Aprende a programar con Minecraft

### **Prácticos trucos matemáticos**

En las siguientes secciones, te enseñaré dos habilidades matemáticas más para que mejores tu programación en Python, y luego combinaremos lo que has aprendido hasta ahora en una misión más.

#### **Operadores abreviados**

Muy a menudo, querrás utilizar un operador en una variable y luego almacenar el resultado en la misma variable. Por ejemplo, puede que quieras añadir cinco ovejas a tu rebaño actual:

sheep = 6  
sheep = sheep + 5

Pero escribir oveja = oveja + 5 probablemente te cansará después de un rato. No te preocupes, ¡hay una forma más corta! Python tiene operadores *abreviados* que te permiten utilizar un operador matemático en una variable y reasignar el resultado a la misma variable. Aquí tienes los cuatro operadores abreviados:

- Suma(+=)

- Resta(-=)

- Multiplicación(\*=)

- División(/=)

Por ejemplo, puedes reescribir el ejemplo de la oveja utilizando el operador abreviado de suma:

sheep = 6  
sheep += 5

El valor de oveja es igual a 11, igual que antes.

#### **Jugar con números aleatorios**

Utilizar números aleatorios es una forma de añadir algo de misterio y diversión a tus programas. ¡Nunca sabes con qué acabarás! Muchos juegos de mesa del mundo real se basan en números aleatorios: piensa en cuántos juegos has jugado en los que tenías que tirar los dados para ver cuántos espacios podías mover. Una tirada de dados es un ejemplo clásico de números aleatorios en acción.

Python puede generar números aleatorios por ti fácilmente, así que vamos a simular la tirada de un dado. El número generado debe estar entre 1 y 6:

➊ import random  
➋ diceValue = random.randint(1, 6)

Cuando quieras crear números aleatorios, asegúrate de incluir import random ➊ al principio de tu programa. La función randint() ➋ genera un valor entero que el programa puede utilizar como cualquier otro número. Los números que colocas dentro de los paréntesis como argumentos indican a randint () que genere valores entre el primer número y el segundo. En este ejemplo, el número generado puede tener un valor de 1, 2, 3, 4, 5 ó 6.

Puedes utilizar randint () para añadir un número aleatorio al valor de una variable, e incluso puedes generar números negativos. ¡Veamos cómo!

import random  
score = 0  
score += random.randint(0, 99)  
points = random.randint(-99, 99)

El número más bajo que Python puede generar para añadir a la puntuación es 0, y el más alto es 99. Por otra parte, debido al argumento negativo, ¡los puntos podrían establecerse en un número tan bajo como -99!

#### **Misión nº 10: ¡Super Salta a algún sitio nuevo!**

En la última misión de este capítulo, harás que el jugador salte una distancia aleatoria en los ejes x, y y z almacenando la posición actual del jugador y añadiendo después un número aleatorio a cada una de las tres coordenadas. Utiliza los números aleatorios entre -10 y 10 para cambiar los valores x y z, y asegúrate de que los valores aleatorios para y están entre 0 y 10.

Para empezar, copia [el Listado 3-10](ch03.xhtml#ch3ex10) en un nuevo archivo en IDLE y guárdalo como *randomJump.py*.

*randomJump.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
import random  
  
pos = mc.player.getPos()  
x = pos.x  
y = pos.y  
z = pos.z  
  
➊ x = x + random.randint(-10, 10)  
mc.player.setPos(x, y, z)

*Listado 3-10: El programa de salto aleatorio incompleto*

Falta el código de generación de números aleatorios para las variables y y z, y depende de ti añadirlo. Una vez que lo hagas, el jugador tendrá la posibilidad de saltar a cualquier parte, como hice yo en la [Figura 3](ch03.xhtml#ch3fig10)-10. ¡Deja que el azar te lleve a lugares nuevos y emocionantes!

De momento, el código no utiliza operadores abreviados para cambiar los valores de las variables. Prueba a cambiar la expresión de suma en ➊ para que utilice taquigrafía en su lugar.



*Figura 3-10: He saltado en una dirección aleatoria y he acabado en lo alto de este árbol. ¿Dónde has acabado tú?*

**OBJETIVO EXTRA: TELETRANSPORTE DE BLOQUES ALEATORIOS**

¡Hagamos que el programa *randomJump.* py sea aún más aleatorio! Después de que el jugador salte a una posición aleatoria, coloca un bloque aleatorio debajo de él. También podrías adaptar el programa de recorrido de teletransporte de la Misión 1 (ver *teletransporte*.py en [la página 38](ch02.xhtml#page_38)) para que el jugador se teletransporte a una posición aleatoria cada vez. Si accidentalmente te teletransportas a algún lugar y te quedas atascado, siempre puedes volver a ejecutar teleport *.*py, que debería teletransportarte a un lugar seguro.

[anterior](ch03_5.html)[Subtema 6 de 7: (Ver todo)](ch03.html)[siguiente](ch03_7.html)