Aprende a programar con Minecraft

### **Tuplas**

*Las tuplas* son un tipo de lista inmutable. Pero, al igual que otras listas, son una secuencia de elementos de cualquier tipo variable. Las tuplas utilizan paréntesis en lugar de corchetes, y usan comas para separar los elementos.

Por ejemplo, supongamos que el único atleta olímpico de una nación, procedente de un programa de entrenamiento con escasa financiación, registra una serie de distancias para sus saltos de longitud en metros:

>>> distance = (5.17, 5.20, 4.56, 53.64, 9.58, 6.41, 2.20)

Si el atleta saltara sólo una vez, también podrías crear una tupla con un único valor. Para escribir una tupla con un único valor, debes incluir una coma:

>>> distance = (5.17,)

Cuando defines una tupla, los paréntesis son opcionales, así que puedes definir una tupla simplemente colocando comas entre los valores, de esta forma:

>>> distance = 5.17, 5.20, 4.56, 53.64, 9.58, 6.41, 2.20

Para acceder a los valores de las tuplas, utiliza la notación de corchetes que utilizas con las listas normales. Asignemos el valor del índice 1 de la tupla distancia a la variable salto:

>>> jump = distance[1]  
>>> print(jump)  
5.20

La principal diferencia entre las listas y las tuplas es que las tuplas son inmutables: no puedes cambiar su contenido. No puedes añadir elementos al final de la tupla, insertar elementos, borrar elementos ni actualizar ningún valor. Utiliza tuplas en lugar de listas cuando tu programa no necesite cambiar los valores de los elementos de la tupla.

#### **Establecer variables con tuplas**

Una característica útil de las tuplas es que puedes utilizarlas para establecer más de una variable al mismo tiempo. Esto ahorra espacio y puede mantener agrupadas las variables relacionadas.

Normalmente, te referirías a una tupla como lo harías a una lista, utilizando un único nombre de variable:

measurements = 6, 30

Sin embargo, supongamos que queremos almacenar los valores en dos variables en lugar de en una. La sintaxis para hacerlo no es compleja. Separas los nombres de las variables con comas, utilizas un signo igual y escribes las tuplas al otro lado del signo igual. Cada valor de tupla se asignará a la variable en la posición correspondiente. Echemos un vistazo.

En este ejemplo, dos variables, anchura y altura, se establecen en los valores 6 y 30, respectivamente:

width, height = 6, 30

Ahora tenemos dos variables. Una se llama anchura y tiene el valor 6, y la otra se llama altura y tiene el valor 30. ¡Y lo hemos hecho utilizando una sola línea de código!

#### **Misión nº 49: Deslizamiento**

Establecer variables con tuplas es una forma rápida y sencilla de ahorrar espacio en tus programas. También es útil para establecer variables relacionadas entre sí en un mismo lugar de tu programa. Por ejemplo, a lo largo del libro has utilizado código como éste para establecer los valores de las variables x, y y z:

x = 10  
y = 11  
z = 12

En cambio, puedes utilizar una tupla para establecer todos estos valores en una sola línea:

x, y, z = 10, 11, 12

A continuación, pondrás en práctica tus nuevas habilidades para escribir código. Tu misión es crear un programa que mueva al jugador aleatoriamente por el mundo del juego en pequeños pasos, haciendo que parezca que patinas sobre hielo. He empezado el programa por ti en el Listado [9-3](ch09.xhtml#ch9ex3); faltan algunas partes y tienes que completarlas.

*sliding.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
import random  
import time  
  
➊ # Get the player's position  
  
➋ # Set the x, y, and z variables on the same line using a tuple  
  
while True:  
➌ x += random.uniform(-0.2, 0.2)  
# Change the z variable by a random float  
➍ z +=  
y = mc.getHeight(x, z)  
  
mc.player.setPos(x, y, z)  
time.sleep(0.1)

*Listado 9-3: Inicio del código para hacer que el jugador se deslice por el mapa*

Copia [el Listado 9-3](ch09.xhtml#ch9ex3) en un nuevo archivo y guárdalo como *deslizamiento.py* en tu carpeta *de listas*. Para terminar el programa, necesitas obtener la posición inicial del jugador ➊ y establecer los valores de las variables x, y, z ➋. Utiliza una tupla para fijar estos valores. Este programa también utiliza la función uniforme () ➌, que es como la función randint() (ver "[Jugar con números aleatorios](ch03.xhtml#ch03lev2sec12)" en [la página 66](ch03.xhtml#page_66)) pero devuelve un valor flotante aleatorio en lugar de un valor entero. Utiliza la función uniform() para cambiar el valor de la variable z en el bucle ➍. Esto ya se ha hecho para la variable x ➌.

[La Figura 9-3](ch09.xhtml#ch9fig3) muestra a mi jugador deslizándose lentamente por mi juego.



*Figura 9-3: Deslizándose lentamente hacia atrás por mi jardín*

**OBJETIVO EXTRA: DESLIZAR BLOQUES**

El programa *deslizar.* py hace que el jugador se deslice aleatoriamente por el juego. ¿Puedes averiguar cómo cambiar el programa para que haga deslizar un bloque?

#### **Devolver una tupla**

Algunas de las funciones incorporadas en Python devuelven una tupla. Cuando defines tus propias funciones, también pueden devolver el resultado como una tupla. Para ello, coloca una tupla después de la palabra clave return. Por ejemplo, creemos una función para convertir una fecha en una tupla. Damos la fecha como argumento de cadena, y la función devolverá el año, el mes y el día en una tupla. Éste es el código:

def getDateTuple(dateString):  
year = int(dateString[0:4])  
month = int(dateString[5:7])  
day = int(dateString[8:10])  
return year, month, day

Cuando llamamos a la función y le damos una fecha como cadena, nos devuelve una tupla que contiene el año, el mes y el día en ese orden:

>>> getDateTuple("1997-09-27")  
(1997, 9, 27)

Cuando llamamos a la función, podemos almacenar la tupla devuelta como queramos. Este código almacena cada valor en una variable independiente:

year, month, day = getDateTuple("1997-09-27")

Ahora podemos convertir rápidamente cadenas de fechas en variables individuales. En mi trabajo como desarrollador de software, utilizo código muy similar a éste todo el tiempo.

[anterior](ch09_4.html)[Subtema 5 de 8: (Ver todo)](ch09.html)[siguiente](ch09_6.html)