Aprende a programar con Minecraft

### **Terminar un bucle while con break**

Con los bucles while, tienes control total sobre cómo y cuándo termina el bucle. Hasta ahora sólo has utilizado condiciones para terminar bucles, pero también puedes utilizar una sentencia break. La sentencia break permite que tu código salga inmediatamente de un bucle while. Veamos este concepto.

Una forma de utilizar las sentencias break es ponerlas en una sentencia if anidada en el bucle. Al hacerlo, el bucle se detiene inmediatamente cuando la condición de la sentencia if es Verdadera. El siguiente código pide continuamente la entrada del usuario hasta que escriba "exit":

➊ while True:  
➋ userInput = input("Enter a command: ")  
➌ if userInput == "exit":  
➍ break  
print(userInput)  
➎ print("Loop exited")

Se trata de un bucle infinito porque utiliza while True ➊. Cada vez que se repite el bucle, pide al usuario que introduzca un comando ➋. El programa comprueba si la entrada es " salida" ➌ utilizando una sentencia if. Si la entrada cumple la condición, la sentencia break detiene la repetición del bucle ➍, y el programa continúa en la línea inmediatamente posterior al cuerpo del bucle, imprimiendo " Bucle salido " en la shell de Python ➎.

#### **Misión nº 39: Crear una conversación persistente con un bucle**

En la Misión nº 13[(página 76](ch04.xhtml#page_76)), creaste un programa que publica el mensaje del usuario en el chat utilizando cadenas, entrada y salida. Aunque este programa era útil, estaba bastante limitado porque tenías que volver a ejecutar el programa cada vez que querías publicar un nuevo mensaje.

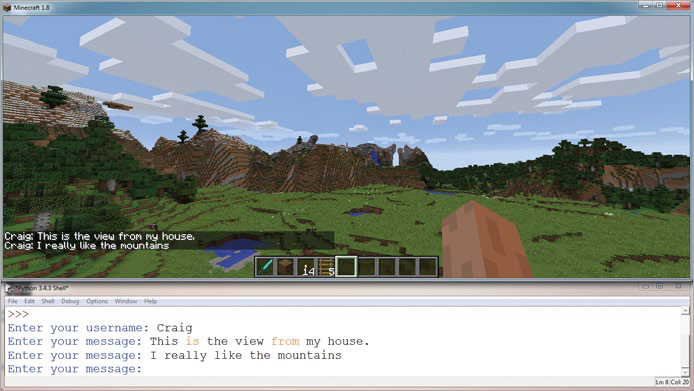
En esta misión, mejorarás tu programa de chat utilizando un bucle while para que los usuarios puedan publicar tantos mensajes como quieran sin reiniciar el programa.

Abre el archivo *userChat.py* en la carpeta *strings* y luego guárdalo como *chatLoop.py* en la carpeta *whileLoops*.

Para publicar un nuevo mensaje cada vez que quieras sin reiniciar el programa, añade lo siguiente a tu código:

1. Añade un bucle while infinito al programa.
2. Añade una sentencia if al bucle para comprobar si la entrada del usuario es "exit". Si la entrada es "exit ", el bucle debería romperse.
3. Asegúrate de que la variable userName está definida antes del inicio del bucle.

Cuando hayas añadido los cambios, guarda tu programa y ejecútalo. Un prompt en el shell de Python te pedirá que escribas un nombre de usuario. Hazlo y pulsa ENTER. A continuación, el programa te pedirá que introduzcas un mensaje. Escribe un mensaje y pulsa INTRO. El programa seguirá pidiéndote que introduzcas un mensaje hasta que escribas exit. [La Figura 7-8](ch07.xhtml#ch7fig8) muestra mi programa de chat en funcionamiento.



*Figura 7-8: Estoy chateando conmigo mismo.*

**OBJETIVO EXTRA: BLOQUEAR EL CHAT**

Amplía la función de chat para que los usuarios puedan crear bloques. Por ejemplo, si el usuario introduce "lana", el programa crea un bloque de lana. Puedes hacerlo añadiendo sentencias elif a tu sentencia if para comprobar la entrada del usuario.

#### **Sentencias while-else**

Al igual que las sentencias if, los bucles while pueden tener condiciones secundarias activadas por sentencias else.

La sentencia else se ejecuta cuando la condición de una sentencia while es False. A diferencia del cuerpo de una sentencia while, la sentencia else se ejecutará sólo una vez, como se muestra aquí:

message = input("Please enter a message.")  
  
while message != "exit":  
print(message)  
message = input("Please enter a message.")  
else:  
print("User has left the chat.")

Este bucle se repite mientras el mensaje introducido no sea igual a "exit". Si el mensaje es "exit ", el bucle dejará de repetirse, y el cuerpo de la sentencia else imprimirá "El usuario ha abandonado el chat".

Si utilizas una sentencia break en la sentencia while, la sentencia else no se ejecutará. El siguiente código es similar al ejemplo anterior, pero incluye una sentencia if anidada y una sentencia break. Cuando el usuario escriba abort en lugar de exit, el bucle de chat saldrá sin imprimir el mensaje "El usuario ha abandonado el chat. " en el chat.

message = input("Please enter a message.")  
  
while message != "exit":  
print(message)  
message = input("Please enter a message.")  
if message == "abort":  
break  
else:  
print("User has left the chat.")

La sentencia if comprueba si el mensaje introducido es "abortar ". Si es True, se ejecuta la sentencia break y el bucle saldrá. Como se ha utilizado la sentencia break, el cuerpo de la sentencia else no se ejecutará y no se imprimirá "El usuario ha abandonado el chat".

#### **Misión 40: Frío y caliente**

En esta misión, crearemos una partida de Frío y Caliente en Minecraft. Si nunca has jugado, la idea es que tu amigo esconde un objeto y tú tienes que encontrarlo. Tu amigo te da pistas en función de lo lejos que estés del objeto. Si estás cerca, tu amigo dirá "Caliente", y si estás lejos, dirá "Frío". Cuando estés justo al lado del objeto, dirá "¡Estás ardiendo!" y si estás muy lejos, dirá "¡Congelación!".

El objetivo del juego es encontrar y colocarte sobre el bloque de diamante que se ha colocado aleatoriamente en el mundo del juego. En esta versión del juego, jugarás tú solo, y el programa Python te dirá a qué distancia del bloque oculto te encuentras. El juego termina cuando te sitúas sobre el bloque diamante.

El listado[7-2](ch07.xhtml#ch7ex2) coloca un bloque en un lugar aleatorio.

*blockHunter.py*

from mcpi.minecraft import Minecraft  
import math  
import time  
import random  
mc = Minecraft.create()  
  
destX = random.randint(-127, 127)  
destZ = random.randint(-127, 127)  
➊ destY = mc.getHeight(destX, destZ)  
  
block = 57  
➋ mc.setBlock(destX, destY, destZ, block)  
mc.postToChat("Block set")  
  
while True:  
pos = mc.player.getPos()  
➌ distance = math.sqrt((pos.x - destX) \*\* 2 + (pos.z - destZ) \*\* 2)  
  
➍ if distance > 100:  
mc.postToChat("Freezing")  
elif distance > 50:  
mc.postToChat("Cold")  
elif distance > 25:  
mc.postToChat("Warm")  
elif distance > 12:  
mc.postToChat("Boiling")  
elif distance > 6:  
mc.postToChat("On fire!")  
elif distance == 0:  
➎ mc.postToChat("Found it")

*Listado 7-2: Inicio del programa Frío y Caliente*

Antes de colocar un bloque al azar, el programa se asegura de que el bloque no se colocará bajo tierra. Para ello, utiliza la función getHeight() ➊, que encuentra el bloque que tiene la coordenada y más alta (es decir, en la superficie) para cualquier posición del juego. Luego coloca un bloque de diamante en una posición aleatoria ➋.

El código en ➌ calcula la distancia al bloque de diamante. Utiliza la función sqrt (), que está en el módulo matemático; por eso es necesario importar matemática al principio del programa. La función sqrt () calcula la raíz cuadrada de un número.

**NOTA**

[*El listado 7-2*](ch07.xhtml#ch7ex2) *utiliza una fórmula llamada* teorema de Pitágoras*. La fórmula utiliza dos lados de un triángulo para calcular la longitud del tercero. En este caso, utilizo la distancia del jugador al bloque oculto en el eje x y en el eje z para calcular la distancia al bloque oculto en línea recta.*

El mensaje que muestra el programa depende de la distancia a la que te encuentres del bloque, que puedes averiguar utilizando una sentencia if y la variable de distancia ➍. El programa muestra "Congelación" si estás muy lejos y " ¡En llamas!" si estás muy cerca.

Copia [el Listado 7-2](ch07.xhtml#ch7ex2) en un nuevo archivo en IDLE y guarda el programa como *blockHunter.py* en la carpeta *whileLoops*.

De momento el programa funciona, pero no termina cuando encuentras el bloque. Para terminar el código, tienes que añadir una sentencia break cuando la distancia del jugador al bloque sea 0 ➎.

Cuando hayas terminado el programa, guárdalo y ejecútalo. Se generará un bloque aleatorio y tendrás que encontrarlo. El programa debería detenerse cuando encuentres el bloque y te pares sobre él. [La Figura 7-9](ch07.xhtml#ch7fig9) muestra que acabo de encontrar el bloque.



*Figura 7-9: He encontrado el bloque y ahora sólo tengo que ponerme encima.*

**OBJETIVO EXTRA: TIEMPO AL TIEMPO**

El programa *blockHunter.* py te da todo el tiempo que necesites para encontrar el bloque. ¿Se te ocurre alguna forma de mostrar el tiempo que tarda el jugador en encontrar el bloque o incluso de limitar el tiempo que tiene para jugar?

[anterior](ch07_4.html)[Subtema 5 de 6: (Ver todo)](ch07.html)[siguiente](ch07_6.html)