Aprende a programar con Minecraft

### **Utilizar archivos**

Cuando trabajas con ordenadores, trabajas con archivos todo el tiempo. Cada vez que escribes un documento de texto o algo de código Python y lo guardas, estás trabajando con archivos. El texto, las imágenes, los vídeos y la música son archivos. Incluso este libro se guardó como un archivo de texto mientras lo escribía. Las funciones de manejo de archivos de Python son fáciles de aprender y te permitirán crear archivos, guardar archivos y leer información de archivos para hacer cosas interesantes en Minecraft. Empecemos por lo básico y aprendamos a leer y escribir en un archivo de texto en Python.

#### **Abrir un archivo**

Abrir un archivo es el primer paso cuando trabajas con archivos. Para abrir un archivo en Python, utilizas la función open(), que toma dos argumentos: la ubicación del archivo y sus permisos. La *ubicación* de un archivo es dónde está almacenado en tu ordenador. La pasarás a la función open () en forma de cadena. *Los permisos* del archivo controlan si Python puede o no leer o modificar el archivo.

Para abrir (o crear) un archivo de texto llamado archivosecreto *.txt* en Python, utilizarías el argumento "archivosecreto.txt":

secretFile = open("secretFile.txt", "w")

El segundo argumento, "w", es el argumento de permisos, que especifica lo que el programa puede hacer con el archivo. En este caso, w significa que el programa puede escribir datos en el archivo archivoarchivo\_secreto *.*txt.

Cuando un programa llama a la función open() con un nombre de archivo, Python comprueba primero si ya existe un archivo con ese nombre. Si el archivo existe, Python utilizará su contenido en el programa. Si no existe, Python creará un nuevo archivo con ese nombre.

Si no especificas un *directorio* con el nombre del archivo (una carpeta y un directorio son lo mismo), Python buscará el archivo en el directorio donde se encuentre el programa. Si el archivo está almacenado en un directorio diferente, debes especificarlo en el argumento. Por ejemplo, si secretFile *.txt* estuviera en el directorio *secrets*, el primer argumento sería "/secrets/secretFile.txt":

secretFile = open("/secrets/secretFile.txt", "w")

Si proporcionas un directorio en el argumento y el directorio no existe, o si el archivo no existe, recibirás un mensaje de error.

Hay cuatro opciones para el argumento permisos:

w Significa sólo *escritura*. Los permisos de sólo escritura permiten al programa escribir nuevos datos en el archivo y sobrescribir el contenido que ya está en el archivo, pero el programa no puede leer el contenido del archivo. Si no existe un archivo con el nombre que proporcionas como primer argumento, el programa creará uno nuevo.

r Significa *sólo* lectura. Los permisos de sólo lectura permiten al programa leer el contenido del archivo, pero no pueden modificarlo. Este permiso no puede utilizarse para crear un nuevo archivo.

r+ Significa *lectura y* escritura. Los permisos de lectura y escritura permiten al programa leer y modificar el contenido del archivo. El programa también puede escribir sobre cualquier contenido que ya esté en el archivo. Sin embargo, si el archivo no existe, no se creará uno nuevo; en su lugar, obtendrás un error.

a Significa *añadir*. Los permisos para añadir permiten al programa escribir nuevos datos sólo al final del archivo, dejando intacto el resto del contenido. El programa tampoco puede leer el contenido del archivo. Este permiso puede utilizarse para crear un nuevo archivo.

Hay distintas circunstancias en las que utilizarías cada tipo de permiso. Digamos que escribes unas instrucciones para llegar a una impresionante mina de diamantes que has encontrado, y quieres cargar las instrucciones en Minecraft sin cambiarlas accidentalmente. En ese caso, querrías utilizar el permiso de sólo lectura para asegurarte de que no cambia nada del archivo. Alternativamente, si quieres que alguien pueda añadir datos a un archivo pero no quieres que vea los demás datos almacenados en el archivo, utilizarías el permiso append. Por ejemplo, podrías utilizar append si quieres que tus amigos añadan notas a un diario de viaje compartido, ¡sin dejar que se metan dentro y lean todos tus tesoros secretos!

A continuación, aprenderás a escribir datos en un archivo abierto, y aprenderás a cerrar ese archivo para utilizar esos datos más tarde.

#### **Escribir y guardar un archivo**

La función write( ) escribe datos en un archivo que el programa ha abierto. Es el pan de cada día del trabajo con archivos, porque te permite guardar todo tipo de datos. Proporcionas los datos que quieres escribir en el archivo como argumento de la función write( ).

Por ejemplo, vamos a abrir un fichero y escribir en él una simple cadena:

secretFile = open("secretFile.txt", "w")  
secretFile.write("This is a secret file. Shhh! Don't tell anyone.")  
➊ secretFile.close()

En primer lugar, debes abrir el fichero utilizando la función open( ). A continuación, utiliza la notación por puntos para llamar a la función escribir( ) y escribir una cadena en *archivo secreto.txt*. A continuación, debes llamar a la función close( ) ➊, que guarda y cierra el archivo. Es importante que recuerdes incluir la función close( ); de lo contrario, los datos no se guardarán en el archivo.

Ejecuta el programa y luego abre *secretFile.txt* en un editor de texto para ver si se ha guardado tu mensaje secreto. Intenta cambiar la cadena para escribir algo diferente en el archivo y vuelve a ejecutar el programa. ¿Qué ocurre? ¡El mensaje antiguo debería haber sido sustituido por el mensaje nuevo! Prueba a cambiar de nuevo el mensaje, pero en lugar de pasar "w " pasa en su lugar "a". ¿Qué ocurre ahora? Muy chulo, ¿eh?

#### **Leer un archivo**

La función read( ) lee todo el contenido de un archivo que un programa ha abierto. Puede que quieras utilizar los datos en tu programa, modificarlos y luego devolverlos al archivo, o dar salida a los datos para facilitar su consulta. Sea cual sea el motivo, utilizarás la función read() para leer archivos.

Para leer un fichero, primero debes abrirlo y luego acordarte de cerrarlo cuando hayas terminado. Es importante que aprendas este hábito cuando trabajes con archivos en tus programas, ¡para evitar errores!

Vamos a leer un archivo y luego mostrar su contenido para que podamos ver lo que dice. Este programa, *showSecretFile.py*, muestra el contenido de un fichero utilizando las funciones read() e print():

*showSecretFile.py*

secretFile = open("secretFile.txt", "r")  
  
➊ print(secretFile.read())  
secretFile.close()

En primer lugar, abrimos el archivo y pasamos "r" como argumento de permiso para que nuestro programa pueda leer el archivo. También puedes pasar "r+", pero en este caso no vamos a escribir en el archivo, así que es mejor "r ". Para imprimir el contenido de *archivo\_secreto.txt*, pasamos archivo\_secreto.read() a una sentencia print. Por último, aunque no hayamos escrito ningún dato en el archivo, conviene cerrarlo con la función close().

Ejecuta el programa para ver qué ocurre. El contenido de *archivo\_secreto.txt* debería imprimirse en la pantalla. ¡Ahora puedes leer el archivo sin tener que abrirlo en un editor de texto como harías normalmente!

#### **Leer una línea de un archivo**

Supongamos que tienes un documento de texto largo y quieres ver sólo una parte. Aquí es donde resulta útil la función readline( ). A diferencia de la función read (), que obtiene todo el contenido del archivo, la función readline () obtiene una sola línea del archivo cada vez.

Para probar la función readline (), primero añade un montón de texto a *secretFile.txt*. Puedes hacerlo utilizando un editor de texto o usando tus nuevas habilidades de Python para escribir un montón de información en él. Si utilizas Python para escribir en tu archivo, añade \n a tus cadenas cada vez que quieras una nueva línea. Por ejemplo, si escribes " Guay\nBaile\nFiesta " en un archivo, Python coloca "Guay" en una línea, "Baile" en la siguiente y "Fiesta" en la última, así:

Cool  
Dance  
Party

Cuando hayas añadido el texto a *secretFile.txt*, escribe este código en un archivo Python y guárdalo como *showSecretLines.py* en una nueva carpeta llamada *files*:

*showSecretLines.py*

secretFile = open("secretFile.txt", "r")  
  
print(secretFile.readline())  
print(secretFile.readline())  
print(secretFile.readline())  
  
secretFile.close()

Una vez más, debes abrir secretFile *.txt* antes de poder leer de él utilizando la función readline(). Como quieres que tu programa showSecretLines *.* py lea datos del archivo, debes volver a introducir r (o r+). A continuación, incluye tres sentencias print para imprimir las tres primeras líneas de archivosecreto *.txt*. Por último, vuelve a cerrar el archivo utilizando close().

La función readline () comienza con la primera línea de tu archivo. Cada vez que se utiliza la función readline (), lee automáticamente la línea siguiente. Esta función es muy práctica para imprimir un par de líneas del principio de archivos de texto.

**NOTA**

*La* función readline () *convierte el archivo en una lista de cadenas, donde cada elemento de la lista representa una sola línea. Si quieres imprimir una línea desde la mitad de un documento de texto, ¡podrías escribir un bucle para encontrar e imprimir una cadena concreta de la lista!*

#### **Misión nº 64: Lista de tareas pendientes**

A veces puede que no tengas mucho tiempo libre para jugar a Minecraft. Puede que construyas estructuras complejas a lo largo de varios días en sesiones cortas. A medida que añadas programas para abrir puertas o teletransportar al jugador a algún lugar, tus construcciones se volverán más complejas y podrías tardar más en terminarlas. Trabajar en proyectos a lo largo de varios días puede hacer que olvides lo que estabas haciendo y lo que tienes que hacer a continuación. Esto me ocurre a menudo. Afortunadamente, ¡puedes crear un programa que te ayude a recordar!

Los programas de esta misión crean una lista de tareas pendientes y la muestran en el chat de Minecraft. Puedes utilizar este programa para hacer un seguimiento de tus objetivos en Minecraft, de modo que cuando tengas que dejar de jugar, puedas retomarlo fácilmente donde lo dejaste.

Para hacer la lista de tareas, escribirás dos programas distintos: uno para escribir la lista y otro para mostrarla. Empecemos creando el programa que escribe la lista.

#### **Parte 1: Escribir la lista de tareas pendientes**

En primer lugar, necesitas un programa para crear los elementos de la lista de tareas. [El listado 11-1](ch11.xhtml#ch11ex1) te inicia, utilizando un bucle while y la función input() para añadir elementos a la lista de tareas pendientes. Cópialo en un archivo en IDLE y guárdalo como *inputToDoList.py* en la carpeta de *archivos*.

*inputToDoList.py*

➊ toDoFile =  
  
➋ toDoList = ""  
  
➌ toDoItem = input("Enter a to-do list item: ")  
  
➍ while toDoItem != "exit":  
➎ toDoList = toDoList + toDoItem + "\n"  
toDoItem = input("Enter a to-do list item: ")  
  
➏ # Write the to-do list to the file  
➐ # Close the file

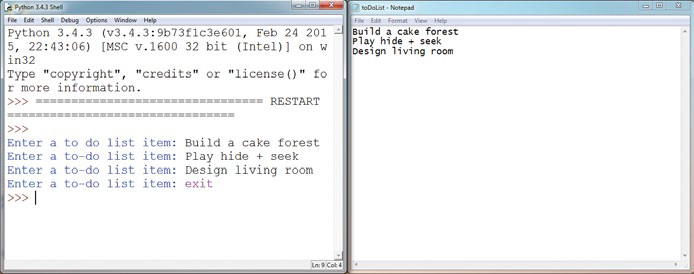
*Lista 11-1: Inicio del programa para escribir elementos en tu lista de tareas pendientes*

El programa crea una cadena vacía llamada toDoList ➋, que almacenará todos los elementos de tu lista de tareas pendientes cuando los introduzcas. Utilizando la función input(), el programa te pide que introduzcas un elemento en la lista de tareas pendientes ➌. A continuación, el bucle while comprueba si la entrada no es igual a "exit " ➍; si no lo es, el programa añade tu elemento a la lista de tareas pendientes con una nueva línea al final utilizando "\n" ➎. Sin embargo, si introduces "exit", el bucle deja de ejecutarse y no podrás añadir más elementos a la lista de tareas pendientes.

Tu misión es terminar el programa. Para ello, necesitas escribir el código que abra el archivo, escriba toDoList en el archivo y, a continuación, cierre el archivo. Utiliza la función open() para abrir el archivo al inicio del programa ➊. Debes abrirlo con permisos de escritura. Nombra el archivo que abre la función *comoHacerLista.txt*. Tu programa creará el archivo si no existe ya en el directorio.

Al final del programa, escribe el contenido de la lista de tareas en el archivo para poder acceder a él más tarde. Utiliza la función write() para escribir la variable toDoList en el archivo toDoFile ➏. Después de escribir en el archivo, asegúrate de cerrarlo con la función close () en la última línea ➐.

[La Figura 11-1](ch11.xhtml#ch11fig1) me muestra escribiendo una lista de tareas con el programa. Cuando termino, escribo exit.



*Figura 11-1: Introduciendo cosas que hacer, como construir un bosque de pasteles y jugar al escondite*

#### **Parte 2: Visualización de la lista de cosas por hacer**

Ahora que tienes un programa para escribir una lista de cosas por hacer en un archivo, necesitas mostrar la lista de cosas por hacer en el chat de Minecraft, línea a línea. El listado [11-2](ch11.xhtml#ch11ex2) inicia el programa por ti. Copia el listado en un nuevo archivo y guárdalo como *outputToDoList.py* en la carpeta de *archivos*.

*outputToDoList.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
➊ toDoList =  
  
for line in toDoList:  
➋ # Output "line" to the chat

*Listado 11-2: Programa para enviar el archivo de la lista de tareas al chat de Minecraft*

[El listado 11-2](ch11.xhtml#ch11ex2) utiliza un bucle for para enviar cada línea del archivo *toDoList.txt* al chat de Minecraft, de una en una. De momento, el programa está incompleto. Para terminar el programa, añade la función open() para abrir el archivo *toDoList.t* xt que creaste con *inputToDoList.py* ➊. Asegúrate de que el archivo tiene permisos de lectura. Después de abrir el archivo, añade código dentro del bucle for para enviar la cadena almacenada en la variable line al chat de Minecraft ➋. Para ello tendrás que utilizar las funciones readline() y postToChat().

[La Figura 11](ch11.xhtml#ch11fig2) -2 muestra mi lista de tareas pendientes en el chat de Minecraft.



*Figura 11-2: Ahora, cuando vuelva a construir, podré ver lo que tengo que hacer.*

[anterior](ch11_1.html)[Subtema 2 de 7: (Ver todo)](ch11.html)[siguiente](ch11_3.html)