Aprende a programar con Minecraft

### **Un simple bucle for**

Un bucle for repite un bloque de código para cada elemento de una lista hasta que ésta termina, en lugar de utilizar una condición como un bucle while o una sentencia if.

La lista que utilices en una sentencia for puede contener cualquier número de elementos de cualquier tipo de datos. El bucle for recorrerá cada uno de ellos en orden, es decir, por su índice. Por ejemplo, para imprimir todos los elementos de la lista de la sopa de fideos, utilizaríamos el código siguiente:

noodleSoup = ["water", "soy sauce", "spring onions", "pepper", "noodles",  
"beef", "vegetables"]  
  
for ingredient in noodleSoup:  
print(ingredient)

Utilizamos el operador for para indicar a Python que estamos utilizando un bucle.

Tras el operador for hay una variable, ingrediente, que representa el elemento que el bucle está utilizando en ese momento. El valor cambia cada vez que el bucle itera hasta que ha recorrido cada elemento de la lista. La primera vez que se ejecute el bucle, el valor será el elemento en la posición 0 del índice (en este caso "agua"), la segunda vez el valor será el elemento en el índice 1 ("salsa de soja"), la tercera vez el valor será el elemento en el índice 2 ("cebolletas"), y así sucesivamente.

El operador in y el nombre de la lista al final de la sentencia indican a Python qué lista estás utilizando. En este ejemplo, el nombre de la lista es sopa de fideos.

El bucle se ejecuta una vez por cada elemento de la lista y termina cuando llega al final de la lista. Aquí tienes la salida de este programa:

water  
soy sauce  
spring onions  
pepper  
noodles  
beef  
vegetables

Se imprimen todos los elementos de la lista. Ahora vamos a divertirnos un poco con los bucles for en Minecraft.

#### **Misión nº 55: Varita mágica**

Cada herramienta en Minecraft tiene su propio propósito. La pala cava tierra, el pico rompe bloques de piedra, el hacha corta madera y la espada golpea a los malos. Normalmente, no puedes cambiar cómo se comportan las herramientas; sólo tienes que aceptar que la espada sólo golpea a los enemigos. Pero con Python, puedes cambiar cómo funcionan las herramientas. En este programa, convertiremos la espada en una varita mágica.

En el [Capítulo 9](ch09.xhtml#ch09), aprendiste sobre la función pollBlockHits(). Esta función devuelve una lista de coordenadas de bloque que la espada ha golpeado. Utilizando un bucle for, puedes acceder a cada conjunto de coordenadas de esta lista. Vamos a convertir en melones todos los bloques que hayamos golpeado en los últimos 60 segundos. Puedes ver cómo funciona en la [Figura 10-1](ch10.xhtml#ch10fig1).



*Figura 10-1: ¡Abracadabra! Todos los bloques que golpeo son ahora melones.*

[El listado 10-1](ch10.xhtml#ch10ex1) contiene el inicio del programa. Guárdalo como *varita mágica.py* en una nueva carpeta llamada *forLoops*.

*magicWand.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
import time  
  
time.sleep(60)  
  
➊ hits = mc.events.pollBlockHits()  
block = 103  
  
➋ for  
➌ x, y, z = hit.pos.x, hit.pos.y, hit.pos.z  
➍ # Set melon blocks at the coordinates

*Listado 10-1: El inicio del programa varita mágica*

Para obtener la lista de éxitos de bloque, llamamos a la función pollBlockHits() y almacenamos el resultado en la variable éxitos ➊.

Se incluye una línea de código que obtendrá la posición de cualquier bloque que golpees y almacenará sus coordenadas en las variables x, y, y z ➌. Utiliza una tupla (introducida en "[Tuplas](ch09.xhtml#ch09lev1sec04)" en [la página 179](ch09.xhtml#page_179)) para asignar las tres variables en una sola línea.

De momento, esta línea de código no funcionará porque la variable hit no existe. Crea un bucle for en ➋ y llama hit a la variable del bucle for. El bucle for debe iterar sobre la lista de hits. El código de la primera parte del bucle for debería tener este aspecto:

for hit in hits:

Asegúrate de sangrar la línea de código que obtiene los valores x, y, y z dentro del bucle for en ➌. En la última línea del bucle for, añade la función setBlock() para establecer un bloque de melón en las coordenadas x, y y z ➍.

Cuando el usuario ejecute el programa terminado, tendrá 60 segundos para correr y hacer clic con el botón derecho del ratón en todos los bloques que pueda con su espada. Pasados los 60 segundos, todos los bloques que hayan sido golpeados con la espada se convertirán en melones.

**OBJETIVO EXTRA: ERES UN MAGO**

Cambia el programa *magicWand.py* para que teletransporte al jugador: el primer golpe fija la ubicación, y el segundo lo lleva allí.

#### **La función range()**

La función range () crea una lista de números enteros. Es una buena forma de crear rápidamente una lista de números para tus bucles for. Echemos un vistazo y pasemos dos argumentos, 0 y 5, a la función range( ):

aRange = range(0, 5)

Esta es una forma más rápida de crear una lista que escribir cada elemento de la lista individualmente, lo que tendría este aspecto:

aRange = [0, 1, 2, 3, 4]

Observa que el segundo argumento de la función range () es 5, pero el último elemento de la lista es 4. Esto se debe a que la función sólo crea valores que sean menores pero no iguales que el segundo argumento.

Para crear un bucle que utilice la función rango () para imprimir los números del 1 al 15, utilizarías el siguiente código:

for item in range(1, 16):  
print(item)

Así podrías imprimir el doble del valor de cada elemento de una lista:

for item in range(1, 16):  
print(item \* 2)

Puedes hacer lo mismo con un bucle while, que conociste en el [Capítulo 7](ch07.xhtml#ch07). El siguiente código utiliza un bucle while en lugar de un bucle for para imprimir los números del 1 al 15:

count = 1  
while count < 16:  
print(count)  
count += 1

Observa que el bucle for es más sencillo y fácil de leer. En programas grandes y complejos, un bucle for suele ser mejor opción que un bucle while con count.

#### **Misión nº 56: Escaleras mágicas**

Una de las mejores características de usar Minecraft con Python es que puedes construir cosas rápidamente con sólo unas pocas líneas de código. En lugar de pasar mucho tiempo construyendo muros, puedes ejecutar un poco de código y listo. También puedes reutilizar el código tantas veces como quieras, ahorrando tiempo y esfuerzo.

Construir escaleras es una tarea que suele llevar mucho tiempo. Afortunadamente, con unas pocas líneas de código Python, puedes crear rápidamente una escalera en Minecraft. En esta misión, utilizarás un bucle for para hacer aparecer una escalera en el mundo del juego.

[El listado 10-2](ch10.xhtml#ch10ex2) crea una escalera en Minecraft utilizando un bucle while. Guárdalo como *escaleras.py* en la carpeta *forLoops*.

*stairs.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
pos = mc.player.getTilePos()  
x, y, z = pos.x, pos.y, pos.z  
  
stairBlock = 53  
  
step = 0  
while step < 10:  
mc.setblock(x + step, y + step, z, stairBlock)  
step += 1

*Listado 10-2: Programa que* crea una escalera utilizando un *bucle* while

Aunque puedes utilizar un bucle while para este programa, como se muestra aquí, en realidad es más adecuado un bucle for. A diferencia del bucle while, un bucle for no requiere una variable count o step. En su lugar, puedes utilizar la función range() para determinar cuántas veces se repite el bucle.

Para completar el programa, cambia el código para que utilice un bucle for en lugar de un bucle while.

Puedes ver el resultado del programa en la [Figura 10-2](ch10.xhtml#ch10fig2).

**OBJETIVO EXTRA: ¿BAJAR?**

De momento, el programa *escaleras.* py sólo construye escaleras en una dirección. Intenta averiguar cómo construir escaleras en otras direcciones. Pista: Utilizarás el argumento opcional estados del bloque en la función setBlock() y añadirás o quitarás variables x o z.



*Figura 10-2: ¿Adónde llevará tu escalera mágica?*

#### **Jugando con range()**

Has aprendido un poco sobre la función range( ) y lo que ocurre cuando pasas dos argumentos a la función. ¿Y si le pasas un solo argumento? Introduce este código en el intérprete de comandos IDLE para ver qué ocurre:

>>> aRange = range(5)  
>>> list(aRange)  
[0, 1, 2, 3, 4]

Cuando le das a la función rango ( ) un solo argumento, empezará en 0 y almacenará cada valor hasta uno menos que el valor que le pases como argumento. En otras palabras, es como si pasaras 0 como primer argumento y 5 como segundo argumento. En este ejemplo, la función list( ) muestra los valores de la lista creada por la función range() (de lo contrario, ¡no los verías!). Como puedes ver, el valor de list(aRango) es una lista de cinco números que empiezan en 0: [0, 1, 2, 3, 4]. Ésta es una forma rápida de crear un rango si quieres empezar con 0 como primer valor.

Como has visto, cuando pasas dos argumentos a range(), la lista comienza en el primer argumento proporcionado y termina antes del segundo argumento:

>>> aRange = range(2, 5)  
>>> list(aRange)  
[2, 3, 4]

Este ejemplo crea un rango equivalente a la lista [2, 3, 4].

Cuando das tres argumentos a range( ), el tercer argumento define el *paso* entre elementos. Normalmente, cada valor de la lista creada por la función range ( ) es uno mayor que el valor anterior. Al cambiar el paso, cambias la diferencia entre valores. Por ejemplo, un paso de 2 haría que el siguiente valor de una lista fuera 2 más que el elemento anterior. Un paso de 3 lo haría 3 más que el elemento anterior, y así sucesivamente.

Por ejemplo, esta lista suma 2 al valor anterior para obtener el siguiente valor:

>>> aRange = range(3, 10, 2)  
>>> list(aRange)  
[3, 5, 7, 9]

Observa que cada elemento es 2 más que el elemento anterior (5 es 3 + 2, 7 es 5 + 2 y 9 es 7 + 2).

Incluso puedes dar a range( ) un valor de paso negativo, como éste:

>>> newRange = range(100, 0, -2)  
>>> list(newRange)  
[100, 98, 96, 94, 92, 90, 88, 86, 84, 82, 80, 78, 76, 74, 72, 70, 68, 66, 64,  
62, 60, 58, 56, 54, 52, 50, 48, 46, 44, 42, 40, 38, 36, 34, 32, 30, 28, 26,  
24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 4, 2]

Observa que los valores de la lista disminuyen en 2 debido al valor de paso negativo.

[anterior](ch10_1.html)[Subtema 2 de 7: (Ver todo)](ch10.html)[siguiente](ch10_3.html)