Aprende a programar con Minecraft

### **Otras funciones de lista**

Como estamos trabajando con listas, vamos a explorar algunas otras funciones diseñadas para interactuar con listas.

La función invertida( ) toma un argumento, la lista que quieras utilizar, y devuelve la lista invertida. El último elemento será el primero, el penúltimo será el segundo, y así sucesivamente. Vamos a invertir una lista anterior:

>>> backwardsList = reversed(aRange)  
>>> list(backwardsList)  
[9, 7, 5, 3]

Los elementos de la lista se han invertido, tal y como queríamos. Este tipo de manipulación de listas resulta útil cuando escribes bucles for.

El siguiente ejemplo genera una lista de números del 1 al 100 utilizando la función range(). A continuación, invierte la lista y la imprime utilizando un bucle for, creando de hecho una cuenta atrás del 100 al 1:

countDown = range(1, 101)  
countDown = reversed(countDown)  
for item in countDown:  
print(item)

¡Ejecútalo para ver el resultado!

100  
99  
98  
97  
96  
--snip--  
3  
2  
1

También puedes invertir la lista cuando declares el bucle for sin necesidad de una variable para almacenar la lista:

for item in reversed(range(0, 101)):  
print(item)

Este programa requiere menos líneas de código, pero tiene el mismo efecto. ¡Utiliza este truco para ahorrar tiempo y poder centrarte en construir!

#### **Misión nº 57: Pilares**

¿No sería genial construir un palacio en Minecraft? Como los palacios deben ser grandiosos, el nuestro debería tener filas de pilares altos e imponentes. Obviamente, no queremos construirlos a mano, así que utilizar un bucle para construirlos es la mejor solución.

Crearemos una función que construya un pilar y luego llamaremos a la función cuando queramos construir uno. [El listado 10-3](ch10.xhtml#ch10ex3) contiene la función para construir un pilar. Cópiala en un nuevo archivo llamado *pilares.py* y guárdalo en la carpeta *forLoops*.

*pillars.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
def setPillar(x, y, z, height):  
""" Creates a pillar. Args set position and height of pillar """  
stairBlock = 156  
block = 155  
  
# Pillar top  
mc.setBlocks(x - 1, y + height, z - 1, x + 1, y + height, z + 1, block, 1)  
mc.setBlock(x - 1, y + height - 1, z, stairBlock, 12)  
mc.setBlock(x + 1, y + height - 1, z, stairBlock, 13)  
mc.setBlock(x, y + height - 1, z + 1, stairBlock, 15)  
mc.setBlock(x, y + height - 1, z - 1, stairBlock, 14)  
  
# Pillar base  
mc.setBlocks(x - 1, y, z - 1, x + 1, y, z + 1, block, 1)  
mc.setBlock(x - 1, y + 1, z, stairBlock, 0)  
mc.setBlock(x + 1, y + 1, z, stairBlock, 1)  
mc.setBlock(x, y + 1, z + 1, stairBlock, 3)  
mc.setBlock(x, y + 1, z - 1, stairBlock, 2)  
  
# Pillar column  
mc.setBlocks(x, y, z, x, y + height, z, block, 2)  
  
pos = mc.player.getTilePos()  
x, y, z = pos.x + 2, pos.y, pos.z  
  
➊ # Add the for loop here  
➋ # Call the function here

*Listado 10-3: Función que crea un pilar*

La función setPillar( ) crea un pilar. Toma cuatro argumentos: las coordenadas x, y, z y la altura del pilar.

Para terminar el programa, añade un bucle for ➊ que llame a la función setPillar () ➋. Queremos crear una fila de 20 pilares separados entre sí 5 manzanas. Para ello, utiliza una función range () con tres argumentos para determinar cuántos pilares se crearán y a qué distancia estarán. Añadiendo los valores almacenados en la variable del bucle for a la variable x o z en la llamada a la función setPillar(), puedes hacer que cada pilar esté a la misma distancia.

[La Figura 10-3](ch10.xhtml#ch10fig3) muestra algunos de los pilares.



*Figura 10-3: Una brillante fila de pilares*

#### **Misión nº 58: Pirámide**

Siguiendo con el tema de construir cosas increíbles con bucles for, vamos a construir una pirámide. Una pirámide se compone de muchos niveles. El nivel inferior es el más ancho, y el superior -la cúspide- es el más estrecho. Cada nivel es un cuadrado de bloques. Haremos una pirámide que sea dos bloques más estrecha en cada nivel. Por ejemplo, si el nivel base tiene siete bloques de ancho, el siguiente nivel tendría cinco bloques de ancho, luego tres bloques y, por último, el nivel superior tendría un bloque de ancho.

[El listado 10-4](ch10.xhtml#ch10ex4) crea una pirámide. Cópialo en un nuevo archivo llamado *pirámide.py* y guárdalo en la carpeta *forLoops*.

*pyramid.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
block = 24 # sandstone  
➊ height = 10  
➋ levels = range(height)  
  
pos = mc.player.getTilePos()  
➌ x, y, z = pos.x + height, pos.y, pos.z  
  
➍ for level in levels:  
➎ mc.setBlocks(x - level, y, z - level, x + level, y, z + level, block)  
y += 1

*Listado 10-4: Un programa pirámide al revés*

Aunque [el Listado 10-4](ch10.xhtml#ch10ex4) crea una pirámide, ¡contiene un pequeño error que debes corregir! Almacenamos la altura de la pirámide en la variable altura ➊. Puedes cambiar el valor de la variable altura por el que quieras. La variable niveles utiliza la función range() para crear una lista que contenga un elemento por cada nivel de la pirámide ➋. La variable altura se añade a la coordenada x del jugador cuando establecemos las variables x, y y z ➌. Si no hiciéramos esto, el jugador quedaría atrapado en el centro de la pirámide cuando se construyera.

El bucle for itera por cada nivel de la lista de niveles ➍. La línea de código que crea cada nivel de la pirámide utiliza la variable nivel para calcular la anchura de cada cuadrado de bloques que crea ➎. La anchura y la longitud de cada nivel de la pirámide serán siempre el doble del tamaño de la variable de nivel.

¿Recuerdas el error que he mencionado antes? Ejecuta el programa para ver cuál es el problema. ¡La pirámide está al revés!

Para solucionar este problema y hacer que la pirámide esté al derecho, tendrás que utilizar la función invertida() en la variable niveles para hacer una lista que se haga más pequeña con el tiempo. O puedes ser astuto y llamar a la función range () con un valor negativo.

La[Figura 10-4](ch10.xhtml#ch10fig4) muestra la pirámide terminada.



*Figura 10-4: Una magnífica pirámide*

[anterior](ch10_2.html)[Subtema 3 de 7: (Ver todo)](ch10.html)[siguiente](ch10_4.html)