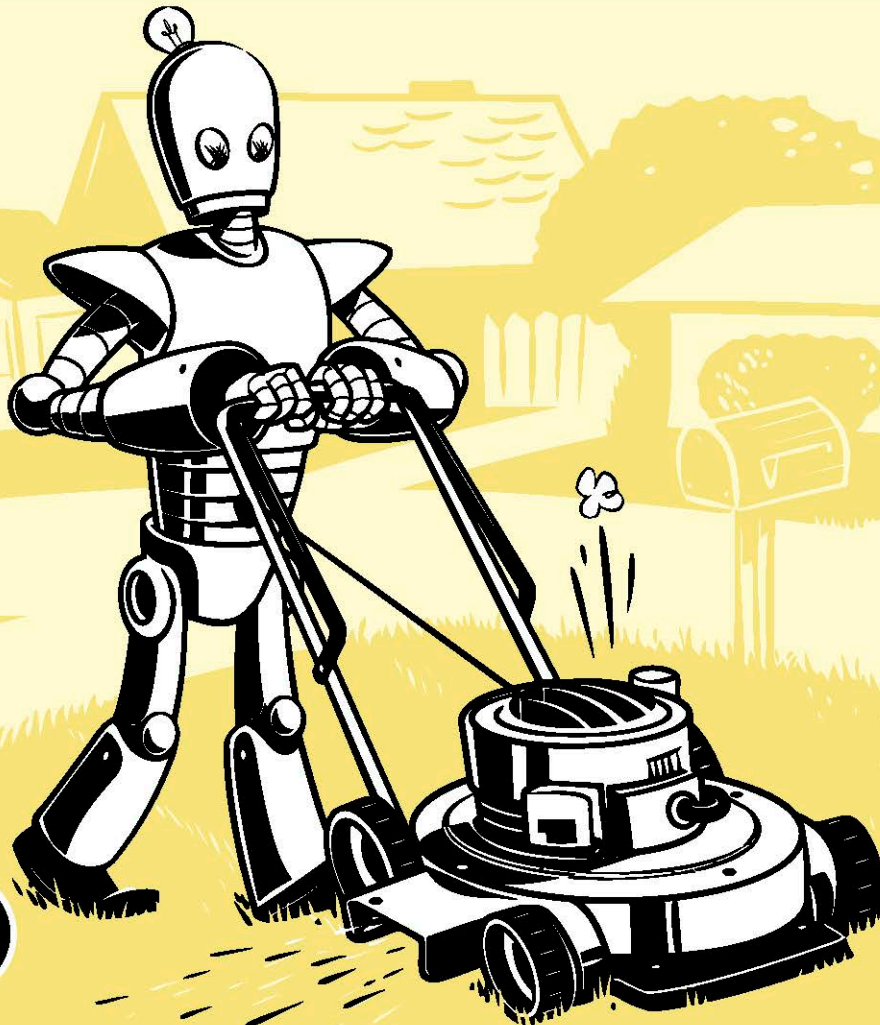


3ª EDICIÓN

# AUTOMATIZA LAS TAREAS ABURRIDAS CON PYTHON

PROGRAMACIÓN PRÁCTICA  
PARA PRINCIPIANTES

AL SWEIGART



ANAYA  
MULTIMEDIA

# ÍNDICE DE CONTENIDOS

Agradecimientos.....	6
Sobre el autor .....	6
Sobre el revisor técnico .....	6
<b>PRÓLOGO</b>	<b>23</b>
<b>INTRODUCCIÓN</b>	<b>25</b>
¿Para quién es este libro? .....	26
Convenciones de código empleadas en este libro.....	26
¿Qué es la programación?.....	27
¿Qué es Python?.....	28
Mitos habituales sobre la programación .....	28
Los programadores no necesitan saber mucho de matemáticas .....	28
No eres demasiado viejo para aprender a programar .....	29
La IA no sustituirá a los programadores .....	30
Sobre este libro .....	32
Descargar e instalar Python.....	34
Descarga e instalación de Mu .....	34
Iniciar Mu .....	34
Iniciar IDLE.....	35
El intérprete de comandos interactivo .....	35
Cómo buscar ayuda.....	36
Hacer buenas preguntas sobre la programación.....	37
Novedades en la tercera edición .....	38
Resumen .....	38
<b>PARTE I. PROGRAMACIÓN BÁSICA</b>	<b>39</b>
<b>1. PYTHON BÁSICO</b>	<b>41</b>
Introducir expresiones en el intérprete de comandos interactivo .....	42
Los tipos de datos enteros, flotantes y cadenas .....	45
Concatenación y replicación de cadenas.....	46
Almacenar valores en variables .....	48
Declaraciones de asignación .....	48
Nombres de variables .....	49
Tu primer programa .....	50
Diseción del programa.....	52
Comentarios.....	52
La función print() .....	52
La función input() .....	53
El mensaje de saludo .....	53
La función len() .....	54

Las funciones <code>str()</code> , <code>int()</code> y <code>float()</code> .....	55
La función <code>type()</code> .....	57
Las funciones <code>round()</code> y <code>abs()</code> .....	58
Cómo guardan los ordenadores datos con números binarios .....	59
Resumen .....	62
Preguntas prácticas .....	62
<b>2. IF-ELSE Y EL CONTROL DE FLUJO</b> .....	<b>65</b>
Valores booleanos.....	66
Operaciones de comparación .....	67
Operadores booleanos.....	69
Combinar operadores booleanos y de comparación .....	70
Componentes del control de flujo.....	71
Condiciones.....	71
Bloques de código .....	72
Ejecución de programas .....	72
Las declaraciones de control de flujo.....	73
if.....	73
else .....	73
elif .....	75
Un programa corto: el Día de todo al revés .....	80
Un programa corto: una calculadora de capacidad falsa .....	81
Resumen .....	83
Preguntas prácticas .....	84
<b>3. BUCLES</b> .....	<b>85</b>
Declaraciones de bucle <code>while</code> .....	85
Un molesto bucle <code>while</code> .....	88
Declaraciones <code>break</code> .....	89
Declaraciones <code>continue</code> .....	91
Bucles <code>for</code> y la función <code>range()</code> .....	94
Un bucle <code>while</code> equivalente .....	96
Argumentos para <code>range()</code> .....	97
Importación de módulos .....	98
Terminar un programa antes de tiempo con <code>sys.exit()</code> .....	99
Un programa corto: adivina el número .....	100
Un programa corto: piedra, papel o tijera .....	102
Resumen .....	105
Preguntas prácticas .....	106
<b>4. FUNCIONES</b> .....	<b>107</b>
Crear funciones .....	107
Argumentos y parámetros .....	109
Valores de retorno y declaraciones <code>return</code> .....	110
El valor <code>None</code> .....	111
Parámetros con nombre .....	112
La pila de llamadas.....	114

Ámbito local y global .....	116
Reglas de ámbito .....	116
La declaración global.....	119
Identificación del ámbito.....	119
Manejo de excepciones.....	121
Un programa corto: zigzag.....	123
Un programa corto: espiga.....	125
Resumen .....	127
Preguntas prácticas .....	127
Programas prácticos.....	128
La secuencia de Collatz.....	128
Validación de la entrada .....	128
<b>5. DEPURACIÓN</b> .....	<b>129</b>
Lanzar excepciones.....	130
Aserciones .....	131
Registros .....	133
El módulo <code>logging</code> .....	133
Archivos de registro .....	134
Una mala práctica: depurar con <code>print()</code> .....	135
Niveles de registro .....	135
Registro deshabilitado .....	136
El depurador de Mu .....	137
Depurar un programa de suma.....	138
Configurar puntos de interrupción.....	140
Resumen .....	141
Preguntas prácticas .....	142
Programa de práctica: depurar el lanzamiento de moneda.....	142
<b>6. LISTAS</b> .....	<b>143</b>
El tipo de datos lista.....	143
Índices.....	144
Índices negativos .....	145
Rebanadas .....	146
La función <code>len()</code> .....	147
Actualización de valores.....	147
Concatenación y replicación.....	147
Declaraciones del .....	148
Trabajar con listas.....	148
Bucles <code>for</code> y listas .....	149
Los operadores <code>in</code> y <code>not in</code> .....	150
El truco de la asignación múltiple.....	151
Enumeración de elementos de una lista .....	152
Selección y orden aleatorios .....	152
Operadores de asignación aumentada.....	153
Métodos .....	154
Encontrar valores.....	154
Añadir valores.....	154

Quitar valores .....	155
Almacenar valores .....	156
Invertir valores .....	157
Cortocircuito lógico con operadores booleanos .....	158
Un programa corto: <i>Magic 8 Ball</i> con una lista .....	159
Tipo de datos de secuencia .....	159
Tipos de datos mutables e inmutables .....	160
El tipo de datos tupla .....	161
Conversión entre los tipos lista y tupla .....	162
Referencias .....	163
Argumentos .....	164
Las funciones <code>copy()</code> y <code>deepcopy()</code> .....	165
Un programa corto: el salvapantallas de <i>Matrix</i> .....	166
Resumen .....	169
Preguntas prácticas .....	170
Programas de práctica .....	170
Poner comas .....	170
Lanzamientos de moneda .....	171

## 7. DICCIONARIOS Y ESTRUCTURACIÓN DE DATOS 173

El tipo de datos diccionario .....	173
Comparación de listas y diccionarios .....	174
Devolución de claves y valores .....	176
Comprobar si existe una clave .....	177
Establecer valores predeterminados .....	178
Modelar cosas del mundo real con estructuras de datos .....	179

### Proyecto 1: Simulador de un tablero de ajedrez interactivo 181

Paso 1: configurar el programa .....	183
Paso 2: crear una plantilla de tablero de ajedrez .....	183
Paso 3: imprimir el tablero de ajedrez actual .....	184
Paso 4: manipular el tablero de ajedrez .....	186
Diccionarios y listas anidados .....	188
Resumen .....	190
Preguntas prácticas .....	190
Programas de práctica .....	190
Validar el diccionario de ajedrez .....	190
Inventario de juego de fantasía .....	191
Conversión del botón de lista a diccionario .....	191

## 8. EDICIÓN DE CADENAS Y TEXTO 193

Trabajar con cadenas .....	193
Cadenas literales .....	193
Índices y rebanadas .....	196
Los operadores <code>in</code> y <code>not in</code> .....	197
Cadenas formateadas .....	198
Alternativas a las cadenas formateadas: <code>%s</code> y <code>format()</code> .....	199
Métodos de cadena útiles .....	200
Cambiar mayúsculas y minúsculas .....	200

Comprobar las características de una cadena .....	201
Comprobar el principio o el final de una cadena .....	203
Unir y separar cadenas .....	203
Justificar y centrar texto .....	204
Eliminar espacios en blanco .....	205
Puntos de código numéricos de caracteres .....	206
Copiar y pegar cadenas .....	207

### Proyecto 2: Añadir viñetas a marcado wiki 208

Paso 1: copiar y pegar del portapapeles .....	208
Paso 2: separar las líneas de texto .....	209
Paso 3: unir las líneas modificadas .....	210
Un programa corto: <i>pig latin</i> .....	210
Resumen .....	214
Preguntas prácticas .....	214
Programa de práctica: impresión de tablas .....	215

## PARTE II. AUTOMATIZACIÓN DE TAREAS 217

### 9. COINCIDENCIA DE PATRONES DE TEXTO CON EXPRESIONES REGULARES 219

Buscar patrones de texto sin expresiones regulares .....	220
Buscar patrones de texto con expresiones regulares .....	222
La sintaxis de las expresiones regulares .....	223
Agrupar con paréntesis .....	224
Usar caracteres de escape .....	224
Coincidencia de caracteres de grupos alternos .....	225
Devolver todas las coincidencias .....	226
Sintaxis calificadora: qué caracteres hacer coincidir .....	227
Usar clases de caracteres y clases de caracteres negativas .....	227
Utilizar clases de caracteres estenográficas .....	228
Hacer coincidir todo con el carácter punto .....	228
Ten cuidado con lo que haces coincidir .....	229
Sintaxis de cuantificadores: cuántos calificadores buscar .....	230
Buscar un patrón opcional .....	230
Buscar cero o varios calificadores .....	231
Buscar uno o más calificadores .....	231
Buscar un número específico de calificadores .....	232
Búsquedas voraces y no voraces .....	233
Buscar todo .....	233
Buscar caracteres de salto de línea .....	234
Buscar al principio y al final de una cadena .....	235
Búsquedas sin tener en cuenta mayúsculas y minúsculas .....	237
Sustituir cadenas .....	237
Gestionar expresiones regulares complejas con el modo verboso .....	238
Combinar <code>re.IGNORECASE</code> , <code>re.DOTALL</code> y <code>re.VERBOSE</code> .....	239

### Proyecto 3: Extraer información de contacto de documentos grandes 239

Paso 1: crear una expresión regular para números de teléfono .....	240
Paso 2: crear una expresión regular para las direcciones de correo electrónico .....	241

Paso 3: buscar todas las coincidencias en el texto del portapapeles .....	241
Paso 4: unir las coincidencias en una cadena .....	242
Ideas para programas similares.....	243
Humre: un módulo para expresiones regulares legibles para humanos .....	243
Resumen .....	247
Preguntas prácticas .....	248
Programas de práctica .....	249
Detección de contraseñas fuertes .....	249
Versión en expresión regular del método strip().....	249

## 10. LEER Y ESCRIBIR ARCHIVOS 251

Archivos y rutas de archivos.....	251
Estandarizar los separadores de ruta .....	252
Unir rutas.....	253
Acceder al directorio de trabajo actual .....	254
Acceder a la carpeta de inicio .....	255
Especificar rutas absolutas o relativas.....	256
Crear carpetas nuevas.....	257
Manejar rutas absolutas y relativas .....	257
Obtener las partes de una ruta de archivo .....	258
Encontrar tamaños de archivo y sellos de tiempo .....	260
Buscar archivos con patrones globales .....	261
Comprobar la validez de una ruta .....	262
El proceso de lectura y escritura de archivos .....	263
Abrir archivos .....	264
Leer el contenido de archivos .....	265
Escribir en archivos .....	266
Usar declaraciones with .....	267
Guardar variables con el módulo shelve .....	268
<b>Proyecto 4: Generar archivos de cuestionarios aleatorios</b> .....	<b>269</b>
Paso 1: guardar los datos del cuestionario en un diccionario .....	270
Paso 2: crear el archivo de cuestionario .....	271
Paso 3: crear las opciones de respuesta .....	272
Paso 4: escribir contenido en los archivos.....	272
Resumen .....	274
Preguntas prácticas .....	274
Programas de práctica .....	275
<i>Mad Libs</i> .....	275
Buscar expresiones regulares .....	275

## 11. ORGANIZAR ARCHIVOS 277

El módulo shutil .....	278
Copiar archivos y carpetas .....	278
Mover y renombrar archivos y carpetas.....	279
Borrar archivos y carpetas de manera permanente .....	280
Enviar a la Papelera de reciclaje .....	280
Recorrer un árbol de directorios.....	281

Comprimir archivos con el módulo zipfile .....	283
Crear y añadir a archivos ZIP .....	283
Leer archivos ZIP.....	284
Extraer de archivos ZIP .....	285
<b>Proyecto 5: Hacer una copia de seguridad de una carpeta en un archivo ZIP</b> .....	<b>286</b>
Paso 1: averiguar el nombre del archivo ZIP .....	286
Paso 2: crea el nuevo archivo ZIP .....	287
Paso 3: recorrer el árbol de directorios.....	288
Ideas para otros programas .....	288
Resumen .....	289
Preguntas de práctica .....	289
Programas de práctica .....	289
Copia selectiva.....	289
Eliminar archivos innecesarios.....	290
Renumerar archivos.....	290
Convertir fechas del formato americano al europeo .....	290

## 12. DISEÑAR Y DESPLEGAR PROGRAMAS DE LÍNEA DE COMANDOS 291

Un programa con cualquier otro nombre .....	292
Utilizar el terminal.....	293
Los comandos cd, pwd, dir y ls .....	294
La variable de entorno PATH .....	295
Editar PATH.....	296
Los comandos which y where .....	297
Entornos virtuales .....	297
Instalar paquetes de Python con pip .....	299
Programas de Python autoconscientes .....	301
Diseño de programas basados en texto .....	302
Nombres de comandos cortos.....	303
Argumentos de línea de comandos .....	303
Entrada y salida del portapapeles .....	304
Texto en color con Bext.....	305
Limpiar el terminal .....	306
Notificaciones de sonido y texto.....	306
Un programa corto: tormenta de nieve .....	307
Mensajes emergentes con PyMsgBox .....	309
Desplegar programas de Python .....	310
Windows.....	310
macOS.....	311
Ubuntu Linux .....	312
Un programa corto: copiar el directorio de trabajo actual .....	313
Windows.....	315
macOS.....	315
Ubuntu Linux .....	315
Un programa corto: grabadora del portapapeles .....	316
Windows.....	318
macOS.....	319
Ubuntu Linux .....	319

Compilar programas de Python con PyInstaller .....	320
Resumen .....	321
Preguntas prácticas .....	321
Programa de práctica: haz tus programas desplegables .....	322

### 13. EXTRAER INFORMACIÓN DE SITIOS WEB 323

HTTP y HTTPS .....	324
<b>Proyecto 6: Ejecutar un programa con el módulo webbrowser</b> .....	<b>324</b>
Paso 1: descubrir la URL .....	325
Paso 2: manejar los argumentos de línea de comandos .....	326
Paso 3: recuperar el contenido del portapapeles .....	326
Ideas para programas similares .....	327
Descargar archivos de la web con el módulo requests .....	328
Descargar páginas web .....	328
Buscar errores .....	329
Guardar archivos descargados en el disco duro .....	330
Acceder a una API del tiempo .....	331
Solicitar una latitud y una longitud .....	333
Buscar el tiempo actual .....	334
Obtener una predicción meteorológica .....	334
Explorar las API .....	335
Entender HTML .....	335
Explorar el formato .....	335
Ver el código fuente de una página web .....	336
Abrir las herramientas para desarrolladores de tu navegador .....	337
Buscar elementos HTML .....	338
Analizar HTML con BeautifulSoup .....	340
Crear un objeto BeautifulSoup .....	341
Buscar un elemento .....	341
Obtener datos de atributos del elemento .....	343
<b>Proyecto 7: Abrir todos los resultados de una búsqueda</b> .....	<b>344</b>
Paso 1: abrir la página de búsqueda .....	344
Paso 2: buscar todos los resultados .....	345
Paso 3: abrir navegadores para cada resultado .....	345
Ideas para programas similares .....	346
<b>Proyecto 8: Descargar cómics de XKCD</b> .....	<b>346</b>
Paso 1: diseñar el programa .....	348
Paso 2: descargar la página web .....	348
Paso 3: buscar y descargar la imagen del cómic .....	349
Paso 4: guardar la imagen y buscar el cómic anterior .....	350
Ideas para programas similares .....	351
Controlar el navegador con Selenium .....	352
Iniciar un navegador controlado por Selenium .....	352
Hacer clic en los botones del navegador .....	353
Buscar elementos en la página .....	353
Hacer clic en elementos de la página .....	355

Rellenar y enviar formularios .....	356
Enviar teclas especiales .....	356
Controlar el navegador con Playwright .....	357
Abrir un navegador controlado por Playwright .....	358
Hacer clic en los botones del navegador .....	359
Buscar elementos en la página .....	359
Hacer clic en elementos de la página .....	362
Rellenar y enviar formularios .....	362
Enviar teclas especiales .....	363
Resumen .....	364
Preguntas prácticas .....	364
Programas de práctica .....	365
Descargar imágenes de un sitio .....	365
2048 .....	365
Verificación de enlaces .....	365

### 14. HOJAS DE CÁLCULO DE EXCEL 367

Leer archivos Excel .....	368
Abrir un libro .....	369
Obtener hojas del libro .....	369
Obtener celdas de las hojas .....	370
Conversión entre letras y números de columna .....	372
Obtener filas y columnas .....	372
<b>Proyecto 9: Reunir estadísticas censales</b> .....	<b>374</b>
Paso 1: leer los datos de la hoja de cálculo .....	375
Paso 2: poblar la estructura de datos .....	376
Paso 3: escribir los resultados en un archivo .....	377
Ideas para programas similares .....	378
Escribir en archivos de Excel .....	378
Crear y guardar archivos de Excel .....	379
Crear y eliminar hojas .....	379
Escribir valores en celdas .....	380
<b>Proyecto 10: Actualizar una hoja de cálculo</b> .....	<b>381</b>
Paso 1: configurar una estructura de datos con la información actualizada .....	382
Paso 2: comprobar todas las filas y actualizar los precios incorrectos .....	382
Ideas para programas similares .....	383
Establecer el estilo de fuente de las celdas .....	383
Fórmulas .....	385
Ajustar filas y columnas .....	387
Establecer la altura de fila y el ancho de columna .....	387
Combinar y separar celdas .....	388
Inmovilizar paneles .....	388
Gráficos .....	390
Resumen .....	391
Preguntas prácticas .....	392
Programas de práctica .....	392
Tabla de multiplicar .....	392
Insertar filas en blanco .....	393

<b>15. HOJAS DE CÁLCULO DE GOOGLE</b>	<b>395</b>
Instalar y configurar EZSheets.....	395
Crear un nuevo proyecto en Google Cloud.....	396
Habilitar las API de Hojas de cálculo y Drive.....	396
Configurar la pantalla de consentimiento de OAuth.....	397
Crear credenciales.....	397
Iniciar sesión con el archivo de credenciales.....	398
Revocar el archivo de credenciales.....	398
Objetos Spreadsheet.....	399
Crear, actualizar y hacer listas de hojas de cálculo.....	399
Acceder a los atributos de una hoja de cálculo.....	400
Descargar y actualizar hojas de cálculo.....	401
Borrar hojas de cálculo.....	402
Objetos Sheet.....	403
Leer y escribir datos.....	403
Crear, mover y eliminar hojas.....	408
Copiar hojas.....	410
Formularios de Google.....	411
<b>Proyecto 11: Estafa de criptomonedas con <i>blockchain</i> falso</b>	<b>411</b>
Paso 1: auditar el <i>blockchain</i> falso.....	412
Paso 2: hacer transacciones.....	414
Trabajar con las cuotas de Hojas de cálculo de Google.....	415
Resumen.....	416
Preguntas prácticas.....	416
Programas de práctica.....	416
Descargar datos de Formularios de Google.....	416
Convertir hojas de cálculo a otros formatos.....	417
Buscar errores en una hoja de cálculo.....	417
<b>16. BASES DE DATOS SQLITE</b>	<b>419</b>
Hojas de cálculo vs. bases de datos.....	420
SQLite vs. otras bases de datos SQL.....	422
Crear bases de datos y tablas.....	423
Conectarse a bases de datos.....	424
Crear tablas.....	425
Definir tipos de datos.....	425
Listas de tablas y columnas.....	427
Operaciones CRUD de base de datos.....	428
Introducir datos en la base de datos.....	428
Leer datos de la base de datos.....	430
Actualizar datos en la base de datos.....	435
Eliminar datos de la base de datos.....	436
Deshacer transacciones.....	437
Hacer copias de seguridad de bases de datos.....	438
Modificar y eliminar tablas.....	439
Unir varias tablas con claves foráneas.....	440
Bases de datos y copias de seguridad en memoria.....	442

Copiar bases de datos.....	443
Apps con SQLite.....	443
Resumen.....	444
Preguntas prácticas.....	445
Programas de práctica.....	445
Comprobador de vacunación de gatos.....	446
Base de datos de ingredientes de comidas.....	446

## 17. DOCUMENTOS DE WORD Y PDF **447**

Documentos en PDF.....	447
Extraer texto.....	448
Postprocesamiento con IA.....	449
Extraer imágenes.....	451
Crear PDF a partir de otras páginas.....	452
<b>Proyecto 12: Combinar páginas seleccionadas de varios PDF</b>	<b>458</b>
Paso 1: buscar todos los archivos PDF.....	458
Paso 2: abrir cada PDF.....	459
Paso 3: guardar los resultados.....	460
Ideas para programas similares.....	460
Documentos de Word.....	460
Leer documentos de Word.....	461
Sacar todo el texto de un archivo .docx.....	462
Dar estilo a objetos Paragraph y Run.....	463
Aplicar atributos de Run.....	464
Escribir documentos de Word.....	466
Añadir encabezados.....	467
Añadir saltos de línea y de página.....	468
Añadir imágenes.....	469
Resumen.....	469
Preguntas prácticas.....	470
Programas de práctica.....	470
Paranoia PDF.....	470
Invitaciones personalizadas.....	471
Revienta contraseñas de PDF.....	471

## 18. ARCHIVOS CSV, JSON Y XML **473**

El formato CSV.....	474
Leer archivos CSV.....	475
Acceder a los datos de un bucle for.....	476
Escribir archivos CSV.....	476
Usar tabuladores en vez de comas.....	477
Manejar filas de encabezado.....	478
<b>Proyecto 13: Quitar el encabezado de archivos CSV</b>	<b>480</b>
Paso 1: pasar en bucle por todos los archivos.....	481
Paso 2: leer el archivo.....	481
Paso 3: escribir el nuevo archivo CSV.....	482
Ideas para programas similares.....	483

Formatos de texto simple versátiles.....	483
JSON.....	485
XML.....	487
Resumen.....	493
Preguntas prácticas.....	493
Programa de práctica: convertidor de Excel a CSV.....	494

## 19. MANEJAR EL TIEMPO, PROGRAMAR TAREAS Y LANZAR PROGRAMAS 495

El módulo time.....	495
Devolver la marca de tiempo <i>epoch</i> .....	496
Pausar programas.....	497

### Proyecto 14: Supercronómetro 498

Paso 1: configurar el programa para que rastree las horas.....	498
Paso 2: rastrear e imprimir los tiempos de las vueltas.....	498
Ideas para programas similares.....	499

El módulo <i>datetime</i> .....	500
Representar la duración.....	501
Pausar hasta una fecha específica.....	503
Convertir objetos <i>datetime</i> en cadenas.....	503
Convertir cadenas en objetos <i>datetime</i> .....	504
Iniciar otros programas desde Python.....	506
Pasar argumentos de línea de comandos a procesos.....	508
Recibir texto de salida de comandos lanzados.....	508
Ejecutar el Programador de tareas, <i>launchd</i> y <i>cron</i> .....	509
Abrir archivos con aplicaciones predeterminadas.....	509

### Proyecto 15: Temporizador sencillo 510

Paso 1: la cuenta atrás.....	510
Paso 2: reproducir el archivo de sonido.....	511
Ideas para programas similares.....	511

Resumen.....	512
Preguntas prácticas.....	512
Programas de práctica.....	512
Cronómetro con mejor aspecto.....	513
Buscador de viernes 13.....	513

## 20. ENVIAR CORREOS ELECTRÓNICOS, TEXTOS Y NOTIFICACIONES PUSH 515

La API de Gmail.....	516
Habilitar la API.....	516
Enviar correos.....	516
Leer correos.....	517
Buscar correos.....	518
Descargar adjuntos.....	519
Pasarelas de correo a SMS.....	520
Notificaciones <i>push</i> .....	521
Enviar notificaciones.....	522
Transmitir metadatos.....	523
Recibir notificaciones.....	523

Resumen.....	525
Preguntas prácticas.....	526
Programas de práctica.....	526
Recordatorio de paraguas.....	526
Cancelador de suscripciones automático.....	527
Control informático basado en correo electrónico.....	527

## 21. HACER GRÁFICOS Y MANIPULAR IMÁGENES 529

Fundamentos de la imagen digital.....	530
Colores y valores RGBA.....	530
Coordenadas y tuplas de cuadros.....	531
Manipular imágenes con Pillow.....	532
Trabajar con el tipo de datos <i>Image</i> .....	534
Recortar imágenes.....	535
Pegar imágenes en otras imágenes.....	536
Cambiar el tamaño de imágenes.....	539
Rotar y voltear imágenes.....	540
Cambiar píxeles individuales.....	542

### Proyecto 16: Añadir un logo 543

Paso 1: abrir la imagen del logotipo.....	544
Paso 2: pasar en bucle por todos los archivos.....	545
Paso 3: redimensionar las imágenes.....	545
Paso 4: añadir el logo y guardar los cambios.....	546
Ideas para programas similares.....	548
Dibujar en imágenes.....	548
Formas.....	548
Texto.....	550
Copiar y pegar imágenes en el portapapeles.....	552
Crear gráficos con Matplotlib.....	553
Gráficos de líneas y diagramas de dispersión.....	553
Gráficos de barras y circulares.....	555
Componentes adicionales.....	557
Resumen.....	558
Preguntas prácticas.....	559
Programas de práctica.....	560
Creador de mosaicos.....	560
Identificar carpetas de fotos en el disco duro.....	560
Crear tarjetas de asiento personalizadas.....	561

## 22. RECONOCER TEXTO EN IMÁGENES 563

Instalar Tesseract y PyTesseract.....	564
Windows.....	564
macOS.....	564
Linux.....	564
PyTesseract.....	564
Fundamentos del OCR.....	565
Preprocesar una imagen.....	565
Corregir errores con modelos de lenguaje de gran tamaño.....	566

Reconocer texto en idiomas distintos del inglés .....	568
La aplicación de escaneo NAPS2 .....	569
Instalar y configurar NAPS2 .....	570
Ejecutar NAPS2 desde Python .....	570
Especificar entrada .....	571
Resumen .....	572
Preguntas prácticas .....	572
Programa de práctica: extractor de texto de navegador .....	573
<b>23. CONTROLAR EL RATÓN Y EL TECLADO</b> .....	<b>575</b>
Configurar aplicaciones de accesibilidad en macOS .....	576
Mantener el rumbo .....	576
Pausas y medidas de seguridad .....	577
Cierre de sesión .....	577
Controlar el movimiento del ratón .....	577
Mover el ratón .....	578
Obtener la posición actual .....	579
Controlar la interacción del ratón .....	580
Hacer clic .....	580
Arrastrar .....	581
Desplazarse .....	582
Planificar los movimientos del ratón .....	583
Hacer capturas de pantalla .....	584
Reconocimiento de imágenes .....	586
Obtener información de la ventana .....	588
Obtener la ventana activa .....	588
Buscar ventanas con otras funciones .....	589
Manipular ventanas .....	589
Controlar el teclado .....	592
Enviar cadenas de pulsaciones de teclas .....	592
Especificar nombres de teclas .....	593
Pulsar y soltar teclas .....	594
Ejecutar combinaciones de teclas rápidas .....	595
Configurar <i>scripts</i> de automatización de GUI .....	595
Mostrar cuadros de mensaje .....	597
Resumen .....	598
Preguntas prácticas .....	598
Programas de práctica .....	599
Parecer ocupado .....	599
Leer campos de texto con el portapapeles .....	599
Escribir un bot que juega .....	600
<b>24. MOTORES DE TEXTO A VOZ Y RECONOCIMIENTO DE VOZ</b> .....	<b>601</b>
Motor de texto a voz .....	602
Generar voz .....	602
Guardar sonido en archivos WAV .....	603
Reconocimiento de voz .....	604
Crear archivos de subtítulo .....	606

Descargar vídeos de sitios web .....	607
Resumen .....	609
Preguntas prácticas .....	610
Programas de práctica .....	610
Añadir voz a <i>Adivina el número</i> .....	610
Cantar <i>99 Bottles of Beer</i> .....	611
Transcriptor de YouTube .....	611

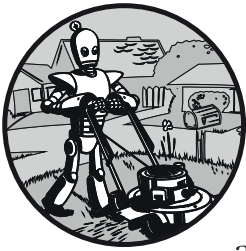
## APÉNDICE A. INSTALAR PAQUETES DE TERCEROS **613**

Instalar pip .....	613
Encontrar pip .....	614
Ejecutar pip desde entornos virtuales .....	614
Instalar los paquetes utilizados en este libro .....	614

## APÉNDICE B. RESPUESTAS A LAS PREGUNTAS PRÁCTICAS **617**

## ÍNDICE ALFABÉTICO **633**

# INTRODUCCIÓN



"Acabas de hacer en dos horas lo que a nosotros tres nos ha llevado dos días". Mi compañero de habitación en la universidad estaba trabajando en una tienda de electrónica a principios de los años 2000. De vez en cuando, la tienda recibía una hoja de cálculo con miles de precios de productos de otras tiendas. Un equipo de tres empleados tenía que imprimir la hoja en una montaña de papeles y repartírsela entre ellos. Para cada precio de producto, debían buscar el de su tienda y anotar todos los que la competencia vendía más baratos. Por lo general, tardaban un par de días.

"Pues podría escribir un programa para que se ocupe de eso si tenéis el archivo original para imprimir", les dijo mi compañero cuando los vio sentados en el suelo, rodeados de papeles.

Al cabo de dos horas, tenía un programa corto que leía los precios de la competencia de un archivo, buscaba el producto en la base de datos de la tienda y anotaba si la competencia era más barata. Todavía era novato en la programación, así que pasó la mayor parte de ese tiempo buscando documentación en un libro de programación. El programa en sí tardó solo unos segundos en ejecutarse. Mi compañero de habitación y sus compañeros de trabajo se tomaron un buen rato para comer aquel día.

Ese es el poder de la programación informática. Un ordenador es como una navaja suiza con herramientas para un sinnúmero de tareas. Mucha gente pasa horas haciendo clics y tecleando para completar tareas repetitivas, sin darse cuenta de que la máquina que están usando podría hacer ese trabajo en cuestión de segundos si le diesen las instrucciones adecuadas.

## ¿Para quién es este libro?

El software es el núcleo de muchas de las herramientas que usamos actualmente: casi todo el mundo usa redes sociales para comunicarse, casi todo el mundo tiene teléfonos interconectados en el bolso o bolsillo y la mayoría de los trabajos de oficina implican interactuar con un ordenador. En consecuencia, la demanda de profesionales que sepan escribir código se ha disparado. Hay innumerables libros, tutoriales y formaciones intensivas para desarrolladores que prometen convertir a ambiciosos principiantes en ingenieros de software con sueldos de seis cifras.

Este libro no es para esa gente. Es para todos los demás.

Por sí solo, el libro no convierte a nadie en desarrollador de software profesional, igual que unas cuantas lecciones de guitarra no convierten a nadie en estrella del rock. Pero, si trabajas en una oficina, eres administrador, profesor o usas un ordenador para trabajo o diversión, aprenderás lo básico de la programación para poder automatizar tareas como estas:

- Mover y renombrar miles de archivos y clasificarlos en carpetas.
- Rellenar formularios en línea sin necesidad de teclear.
- Descargar archivos o copiar texto de un sitio web cada vez que se actualice.
- Hacer que el ordenador envíe notificaciones de texto personalizadas a tu teléfono.
- Actualizar o aplicar formato a hojas de cálculo de Excel.
- Comprobar tu correo electrónico y enviar respuestas prefabricadas.
- Crear bases de datos y consultar información.
- Extraer texto de imágenes y archivos de sonido.

Estas tareas son sencillas, pero consumen mucho tiempo a los humanos y muchas veces son tan triviales o específicas que no existe un software listo para usar que las realice. No obstante, armado con unos pocos conocimientos de programación, puedes conseguir que tu ordenador haga esas tareas por ti.

## Convenciones de código empleadas en este libro

Este libro no está diseñado como un manual de referencia; es una guía para principiantes. El estilo del código a veces no se ajusta a las mejores prácticas (por ejemplo, algunos programas utilizan variables globales), pero de ese modo es más fácil aprenderlo. No se cubren conceptos sofisticados de programación, como la programación orientada a objetos, la comprensión de listas y los generadores, por la complejidad

que suponen. Los programadores veteranos podrían ver maneras de cambiar el código del libro para mejorar la eficiencia, pero el objetivo es conseguir programas que funcionen con el menor esfuerzo posible por parte del lector.

## ¿Qué es la programación?

Las series de la tele y las películas suelen presentar a los programadores tecleando con furia crípticas cadenas de unos y ceros en pantallas brillantes, pero la programación moderna no es tan misteriosa. Programar es escribir instrucciones para el ordenador en un lenguaje que este pueda entender. Tales instrucciones podrían hacer cuentas con números, modificar texto, buscar información en archivos o comunicarse con otros ordenadores por Internet.

Todos los programas usan instrucciones básicas como bloques de construcción. Aquí tenemos algunas de las más habituales, en español:

"Haz esto; luego haz eso".

"Si se da esta condición, realiza esta acción; si no, haz esta otra".

"Realiza esta acción exactamente 27 veces".

"Sigue haciendo eso hasta que esta condición sea verdadera".

También podemos combinar estos bloques de construcción para implementar decisiones más complejas. Por ejemplo, a continuación tenemos las instrucciones de programación, denominadas código fuente, para un programa sencillo escrito en el lenguaje de programación Python. Empezando desde arriba, el software de Python ejecuta líneas de código (algunas de las cuales solo si se da una condición determinada, si no, Python ejecuta otra línea) hasta llegar al final:

---

```

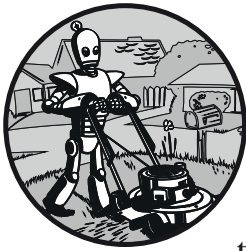
❶ password_file = open('ArchivoContraseñaSecreta.txt')
❷ secret_password = password_file.read()
❸ print('Introduce contraseña.')
   typed_password = input()
❹ if typed_password == secret_password:
   ❺ print('Acceso concedido')
   ❻ if typed_password == '12345':
   ❼ print('Esa es la típica contraseña que pone un idiota en la maleta.')
else:
   ❽ print('Acceso denegado')
```

---

Es posible que no sepas nada de programación, pero probablemente puedas intuir lo que hace el código anterior leyéndolo sin más. Primero, se abre el archivo `ArchivoContraseñaSecreta.txt` ❶ y, a continuación, se lee la contraseña secreta ❷. Después, se insta al usuario a introducir una contraseña (con el teclado) ❸. Se comparan las dos contraseñas ❹ y, si coinciden, el programa escribe en la pantalla "Acceso concedido" ❺. Luego el programa comprueba si la contraseña es "12345" ❻ y hace notar que esa opción no es la mejor para una contraseña ❼. Si las contraseñas no coinciden, el programa escribe "Acceso denegado" en la pantalla ❽.

# 3

## BUCLES



En el capítulo anterior, aprendiste a hacer que los programas ejecuten algunos bloques de código y se salten otros, pero el control de flujo va más allá. En este capítulo, aprenderás a ejecutar repetidamente bloques de código utilizando bucles. Los dos tipos de bucles de Python, `while` y `for`, desatan todo el poder de la automatización, ya que pueden ejecutar líneas de código millones de veces por segundo. También veremos cómo importar librerías de código, llamadas módulos, para poner más funciones todavía a disposición de tus programas.

### Declaraciones de bucle `while`

Podemos hacer que un bloque de código se ejecute una y otra vez utilizando una declaración `while`. El código de la proposición `while` se ejecutará mientras la condición de la declaración sea `True`. El código de una declaración `while` siempre consta de lo siguiente:

- La palabra clave `while`.
- Una condición (es decir, una expresión que se evalúe como `True` o `False`).
- Dos puntos.
- A partir de la siguiente línea, un bloque de código sangrado (llamado proposición o bloque `while`).

Como verás, una declaración `while` se parece mucho a una declaración `if`. La diferencia está en cómo se comportan: al final de una proposición `if`, la ejecución del programa continúa después de la declaración `if`, mientras que, al final de una proposición `while`, la ejecución vuelve al principio de la declaración `while`. La proposición `while` se denomina a menudo el bucle `while` o, simplemente, el bucle.

Vamos a ver una declaración `if` y un bucle `while` que emplean la misma condición y realizan las mismas acciones basándose en dicha condición. Este es el código con una declaración `if`:

---

```
spam = 0
if spam < 5:
    print('Hello, world.')
    spam = spam + 1
```

Y este es el código con una declaración `while`:

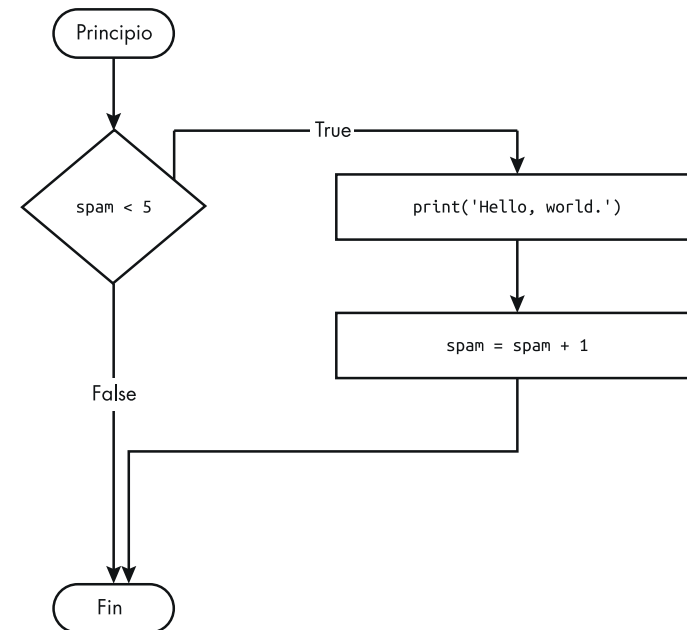
```
spam = 0
while spam < 5:
    print('Hello, world.')
    spam = spam + 1
```

---

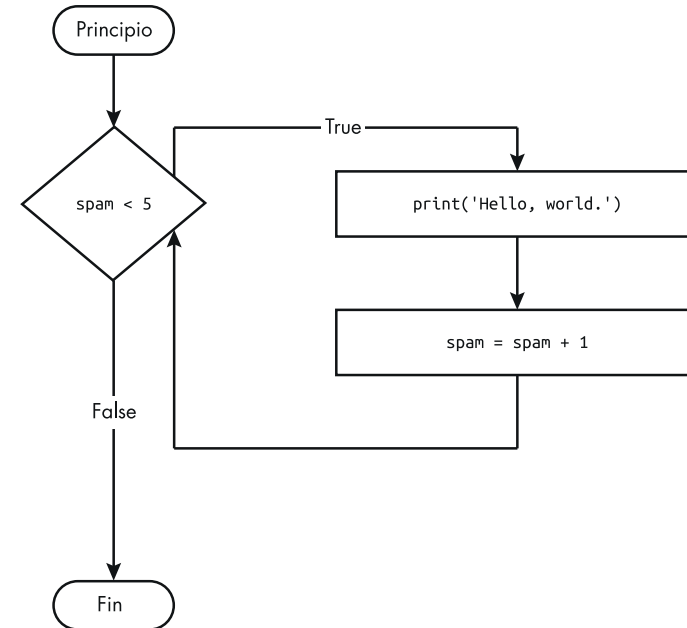
Las dos declaraciones son similares: ambas comprueban el valor de `spam` y, si es menor que 5, muestran un mensaje. Pero, al ejecutar estos dos fragmentos de código, ocurrirá algo muy distinto con cada uno. Para la declaración `if`, la salida será simplemente "Hello, world.". En cambio, para la declaración `while`, será "Hello, world." repetido cinco veces. Echa un vistazo a los diagramas de flujo de estos dos códigos, representados en las figuras 3.1 y 3.2, para ver por qué pasa eso.

El código con la declaración `if` comprueba la condición e imprime "Hello, world." solo una vez si esa condición es verdadera. El código con el bucle `while`, por su parte, lo imprime cinco veces. El bucle se detiene después de cinco impresiones porque el entero de `spam` aumenta en uno al final de cada iteración del bucle, lo que significa que este se ejecutará cinco veces antes de que `spam < 5` sea `False`.

En el bucle `while`, la condición siempre se comprueba al inicio de cada iteración (es decir, cada vez que se ejecuta el bucle). Si la condición es `True`, se ejecuta la proposición y se vuelve a comprobar la condición. La primera vez que la condición resulte ser `False`, se omitirá el bucle `while`.



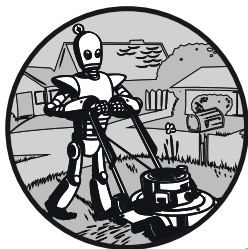
**Figura 3.1.** Diagrama de flujo para el código de la declaración `if`.



**Figura 3.2.** Diagrama de flujo para el código de la declaración `while`.

# 8

## EDICIÓN DE CADENAS Y TEXTO



El texto es una de las formas de datos más habituales que manejarán tus programas. Ya sabes cómo concatenar dos valores de cadena con el operador +, pero podemos hacer mucho más que eso, como extraer cadenas parciales de valores de cadena, añadir o quitar espacios, poner letras en mayúscula o minúscula y comprobar si una cadena tiene el formato correcto. Podemos incluso escribir código Python para acceder al portapapeles que usamos para copiar y pegar texto.

En este capítulo, aprenderás todo esto y más. Después, trabajaremos en un proyecto de programación para automatizar la aburrida tarea de añadir viñetas a un texto.

### Trabajar con cadenas

Echemos un vistazo a algunas de las maneras en las que Python nos permite escribir, imprimir y acceder a cadenas en nuestro código.

### Cadenas literales

Aunque los valores de cadena se guardan en la memoria del programa, los valores de cadena que aparecen literalmente en el código se llaman "cadenas literales". Escribir cadenas literales en código Python parece fácil: empiezan y terminan con una comilla

simple y llevan el texto del valor de cadena entre medio. ¿Pero cómo podemos usar comillas dentro de una cadena? Escribir `'That is Alice's cat.'` no funcionará porque Python pensará que la cadena termina después de `Alice` y tratará el resto (`s cat.'`) como código no válido. Por suerte hay varias formas de escribir cadenas literales. El término "cadena" se refiere a un único valor en el contexto de un programa en ejecución y a una cadena literal cuando estamos hablando de escribir código fuente en Python.

### Comillas dobles

Las cadenas literales pueden empezar y terminar con comillas dobles, igual que con comillas simples. Una ventaja de usar las dobles es que la cadena puede contener una comilla simple dentro. Escribe lo siguiente en el intérprete de comandos interactivo:

```
>>> spam = "That is Alice's cat."
```

Como la cadena empieza y acaba con comillas dobles, Python sabe que la comilla simple forma parte de la cadena y no la corta ahí. No obstante, si necesitas usar comillas simples y dobles dentro de la cadena, tendrás que recurrir a los caracteres de escape.

### Caracteres de escape

Un carácter de escape permite usar caracteres que de otro modo sería imposible poner en una cadena literal. Un carácter de escape consiste en una barra invertida (`\`) seguida del carácter que se quiere añadir a la cadena. Por ejemplo, `\'` es el carácter de escape para una comilla simple y `\n` es el carácter de escape para el carácter de salto de línea. (Pese a consistir en dos caracteres, a menudo se usa el nombre carácter de escape en singular). Puedes usar esta sintaxis dentro de una cadena que empieza y termina con comillas simples. Para ver cómo funcionan los caracteres de escape, escribe lo siguiente en el intérprete de comandos interactivo:

```
>>> spam = 'Say hi to Bob\'s mother.'
```

Python sabe que, como la comilla de `Bob\'s` lleva una barra invertida, no es una comilla que pretenda terminar el valor de cadena. Los caracteres de escape `\'` y `\"` permiten poner comillas simples y dobles, respectivamente, dentro de una cadena.

La tabla 8.1 recoge los caracteres de escape que puedes usar.

**Tabla 8.1.** Caracteres de escape.

Carácter de escape	Se imprime como...
<code>\'</code>	Comilla simple
<code>\"</code>	Comillas dobles
<code>\t</code>	Tabulación
<code>\n</code>	Salto de línea
<code>\\</code>	Barra invertida

Para practicar con esto, escribe lo siguiente en el intérprete de comandos interactivo:

```
>>> print("Hello there!\nHow are you?\nI\'m doing fine.")
Hello there!
How are you?
I'm doing fine.
```

Ten en cuenta que como `\` inicia un carácter de escape, si necesitas poner una barra invertida en tu cadena, debes usar el carácter de escape `\\`.

### Cadenas crudas

Puedes poner una `r` antes de la comilla inicial de una cadena para marcarla como cadena literal cruda. Una cadena cruda hace que sea más fácil introducir valores de cadena que contengan barras invertidas ignorando todos los caracteres de escape. Para ver un ejemplo, pon esto en el intérprete de comandos interactivo:

```
>>> print(r'The file is in C:\Users\Alice\Desktop')
The file is in C:\Users\Alice\Desktop
```

Como se trata de una cadena cruda, Python considera que la barra invertida es parte de la cadena y no el inicio de un carácter de escape:

```
>>> print('Hello...\n\n...world!') # Sin cadena cruda.
Hello...
...world!
```

```
>>> print(r'Hello...\n\n...world!') # Con cadena cruda.
Hello...\n\n...world!
```

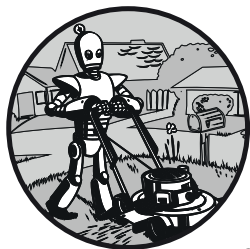
Las cadenas crudas son útiles si los valores de cadena contienen varias barras invertidas, como las utilizadas para las rutas de archivo en Windows, como `r'C:\Users\Al\Desktop'`, o expresiones regulares, que veremos en el siguiente capítulo.

### Cadenas multilínea

Aunque puedes usar el carácter de escape `\n` para insertar un salto de línea en una cadena, suele ser más fácil usar cadenas multilínea. Una cadena multilínea de Python empieza y termina con tres comillas simples o dobles. Cualquier comilla, tabulación o salto de línea que quede entre las "comillas triples" se considera parte de la cadena. Las reglas de sangrado de Python para los bloques no se aplican a las líneas de una cadena multilínea.

# 24

## MOTORES DE TEXTO A VOZ Y RECONOCIMIENTO DE VOZ



Este capítulo trata sobre un paquete de conversión de texto a voz, `pytt3`, y un paquete de reconocimiento de voz, `Whisper`. Los paquetes de texto a voz pueden convertir cadenas de texto en palabras habladas y, después, enviarlas a los altavoces del ordenador o guardarlas en un archivo de sonido. Al añadir esta nueva dimensión a los programas, se libera al usuario de tener que leer el texto en la pantalla. Por ejemplo, una aplicación de recetas de cocina podría leer en voz alta la lista de ingredientes mientras nos movemos por la cocina, y un *script* diario podría recopilar artículos de noticias (o correos electrónicos) y luego preparar un MP3 para reproducirlo durante el trayecto matutino al trabajo.

Por otro lado, las tecnologías de reconocimiento de voz pueden convertir archivos de audio de palabras habladas en valores de cadenas de texto. Podemos utilizar esta capacidad para añadir comandos de voz a nuestros programas o automatizar la transcripción de podcasts. Y, a diferencia de lo que ocurre con los seres humanos, siempre se puede silenciar a un ordenador que habla demasiado.

Tanto `pyttsx3` como `Whisper` son de uso gratuito y no requieren conexión a Internet. Los motores de texto a voz y de reconocimiento de voz que se presentan en este capítulo no se limitan al inglés, sino que funcionan con la mayoría de los idiomas humanos más hablados.

## Motor de texto a voz

Para producir sonido hablado, el paquete de terceros `pyttsx3` utiliza el motor de texto a voz integrado en nuestro sistema operativo: Microsoft Speech API (SAPI5) en Windows, NSSpeechSynthesizer en macOS y eSpeak en Linux. En Linux, es posible que tengas que instalar el motor ejecutando `sudo apt install espeak` desde una ventana del terminal. Puedes instalar `pyttsx3` ejecutando `pip install pyttsx3` desde un terminal. El apéndice A contiene instrucciones completas para instalar paquetes de terceros.

El nombre del paquete viene de `py` para Python, `tts` para texto a voz, `x` porque es una extensión del paquete `pytts` original y `3` porque es para Python 3.

## Generar voz

Producir una voz computarizada es un tema complejo en informática. Por suerte, el motor de texto a voz del sistema operativo hace el trabajo difícil por nosotros y la interacción con este motor es sencilla. Abre el editor de archivos, escribe lo siguiente y guárdalo como `hello_tts.py`:

```
import pyttsx3
engine = pyttsx3.init()
engine.say('Hello. How are you doing?')
engine.runAndWait() # El ordenador habla.
feeling = input('>')
engine.say('Yes. I am feeling ' + feeling + ' as well.')
engine.runAndWait() # El ordenador habla otra vez.
```

Después de importar el módulo `pyttsx3`, llamamos a la función `pyttsx3.init()` para inicializar el motor de voz. Esta función devuelve un objeto `Engine`. Podemos pasar una cadena de texto a su método `say()` para indicar al motor qué decir, pero la voz no saldrá hasta que llamemos al método `runAndWait()`. Este método se bloquea (es decir, no vuelve) hasta que el ordenador haya terminado de decir toda la cadena.

El programa no produce salida de texto porque nunca llama a la función `print()`. En su lugar, deberíamos oír al ordenador decir "Hello. How are you doing?". (Asegúrate de que no está silenciado). El usuario puede escribir una respuesta con el teclado, a la que el ordenador replicará oralmente "Yes. I am feeling <tu respuesta> as well".

El módulo `pyttsx3` permite hacer algunos cambios en la voz del ordenador. Podemos pasar las cadenas `'rate'`, `'volume'` y `'voices'` al método `getProperty()` del objeto `Engine` para ver su configuración actual. Introduce lo siguiente en el intérprete de comandos interactivo:

```
>>> import pyttsx3
>>> engine = pyttsx3.init()
>>> engine.getProperty('volume')
1.0
>>> engine.getProperty('rate')
200
>>> engine.getProperty('voices')
[<pyttsx3.voice.Voice object at 0x0000029DA7FB4B10>,
<pyttsx3.voice.Voice object at 0x0000029DAA3DAAD0>]
```

Ten en cuenta que la salida puede ser distinta en tu ordenador. El ajuste del volumen es un flotante, donde `1.0` indica el 100 %. La voz del ordenador habla a una velocidad de 200 palabras por minuto. Continúa el ejemplo con este código:

```
>>> for voice in engine.getProperty('voices'): # Enumera todas las voces disponibles.
...     print(voice.name, voice.gender, voice.age, voice.languages)
...
Microsoft David Desktop - English (United States) None None []
Microsoft Zira Desktop - English (United States) None None []
```

En mi portátil Windows con idioma inglés (Estados Unidos), `getProperty('voices')` devuelve dos objetos `Voice`. (Fíjate en que la cadena está en plural, `'voices'`, no en singular, `'voice'`). Estos objetos `Voice` tienen atributos `name`, `gender` y `age`, aunque `gender` y `age` están configurados como `None` cuando el sistema operativo no guarda esa información. El atributo `languages` es una lista de los idiomas compatibles con la voz y será una lista vacía si se desconoce esta información.

Sigamos con el ejemplo del intérprete de comandos interactivo llamando al método `setProperty()` para cambiar estos ajustes:

```
>>> engine.setProperty('rate', 300)
>>> engine.setProperty('volume', 0.5)
>>> voices = engine.getProperty('voices')
>>> engine.setProperty('voice', voices[1].id)
>>> engine.say('The quick brown fox jumps over the yellow lazy dog.')
>>> engine.runAndWait()
```

En este ejemplo, hemos cambiado la velocidad a 300 palabras por minuto y hemos bajado el volumen a la mitad pasando `0.5` para `'volume'`. Después, hemos cambiado la voz a la femenina "Zira" que ofrece Windows pasando el atributo `id` del objeto `Voice` en el índice 1 de la lista que devuelve `getProperty('voices')`. Observa también que, para configurar la voz, hemos usado la cadena `'voice'`, en singular, no `'voices'` en plural.

## Guardar sonido en archivos WAV

El método `save_to_file()` del módulo `pyttsx3` puede guardar el sonido generado en un archivo WAV (con extensión `.wav`). Escribe lo siguiente en el intérprete de comandos interactivo:

```
>>> import pytttsx3
>>> engine = pytttsx3.init()
>>> engine.save_to_file('Hello. How are you doing?', 'hello.wav')
>>> engine.runAndWait() # El ordenador crea hello.wav.
```

El primer argumento para `save_to_file()` es una cadena con el discurso que se genera, mientras que el segundo es el nombre del archivo `.wav`. La cadena de texto puede ser una oración corta, como en el ejemplo, o páginas enteras de texto. En mi ordenador, `pytttsx3` fue capaz de convertir una cadena de 1.800 palabras en un archivo de audio de 10 minutos en unos dos segundos. Es importante destacar que solo llamar a `save_to_file()` no es suficiente, también hay que llamar al método `runAndWait()` para que Python cree el archivo `.wav`. El módulo `pytttsx3` puede guardar solo archivos `.wav`, no `.mp3` ni otros formatos de audio.

## Reconocimiento de voz

Whisper es un sistema de reconocimiento de voz que puede reconocer varios idiomas. A partir de un archivo de audio o vídeo, Whisper puede devolver el discurso como texto en una cadena de Python. También devuelve los tiempos de inicio y fin de grupos de palabras, que se pueden utilizar para generar archivos de subtítulos.

Instala Whisper ejecutando `pip install openai-whisper` desde el terminal. (Observa que el nombre del paquete de reconocimiento de voz es `openai-whisper`; el paquete `whisper` del sitio web PyPI es otra cosa). Es una descarga de gran tamaño y la instalación puede tardar varios minutos. Además, la primera vez que llamemos a la función `load_model()`, el ordenador descargará el modelo de reconocimiento de voz, que puede tener un tamaño de cientos de megabytes o más.

Supongamos que tenemos un archivo de audio llamado `hello.wav` en el directorio de trabajo actual. (Whisper también puede manejar `.mp3` y varios otros formatos de audio). Podríamos introducir lo siguiente en el intérprete de comandos interactivo:

```
>>> import whisper
>>> model = whisper.load_model('base')
>>> result = model.transcribe('hello.wav')
>>> print(result['text'])
Hello. How are you doing?
```

Después de importar el módulo `whisper`, hay que cargar el modelo de reconocimiento de voz llamando a la función `whisper.load_model()` y pasándole la cadena del modelo de aprendizaje automático entrenado que queremos usar: `'tiny'`, `'base'`, `'small'`, `'medium'` o `'large-v3'`. (Seguirán saliendo nuevos modelos). Los modelos más pequeños pueden transcribir más deprisa, pero los más grandes transcriben con mayor precisión, incluso cuando el audio tiene sonido ambiente de fondo. La primera vez que cargues un modelo, tu ordenador debe estar conectado a Internet para que el módulo `whisper` pueda descargarlo de los servidores de OpenAI. La

tabla 24.1 recoge los nombres de los modelos como cadenas que puedes pasar a la función `whisper.load_model()`, junto con su tamaño de archivo, uso de memoria y los resultados de algunas pruebas de tiempo de ejecución en mi ordenador portátil.

**Tabla 24.1.** Propiedades de los modelos de reconocimiento de voz de Whisper.

Nombre del modelo	Tamaño de archivo del modelo	Memoria necesaria estimada	Tiempo de ejecución para una muestra de audio de 10 palabras, 3 segundos	Tiempo de ejecución para una muestra de audio de 1.800 palabras, 15 minutos
'tiny'	74 MB	1 GB	1,9 s	1 m 20 s
'base'	142 MB	1 GB	3,0 s	2 m 34 s
'small'	472 MB	2 GB	9,6 s	6 m 37 s
'medium'	1,5 GB	5 GB	28,6 s	20 m 17 s
'large-v3'	3 GB	10 GB	51,9 s	32 m 58 s

Mi opinión personal es que, para el 99 % de los casos, el modelo `'base'` debería ser adecuado y el modelo `'medium'` es suficiente cuando se necesita mayor precisión. Todos los modelos producen errores, por lo que el resultado siempre debe ser revisado por un humano. Puedes probar con modelos más grandes si encuentras muchos errores de transcripción en el texto, o con modelos más pequeños si Whisper tarda demasiado en transcribir el audio. Sin embargo, como se aprecia en la tabla 24.1, hay una diferencia sustancial entre los dos minutos y 34 segundos que tarda el modelo `'base'` en transcribir 15 minutos de audio y los casi 33 que tarda el modelo `'large-v3'`.

Con el objeto `model.whisper` que devuelve `whisper.load_model()`, podemos llamar al método `transcribe()` para realizar la transcripción real. Pasa al método una cadena con el nombre del archivo de audio. Este método tardará entre unos segundos y unas horas en ejecutarse, dependiendo del modelo y la duración del archivo de audio. Whisper puede aceptar cualquier archivo de audio o vídeo y lo convierte automáticamente al formato que requiere.

Whisper puede detectar automáticamente el idioma del audio, pero podemos especificar una lengua pasando un argumento de idioma para transcribir, como `model.transcribe('hello.wav', language='English')`. Para encontrar los idiomas que admite Whisper, podemos ejecutar `whisper --help` desde el terminal. Whisper es bastante bueno (aunque no perfecto) a la hora de adivinar dónde debe insertar la puntuación y poner en mayúscula los nombres propios. Sin embargo, siempre debes revisar el resultado para corregir cualquier error.

El diccionario que devuelve `model.transcribe()` tiene varios pares clave-valor, pero la clave `'text'` contiene la cadena de la transcripción.

Por defecto, Whisper utiliza la CPU para transcribir texto, pero, si el ordenador tiene una tarjeta gráfica 3D, se puede acelerar considerablemente las transcripciones configurándolo para utilizar la unidad de procesamiento gráfico (GPU). Encontrarás

## APRENDE PYTHON, HAZ COSAS



MÁS DE 750 000  
COPIAS VENDIDAS  
EN TODO  
EL MUNDO

Si alguna vez has tenido que pasar horas renombrando archivos o actualizando cientos de celdas en una hoja de cálculo, ya sabes lo tedioso que resulta. ¿Y si te dijera que el ordenador puede hacer todo eso por ti?

En esta tercera edición totalmente revisada de *Automatiza las tareas aburridas con Python*, aprenderás a usar Python para escribir programas que hagan en minutos lo que a ti te llevaría horas, y sin necesidad de haber programado nunca. Los primeros capítulos te enseñarán lo básico de Python mediante explicaciones claras y ejemplos entretenidos. Escribirás tu primer programa de Python; trabajarás con cadenas, listas, diccionarios y otras estructuras de datos y, después, usarás expresiones regulares para buscar y manipular patrones de texto.

Cuando ya domines lo básico, emprenderás proyectos que te enseñarán a usar Python para automatizar tareas como:

### Buscar por Internet, descargar contenido y rellenar formularios.

- Buscar, extraer y manipular texto y datos en archivos y hojas de cálculo.
- Copiar, mover, renombrar o comprimir archivos guardados en tu ordenador.

- Dividir, fusionar y extraer texto de documentos de Word o en PDF.
- Interactuar con aplicaciones con macros personalizadas para ratón y teclado.
- Gestionar tu buzón de entrada, desapuntarte de listas de distribución y enviar correos o notificaciones de texto.

**Novedades en esta edición:** Todos los ejemplos de código se han actualizado. También encontrarás cuatro capítulos nuevos sobre la integración de bases de datos, el reconocimiento de voz y la edición de audio y vídeo, además de 16 nuevos proyectos de programación y mayor cobertura de técnicas de desarrollo, como la creación de programas de línea de comandos.

No pierdas tiempo haciendo lo que un mono bien adiestrado podría hacer. Aunque nunca hayas escrito ni una línea de código, puedes delegar ese trabajo monótono en tu ordenador. *Automatiza las tareas aburridas con Python*.

### SOBRE EL AUTOR

**Al Sweigart** es desarrollador de software, miembro de la Python Software Foundation y autor de numerosos libros de programación populares, incluidos *The Big Book of Small Python Projects*, *Beyond the Basic Stuff with Python*, *Coding with Minecraft* y *The Recursive Book of Recursion* (todos de la editorial No Starch Press).

CUBRE PYTHON 3.X

**ANAYA**  
MULTIMEDIA

[www.anayamultimedia.es](http://www.anayamultimedia.es)



ISBN 978-84-415-5278-4



9 788441 552784

2315285