Aprende a programar con Minecraft

### **Bucles for-else**

También puedes utilizar la sentencia else con un bucle for. Cuando utilizas else con un bucle for, se ejecuta después de que el bucle for llegue al final de la lista. Si el bucle for no llega al final de su lista, la sentencia else no se ejecutará.

Por ejemplo, aquí tienes un trozo de código que imprime los ingredientes de un bocadillo y luego utiliza una sentencia else:

sandwich = ["Bread", "Butter", "Tuna", "Lettuce", "Mayonnaise", "Bread"]  
  
for ingredient in sandwich:  
print(ingredient)  
else:  
print("This is the end of the sandwich.")

Cuando ejecutas este código, imprime lo siguiente:

Bread  
Butter  
Tuna  
Lettuce  
Mayonnaise  
Bread  
This is the end of the sandwich.

Podrías pensar que este código es lo mismo que escribir lo siguiente:

for ingredient in sandwich:  
print(ingredient)  
print("This is the end of the sandwich.")

Pues sí. Ambos trozos de código harán lo mismo. Entonces, ¿qué sentido tiene utilizar else con un bucle for? Bueno, cuando se utiliza con una sentencia break, la sentencia else se comportará de forma diferente. Veámoslo a continuación.

#### **Romper un bucle for-else**

Utilizar una sentencia break para salir de un bucle for es una forma de evitar que se ejecute la sentencia else.

El siguiente ejemplo incorpora una sentencia break dentro de una sentencia if. El bucle se romperá si el elemento actual es "Mayonesa":

sandwich = ["Bread", "Butter", "Tuna", "Lettuce", "Mayonnaise", "Bread"]  
  
for ingredient in sandwich:  
if ingredient == "Mayonnaise":  
print("I don't like mayonnaise on my sandwich.")  
break  
else:  
print(ingredient)  
else:  
print("This is the end of the sandwich.")

¿Puedes predecir cuál será la salida? Piénsalo antes de ejecutar este código, y luego ejecútalo a ver qué pasa.

#### **Misión nº 60: El buscador de diamantes**

A veces, cuando juego a Minecraft con amigos, no me dejan utilizar programas Python para generar bloques de diamantes. Pero sigo necesitando diamantes para armaduras, herramientas y para construir castillos de diamantes. Excavar en busca de diamantes es bastante fácil, pero no siempre los encuentras.

Para ahorrar algo de tiempo, escribí un programa que comprueba si hay algún mineral de diamante directamente debajo de mí. El programa obtiene mi posición actual y luego utiliza un bucle for para comprobar uno a uno los bloques que hay debajo de mí para ver si son mineral de diamante. Si se encuentra mineral de diamante, el programa me dice a qué profundidad está el mineral; si no se encuentra mineral de diamante, el programa publica un mensaje para decirme que no hay mineral de diamante debajo de mí.

Crea un nuevo programa y guárdalo como *diamondSurvey.py* en la carpeta *forLoops*.

Utiliza un bucle for para cambiar el valor de la variable y en -1 cada vez que el bucle itere. En total, el bucle debe repetirse 50 veces para comprobar 50 bloques de profundidad. En cada iteración, utiliza una sentencia if para comprobar si el bloque en esa posición es mineral de diamante (bloque ID 56). Si es un bloque de mineral de diamante, envía un mensaje al chat para decir a qué distancia está el bloque por debajo del jugador y rompe el bucle. Si no se encuentran bloques de mineral de diamante, utiliza una sentencia else en tu bucle for para enviar un mensaje que diga que no hay bloques de mineral de diamante directamente debajo del jugador.

La[Figura 10-6](ch10.xhtml#ch10fig6) muestra el programa en funcionamiento.



*Figura 10-6: Parece que hay un bloque de mineral de diamante cuatro bloques por debajo de mí. ¡Es hora de empezar a cavar!*

**OBJETIVO EXTRA: ORO EN LAS COLINAS**

Cambia el programa *diamondSurvey.* py para que busque también otros bloques de mineral, como mineral de hierro o mineral de oro.

[anterior](ch10_4.html)[Subtema 5 de 7: (Ver todo)](ch10.html)[siguiente](ch10_6.html)