamente

con No Starch Press, Inc: Número de control de la Biblioteca del Congreso: 2014953114

No Starch Press y el logotipo de No Starch Press son marcas registradas de No Starch Press, Inc. Otros nombres de productos y empresas aquí mencionados pueden ser marcas comerciales de sus respectivos propietarios. En lugar de utilizar un símbolo de marca comercial cada vez que aparece un nombre de marca comercial, utilizamos los nombres únicamente a modo editorial y en beneficio del propietario de la marca comercial, sin intención de infringir la marca comercial.

La información contenida en este libro se distribuye "tal cual", sin garantía. Aunque se han tomado todas las precauciones posibles en la preparación de esta obra, ni el autor ni No Starch Press, Inc. tendrán responsabilidad alguna ante ninguna persona o entidad con respecto a cualquier pérdida o daño causado o supuestamente causado directa o indirectamente por la información contenida en ella.

Esta obra está bajo licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 Unidos. Para ver una copia de esta licencia, visite http://creativecommons.org/licenses/by-nc-sa/3.0/us/ o envíe una carta a Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Prensa sin almidón

2016-01-14T10:12:21-08:00