Aprende a programar con Minecraft

### **Utilizar listas**

Hacer una lista con Python es sencillo. Para definir una lista, pon entre corchetes cualquier número de valores, o ningún valor, lo que se denomina lista *vacía*. Cada elemento de una lista debe ir separado por una coma.

Por ejemplo, una lista de ingredientes para una sopa de fideos podría tener este aspecto:

>>> noodleSoup = ["water", "soy sauce", "spring onions", "noodles", "beef"]

La lista sopa de fideos contiene varios elementos y todos ellos son cadenas. Puedes crear una lista vacía como ésta:

>>> emptyList = []

Utiliza una lista vacía cuando quieras añadir valores más adelante en tu programa. Puedes almacenar cualquier tipo de datos en tu lista e incluso mezclar distintos tipos de datos. Por ejemplo, puedes tener una lista que contenga enteros y cadenas:

>>> wackyList = ["cardigan", 33, "goofballs"]

A veces tus listas serán muy largas, lo que dificultará su lectura por los humanos. Pero puedes formatear listas largas en varias líneas en Python para que los programadores puedan leerlas fácilmente. Utilizar varias líneas para los elementos no afecta a la ejecución del código Python. Por ejemplo, el siguiente formato para los ingredientes de la sopa funciona igual que la lista anterior de sopa de fideos:

>>> noodleSoup = ["water",  
"soy sauce",  
"spring onions"  
"noodles",  
"beef"]

A continuación veremos cómo puedes acceder a los elementos de una lista y modificarlos.

#### **Acceder a un elemento de la lista**

Para acceder a un valor de una lista, haz referencia a la posición del elemento en la lista, lo que se conoce como su *índice*. Utilizando el ejemplo de la sopa de fideos, puedes acceder al primer elemento de la lista así:

>>> print(noodleSoup[0])  
water

Es importante tener en cuenta que el primer índice de una lista es 0. El segundo elemento es el índice 1, el tercero es el índice 2, y así sucesivamente. La razón es que los ordenadores cuentan desde cero cuando utilizan listas.

Contar desde cero puede parecer una tontería, pero hay una buena razón para ello. Los primeros ordenadores eran muy lentos y tenían muy poca memoria. Era más rápido y eficaz empezar a contar los índices desde cero. Aunque hoy en día los ordenadores son mucho más rápidos, siguen contando desde cero.

También es importante tener en cuenta que si intentas acceder a un índice de lista que sea mayor que el número de elementos de la lista, recibirás un mensaje de error. La siguiente línea intenta imprimir el elemento en la posición 5 del índice:

>>> print(noodleSoup[5])

Aquí tienes parte del mensaje de error:

IndexError: list index out of range

El IndexError me dice que no hay datos en el índice al que quiero acceder. La posición 5 del índice de la lista no tiene datos porque está fuera de la longitud de la lista. ¡Python no puede devolver un valor que no existe!

#### **Cambiar un elemento de la lista**

Al igual que puedes cambiar el valor de las variables, también puedes cambiar elementos individuales de las listas. Esto se debe a que las listas son *mutables*, lo que significa que pueden modificarse. Para cambiar un elemento de una lista, utiliza la posición del índice del elemento y establece su valor del mismo modo que establecerías el valor de una variable (utilizando un signo igual).

Cambiemos el elemento ternera de la sopa de fideos por pollo. La ternera es el quinto elemento de la lista, por lo que tiene un índice de 4 (recuerda que en las listas se cuenta desde cero). Podemos cambiar fácilmente el elemento 4 por pollo, así:

>>> noodleSoup[4] = "chicken"

Ahora vamos a hacer algo guay con las listas en Minecraft.

#### **Misión nº 47: Alto y Bajo**

Cuando exploro el mundo de Minecraft, es interesante repasar mi viaje. Desde las montañas más altas hasta las cuevas más bajas, la exploración es una de mis actividades favoritas en el juego. A veces, cuando juego con amigos, hacemos carreras para ver quién llega más rápido al punto más alto o más bajo del juego. Para que nadie haga trampas, escribí un programa que almacena las coordenadas y más bajas y más altas que alcanza el jugador en 60 segundos.

Cuando ejecuto el programa, me dice los lugares más alto y más bajo del juego a los que he viajado durante un minuto. [El listado 9-1](ch09.xhtml#ch9ex1) contiene el código que he iniciado para ti. Cópialo en un nuevo archivo y guárdalo como *altoYbajo.py* en una nueva carpeta llamada *listas*.

*highAndLow.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
import time  
  
➊ heights = [100, 0]  
count = 0  
  
while count < 60:  
pos = mc.player.getTilePos()  
  
if pos.y < heights[0]:  
➋ # Set the lowest height to the y variable  
elif pos.y > heights[1]:  
➌ # Set the highest height to the y variable  
  
count += 1  
time.sleep(1)  
  
➍ mc.postToChat("Lowest: ") # Output lowest height  
➎ mc.postToChat("Highest: ") # Output highest height

*Listado 9-1: Inicio del código para obtener las posiciones más baja y más alta que visita el jugador*

El programa almacenará las coordenadas y más bajas y más altas a las que ha viajado en una lista llamada alturas ➊. El primer elemento de la lista (posición índice 0) almacena la coordenada más baja y el segundo (posición índice 1) almacena la más alta. Tenemos que empezar con un valor "más bajo" alto y un valor "más alto" bajo para que la primera vez que ejecutemos el programa, la posición del jugador sea el nuevo valor más bajo o más alto y se muestre en el chat. Aquí he utilizado por defecto un valor mínimo de 100 y un valor máximo de 0.

El bucle while se ejecuta una vez por segundo durante 60 segundos para actualizar constantemente los valores de las alturas. La sentencia if comprueba si la altura actual del jugador es inferior al valor más bajo almacenado en la lista ➋. A continuación, la sentencia elif comprueba si la altura actual es mayor que la posición más alta almacenada en la lista ➌.

Para completar el código, tienes que establecer el valor de la altura más baja, altura[0], al valor de pos.y en ➋. Recuerda que estableces los valores en listas como lo harías con una variable, por lo que la línea de código debería tener este aspecto: altura[0 ] = pos.y. También tienes que establecer la altura más alta, altura[1], al valor de pos.y ➌.

Por último, tienes que mostrar el valor de las alturas ➍ más baja y ➎ más alta en las dos últimas líneas del programa. Para ello, tendrás que acceder a las posiciones de los índices de las alturas más baja y más alta de la lista de alturas (de nuevo, el índice 0 es la altura más baja y el índice 1 es la altura más alta).

Ejecuta el programa y empieza a correr por el juego. Comprueba lo alto y lo bajo que puedes llegar. Después de 60 segundos, el bucle se detendrá y el programa mostrará tus alturas más alta y más baja. Ejecuta el programa varias veces y comprueba si puedes batir tu récord.

[La Figura 9-1](ch09.xhtml#ch9fig1) muestra uno de mis intentos.



*Figura 9-1: La coordenada y más baja que visité fue 15 y la más alta 102.*

**OBJETIVO EXTRA: UN ERROR INESPERADO**

En *highAndLow.py*, los valores por defecto para las posiciones más baja y más alta se establecen en 100 y 0. Esto no es un problema siempre que vayas más abajo de 100 y más arriba de 0. Sin embargo, si no vas más abajo de 100 y más arriba de 0, el valor no cambiará, lo que puede hacer que el programa sea impreciso. ¿Puedes solucionar esto?

[anterior](ch09_1.html)[Subtema 2 de 8: (Ver todo)](ch09.html)[siguiente](ch09_3.html)