Aprende a programar con Minecraft

### **Almacenar muchos datos con el módulo shelve**

El módulo pepinillo sólo puede almacenar un dato cada vez. En algunos programas, puede que quieras almacenar varias variables; si utilizas el módulo pickle, tendrás que crear varios archivos, lo que puede ser difícil de gestionar.

El módulo shelve de Python resuelve este problema. Puede almacenar varios datos en un único archivo. Funciona como un diccionario en el que cada valor de datos tiene una clave que puedes utilizar para almacenar y recuperar los datos. Piensa en shelve como en una estantería: cada compartimento de la estantería almacena distintos valores de datos.

#### **Abrir un archivo con shelve**

Tras importar el módulo shelve, utilizarás su función open() para abrir un archivo. Si el archivo no existe, se creará uno nuevo.

El siguiente código abre el archivo *locationsFile.db* y lo almacena en la variable shelveFile:

import shelve  
shelveFile = shelve.open("locationsFile.db")

La función open( ) sólo toma un argumento, el nombre del archivo. No necesitas especificar los permisos del archivo cuando utilices el módulo shelve, ya que éste concede automáticamente privilegios de lectura y escritura.

Al nombrar un archivo con el módulo shelve, debes incluir la extensión *.* db al final del nombre del archivo. Puedes ver el *.d* b al final de mi archivo *locationsFile.db*.

#### **Añadir, modificar y acceder a elementos con shelve**

El módulo shelve funciona como un diccionario. Para añadir datos al archivo, utilizas corchetes con un nombre de clave para almacenar un valor. Por ejemplo, supongamos que una agente secreta llamada Beatriz está en un submarino, y queremos almacenar la ubicación de Beatriz en el diccionario shelveFichero:

import shelve  
shelveFile = shelve.open("locationsFile.db")  
shelveFile['Beatrice'] = 'Submarine'  
shelveFile.close()

Primero abrimos el archivo. A continuación, damos al diccionario shelveFile una clave "Beatriz" y el valor "Submarino". Esta línea crea un nuevo elemento en el diccionario shelveFile con la clave " Beatriz " y el valor "Submarino ".

A continuación, utilizamos la función close() de shelvepara añadir los nuevos datos al archivo y cerrarlo de forma segura.

Si una clave ya existe en un archivo shelve, este código actualizaría el valor antiguo por el nuevo. Supongamos que Beatrice termina su misión y vuelve al cuartel general. Podrías actualizar la ubicación de Beatrice de esta forma:

import shelve  
shelveFile = shelve.open('locationsFile.db')  
shelveFile['Beatrice'] = 'Headquarters'  
shelveFile.close()

Ahora el valor correspondiente de la clave Beatrice es "Cuartel General". Acceder a un valor desde la estantería también funciona como en un diccionario. Utilizas claves para acceder a valores concretos. Por ejemplo, para imprimir la ubicación de Beatriz, utilizaríamos el siguiente código:

import shelve  
shelveFile = shelve.open('locationsFile.db')  
print(shelveFile['Beatrice'])

Esto mostrará la ubicación de Beatriz, Sede Central.

Al igual que un diccionario estándar, el módulo estantería puede almacenar cualquier tipo de datos, incluidos flotantes, cadenas, booleanos, listas multidimensionales, otros diccionarios, etc. De hecho, en la siguiente misión, ¡almacenarás y accederás a listas multidimensionales para guardar y cargar múltiples estructuras!

#### **Misión nº 66: Guardar una colección de estructuras**

Los programas de esta misión almacenarán y cargarán todas tus estructuras guardadas utilizando un único archivo. Esta misión se divide de nuevo en dos programas: uno para guardar y otro para cargar.

Tendrás que convertir los programas de la Misión nº 65 para que utilicen el módulo shelve en lugar del módulo pickle. También añadirás código para tomar la información del usuario, de modo que éste pueda poner nombre a sus edificios. Abre los archivos *saveStructure.py* y *loadStructure.py* y guárdalos como *saveCollection.py* y *loadCollection.py*.

Como hicimos en la misión anterior, vamos a realizar los cambios en estos programas en dos partes.

#### **Parte 1: Guardar una estructura en una colección**

Aquí se incluye y anota parte del archivo original *saveStructure.* py para ayudarte a identificar dónde realizarás los cambios. Aquí tienes la primera línea y las últimas líneas de *saveCollection*.py:

*saveCollection.py*

➊ import pickle  
  
--snip--  
  
# Name the structure  
➋ structureName = input("What do you want to call the structure?")  
# Store the structure in a file  
➌ pickleFile = open("pickleFile", "wb")  
➍ pickleFile.dump(structure)

Se añade una línea extra al archivo para preguntar cómo quieres llamar a la estructura cuando la guardes con pickle ➋. Por ejemplo, mi versión del programa pregunta "¿Cómo quieres llamar a la estructura?" y puedo responder con algo como "Casa" o "Bosque de pasteles". Asegúrate de llamar a cada nueva estructura por un nombre distinto; si una nueva estructura tiene el mismo nombre que otra, la estructura antigua será sobrescrita por la nueva.

Para cambiar este programa para que utilice el módulo shelve en lugar de pickle, tienes que hacer dos cambios. Primero, cambia la importación del módulo pickle por shelve ➊. En segundo lugar, cambia las últimas líneas del código para utilizar shelve en lugar de pickle. Abre un archivo llamado *structuresFile.db* y guárdalo en una variable llamada shelveFile utilizando la función shelve.open( ) ➌. A continuación, almacena la variable de estructura en un diccionario shelve utilizando la variable structureName para el nombre de la clave del diccionario ➍. Debería quedar algo así: shelveFile [nombredelaestructura] = estructura. Por último, cierra shelveFile en la última línea utilizando close().

#### **Parte 2: Cargar una estructura desde una colección**

Ahora tienes que modificar el archivo *loadCollection.py*. He eliminado la parte central del archivo para ahorrar espacio y para que las partes que tienes que cambiar sean más fáciles de ver:

*loadCollection.py*

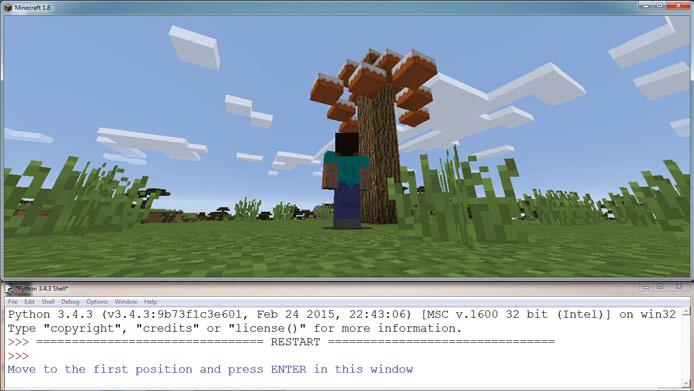
➊ import pickle  
  
--snip--  
➋ structure = pickle.load("pickleFile")  
➌ structureName = input("Enter the structure's name")  
  
pos = mc.player.getTilePos()  
x = pos.x  
y = pos.y  
z = pos.z  
  
➍ buildStructure(x, y, z, structureDictionary[structureName])

He añadido una línea extra al código que pide el nombre de la estructura que quieres construir ➌. Además, he añadido un poco de código a la última línea que obtiene la estructura del diccionario de estanterías y la pasa a la función buildStructure().

Tienes que hacer un par de cambios en este programa. Primero, al igual que en *saveCollection.py*, cambia import por shelve en lugar de pickle ➊. En segundo lugar, carga el archivo shelveFile que creaste en saveCollection *.* py utilizando shelve.open() ➋. Almacena los datos devueltos por la función shelve.open( ) en la variable structureDictionary ➍. El código debería ser algo parecido a lo siguiente: estructuraDiccionario = shelve.load("archivo shelve").

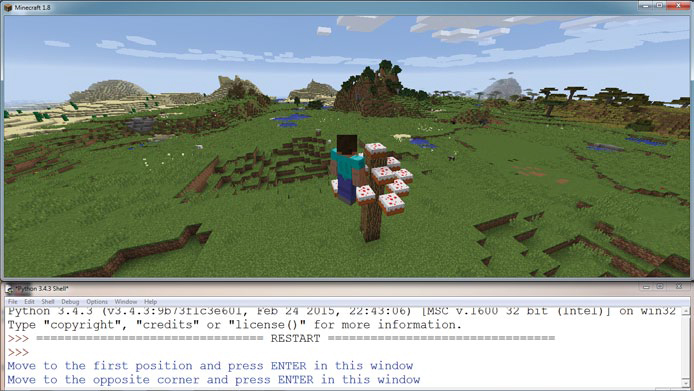
Todos los datos de las estructuras, incluidos sus nombres y bloques, se almacenan en el archivo *structuresFile.db*, lo que significa que no necesitas hacer ningún cambio en *loadCollection.* py antes de ejecutarlo. Todo lo que tienes que hacer es introducir el nombre de la estructura que quieres utilizar cuando ejecutes el programa.

Veamos el programa en acción, utilizando una estructura de mi mundo Minecraft. Primero, copio la estructura utilizando *saveCollection.* py volando hasta una esquina de la estructura y pulsando ENTER[(Figura 11-7](ch11.xhtml#ch11fig7)).



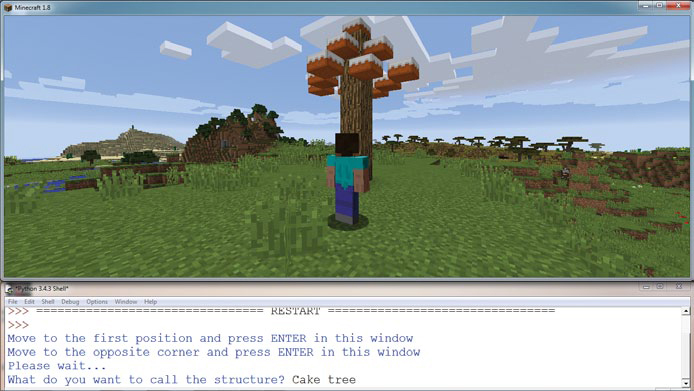
*Figura 11-7: Me desplazo a una esquina de la estructura que quiero guardar.*

Después, vuelo hasta la esquina opuesta de la estructura y pulso ENTER de nuevo[(](ch11.xhtml#ch11fig8)Figura 11-8).



*Figura 11-8: Me desplazo a la esquina opuesta.*

A continuación, el programa me pide que introduzca un nombre para mi estructura. La Figura [11-9](ch11.xhtml#ch11fig9) muestra que he llamado a mi estructura "Árbol de tartas".



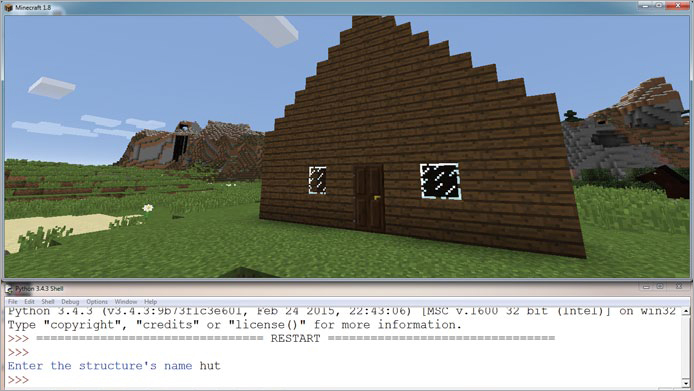
*Figura 11-9: Introduzco el nombre con el que quiero guardar la estructura.*

Por último, ejecuto *loadCollection.py*, vuelo a la ubicación donde quiero construir una copia de la estructura e introduzco el nombre de la estructura que quiero construir[(Figura 11](ch11.xhtml#ch11fig10)-10). El programa empieza a construir delante de mí, ¡como por arte de magia!



*Figura 11-10: Ahora, cuando quiero crear una copia, sólo tengo que introducir el nombre de la estructura y se construye.*

Puedes repetir este proceso con tantos edificios o estructuras como quieras; por ejemplo, yo he hecho una copia de una cabaña en la [Figura 11-11](ch11.xhtml#ch11fig11). Cuando hayas copiado una estructura una vez, podrás cargarla cuando quieras con sólo ejecutar *loadCollection.py* e introducir el nombre de la estructura.



*Figura 11-11: Puedes utilizar el programa para guardar varias estructuras. Aquí he copiado una cabaña.*

[anterior](ch11_3.html)[Subtema 4 de 7: (Ver todo)](ch11.html)[siguiente](ch11_5.html)