Aprende a programar con Minecraft

### **Condiciones extravagantes**

Como los bucles while esperan un valor booleano para su condición, puedes utilizar cualquiera de los comparadores y operadores booleanos que has aprendido hasta ahora. Por ejemplo, ya has visto que los operadores mayor que y menor que funcionan igual que en capítulos anteriores.

Pero también puedes controlar los bucles while con comparadores y operadores booleanos de otras formas. ¡Echemos un vistazo!

Empezaremos escribiendo una condición más interactiva. El siguiente código crea la variable continueAnswer antes de que se inicie el bucle y comprueba que el valor es igual a "Y". Ten en cuenta que no podemos utilizar la palabra continuar como nombre de variable porque es una palabra reservada en Python.

continueAnswer = "Y"  
coins = 0  
while continueAnswer == "Y":  
coins = coins + 1  
continueAnswer = input("Continue? Y/N")  
print("You have " + str(coins) + " coins")

En la última línea del bucle while, el programa pide una entrada al usuario. Si el usuario pulsa algo más que "Y " como respuesta, el bucle saldrá. El usuario puede pulsar repetidamente Y y Y y Y, y cada vez el valor de la variable monedas aumentará en 1.

Observa que la variable que se comprueba, continuarRespuesta, se crea antes de que se inicie el bucle. Si no fuera así, el programa mostraría un error. Por eso, la variable que utilizamos para comprobar la condición debe existir antes de que intentemos utilizarla, y debe ser Verdadera cuando el programa llegue al bucle while la primera vez; de lo contrario, la condición no se cumplirá, y la sentencia del cuerpo del bucle while nunca se ejecutará.

#### **Misión 36: Concurso de inmersión**

Vamos a divertirnos un poco con los bucles while y el comparador igual a(==). En esta misión, crearás un minijuego en el que el jugador se sumerja bajo el agua todo el tiempo que pueda. El programa registrará cuántos segundos permanece bajo el agua y mostrará su puntuación al final del programa. Para felicitar al jugador, el programa le regalará flores si permanece bajo el agua más de 6 segundos.

Aquí tienes algo de código para empezar:

*divingContest.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
import time  
  
score = 0  
pos = mc.player.getPos()  
➊ blockAbove = mc.getBlock(pos.x, pos.y + 2, pos.z)  
  
➋ # Add a while loop here  
time.sleep(1)  
pos = mc.player.getPos()  
➌ blockAbove = mc.getBlock(pos.x, pos.y + 2, pos.z)  
➍ score = score + 1  
mc.postToChat("Current score: " + str(score))  
  
mc.postToChat("Final score: " + str(score))  
  
➎ if score > 6:  
finalPos = mc.player.getTilePos()  
mc.setBlocks(finalPos.x - 5, finalPos.y + 10, finalPos.z - 5,  
finalPos.x + 5, finalPos.y + 10, finalPos.z + 5, 38)

Guarda el programa como *divingContest.py* en tu carpeta *whileLoops*. La variable puntuación lleva la cuenta de cuántos segundos permanece el jugador bajo el agua.

Ejecuta el código para ver qué ocurre. De momento, el programa no está completo: sólo comprueba una vez si el jugador está bajo el agua y luego termina.

Antes de arreglar esto, veamos qué hace el resto del código. La variable bloqueEncima almacena el tipo de bloque situado en la cabeza del jugador ➊. Por ejemplo, si la cabeza del jugador está bajo el agua, esta variable almacenará un valor de 8 (lo que significa que el bloque es agua). Más adelante en el código, volverás a establecer blockAbove para que almacene el valor del bloque situado encima de la cabeza del jugador ➌, de modo que cuando crees tu bucle while, actualizará blockAbove con el bloque actual situado encima de la cabeza del jugador. En ➍, el programa añade 1 punto al total por cada segundo que el jugador esté bajo el agua, y en ➎, utiliza una sentencia if para crear una lluvia de flores sobre el jugador si la puntuación es superior a 6.

Te toca añadir un bucle al programa que utilice la variable bloqueSobre como condición en ➋. Haz que el bucle while compruebe si bloqueSobre es igual a agua (bloque de tipo 8) o igual a agua corriente (bloque de tipo 9). Puedes utilizar la siguiente condición en el bucle while para comprobarlo: while bloqueEncima == 8 o bloqueEncima == 9. Esto comprueba si el jugador está actualmente bajo el agua y seguirá comprobando si el jugador está bajo el agua cada vez que se repita el bucle.

Para probar tu programa, busca un poco de agua que tenga al menos tres bloques de profundidad y sumérgete en ella. El programa sólo se ejecutará si ya estás bajo el agua. Cuando ejecutes el programa, debería empezar a mostrar cuántos segundos has estado bajo el agua. Al cabo de un rato, nada hasta la superficie. El programa mostrará tu puntuación y te colmará de flores si has estado bajo el agua 6 segundos o más. La [Figura 7-4](ch07.xhtml#ch7fig4) muestra al jugador bajo el agua y la puntuación que se muestra. La Figura [7-5](ch07.xhtml#ch7fig5) muestra las flores que aparecen cuando ganas.



*Figura 7-4: Estoy aguantando la respiración bajo el agua, y se muestra el número de segundos que he estado bajo el agua.*



*Figura 7-5: ¡He ganado mi propia celebración florida!*

**OBJETIVO EXTRA: UN GANADOR ERES TÚ**

Prueba a añadir premios extra escribiendo más código en la sentencia if al final del programa. Si el jugador obtiene una puntuación alta, podrías darle un bloque de oro. Prueba a añadir varios niveles de dificultad con premios diferentes para cada uno.

#### **Operadores booleanos y bucles while**

Puedes utilizar operadores booleanos como *y*, *o*, y *no* con un bucle while cuando quieras que el bucle utilice más de una condición. Por ejemplo, el siguiente bucle iterará mientras el usuario no haya introducido la contraseña correcta y haya hecho tres intentos o menos:

password = "cats"  
passwordInput = input("Please enter the password: ")  
attempts = 0  
➊ while password != passwordInput and attempts < 3:  
➋ attempts += 1  
➌ passwordInput = input("Incorrect. Please enter the password: ")  
➍ if password == passwordInput:  
print("Password accepted.")

La condición del bucle while ➊ realiza dos tareas: comprueba si la contraseña es diferente de la introducida por el usuario(contraseña != contraseñaEntrada) y comprueba si el usuario ha intentado introducir la contraseña tres veces o menos(intentos < 3). El operador and permite que el bucle while compruebe ambas condiciones al mismo tiempo. Si la condición es False, el bucle incrementa la variable intent os ➋ y pide al usuario que vuelva a introducir la contraseña ➌. El bucle finalizará si el usuario introduce la contraseña correcta o la variable intentos es mayor que 3. Una vez finalizado el bucle, el programa mostrará Contraseña aceptada sólo si el usuario ha introducido la contraseña correcta ➍.

#### **Comprobar un rango de valores en bucles while**

También puedes comprobar valores en un rango determinado utilizando un bucle while. Por ejemplo, el siguiente código comprueba si el valor que ha introducido el usuario está entre 0 y 10. Si no lo está, el bucle saldrá.

position = 0  
➊ while 0 <= position <= 10:  
position = int(input("Enter your position 0-10: "))  
print(position)

Si la variable de posición es mayor que 10, el bucle no se repetirá ➊. Lo mismo ocurrirá si el valor es menor que 0. Esto es útil en Minecraft cuando compruebas si la posición del jugador se encuentra en una determinada zona del juego, como verás en la siguiente misión.

#### **Misión 37: Hacer una pista de baile**

¡Es hora de bailar! Pero antes de que puedas hacer unos buenos movimientos, necesitarás una pista de baile. El programa de esta misión generará una pista de baile que parpadeará con colores diferentes cada medio segundo mientras el jugador permanezca en la pista.

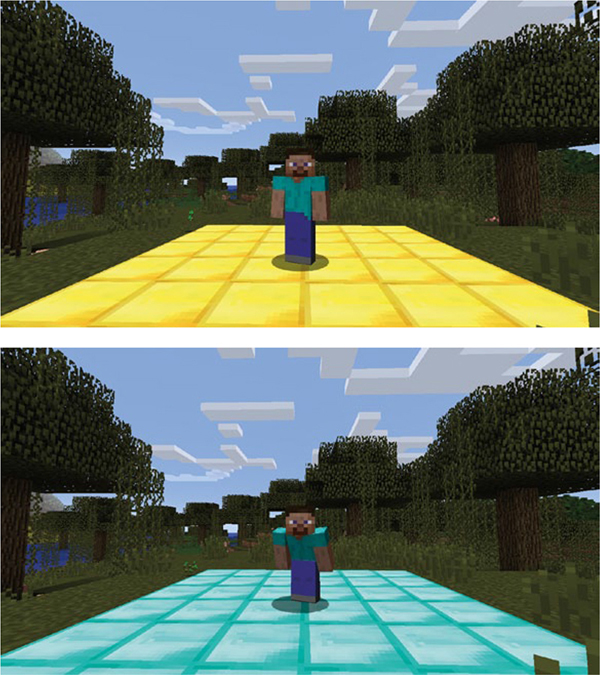
A continuación se muestra el inicio del código. Crea una pista de baile en la posición actual del jugador y utiliza una sentencia if para cambiar los colores. Pero el código no está completo.

*danceFloor.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
import time  
  
pos = mc.player.getTilePos()  
floorX = pos.x – 2  
floorY = pos.y - 1  
floorZ = pos.z – 2  
width = 5  
length = 5  
block = 41  
➊ mc.setBlocks(floorX, floorY, floorZ,  
floorX + width, floorY, floorZ + length, block)  
  
➋ while floorX <= pos.x <= floorX + width and # Check z is within the floor  
➌ if block == 41:  
block = 57  
else:  
block = 41  
  
mc.setBlocks(floorX, floorY, floorZ,  
floorX + width, floorY, floorZ + length, block)  
pos = mc.player.getTilePos()  
time.sleep(0.5)

Abre IDLE, crea un nuevo archivo y guarda el programa como pista de *baile.py* en la carpeta *whileLoops*. El código construye la pista de baile basándose en la posición actual del jugador ➊ y almacena la ubicación y el tamaño de la pista de baile en las variables floorX, floorY, floorZ, width y length. Dentro del bucle while, el código utiliza una sentencia if para alternar los bloques de los que está hecha la pista de baile ➌, haciendo que la pista de baile parezca que parpadea.

Para que el programa funcione correctamente, tienes que cambiar la condición del bucle while para comprobar si la coordenada z del jugador está en la pista de baile ➋. En otras palabras, comprueba si pos.z es mayor o igual que floorZ y menor o igual que floorZ más la longitud. Como orientación, mira cómo he comprobado si pos. x está en la pista de baile utilizando(sueloX <= pos.x <= sueloX + anchura). ¡La [Figura 7-6](ch07.xhtml#ch7fig6) muestra la pista de baile en acción!



*Figura 7-6: Mostrando mis movimientos en la pista de baile.*

Cuando hayas terminado el programa, guárdalo y ejecútalo. Debería aparecer una pista de baile debajo del reproductor y cambiar cada medio segundo. Baila un poco en , ¡diviértete! Cuando hayas terminado, sal de la pista de baile y asegúrate de que deja de parpadear. No volverá a encenderse a menos que vuelvas a ejecutar el programa para crear una nueva pista de baile.

**OBJETIVO EXTRA: SE ACABÓ LA FIESTA**

Cuando el jugador haya terminado de bailar en la pista de baile, haz que ésta desaparezca. Para ello, cambia la pista de baile a aire cuando termine el bucle.

#### **Sentencias if anidadas y bucles while**

Puedes escribir programas más potentes utilizando sentencias if y sentencias if anidadas dentro de bucles while. Puede que hayas visto una sentencia if anidada en el código de la Misión 37[(página 139](ch07.xhtml#page_139)).

En el siguiente ejemplo, la sentencia if anidada comprueba la última palabra que se imprimió y decide si se imprimen las palabras "mine" y "craft". El bucle se repite 50 veces.

word = "mine"  
count = 0  
while count < 50:  
print(word)  
if word == "mine":  
word = "craft"  
else:  
word = "mine"

La variable palabra almacena la primera palabra que se imprimirá. La sentencia if del bucle comprueba si la palabra actual es "mina " y, si lo es, cambia la palabra a "artesanía " y la imprime en la siguiente iteración del bucle. Si la palabra no es "mía ", se cambiará a "artesanía ". Se trata de un bucle infinito, ¡así que asegúrate de utilizar CTRL-C para escapar!

También puedes anidar sentencias elif y otros bucles while dentro de bucles while.

El siguiente programa pregunta al usuario si desea imprimir todos los números comprendidos entre uno y un millón:

userAnswer = input("Print the numbers between 1 and 1000000? (yes/no): ")  
  
➊ if userAnswer = "yes":  
count = 1  
➋ while count <= 1000000:  
print(count)  
count += 1

La sentencia if comprueba si la entrada del usuario es sí ➊. Si lo es, el programa ejecuta el bucle anidado en la sentencia if ➋. Si la entrada es cualquier otra, el programa no ejecutará el bucle y terminará.

#### **Misión 38: El toque de Midas**

Midas es un rey de leyenda. Todo lo que tocaba se convertía en oro. Tu misión es escribir un programa que convierta en oro todos los bloques situados por debajo del jugador, excepto el aire y el agua, claro, ¡o estarías en un buen lío! Recuerda que el bloque dorado tiene un valor de 41, el agua es 9 y el aire es 0.

*midas.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
air = 0  
water = 9  
  
➊ # Add an infinite while loop here  
pos = mc.player.getTilePos()  
blockBelow = mc.getBlock(pos.x, pos.y - 1, pos.z)  
  
➋ # Add if statement here  
mc.setBlock(pos.x, pos.y - 1, pos.z, 41)

Abre IDLE y crea un nuevo archivo. Guarda el archivo como *midas.py* en la carpeta *whileLoops*. Tienes que añadir algo más al programa para que pueda hacer lo que necesitas. En primer lugar, añadirás un bucle while infinito ➊. Recuerda que un bucle while infinito tiene una condición que siempre es Verdadera. También tienes que añadir una sentencia if que compruebe si el bloque situado debajo del jugador no es igual a aire y no es igual a agua estancada ➋. El valor del bloque situado debajo del jugador se almacena en la variable bloqueDebajo, y los valores del aire y del agua se almacenan en las variables aire y agua.

Cuando hayas completado el programa, guárdalo y ejecútalo. El jugador debería dejar un rastro de oro tras de sí. Cuando salte al agua o vuele por el aire, los bloques que hay debajo no deberían cambiar. [La Figura 7-7](ch07.xhtml#ch7fig7) muestra el programa en acción.



*Figura 7-7: Cada bloque que piso se convierte en oro.*

Para salir del bucle infinito, ve a **Shell** image **Reiniciar Shell** en tu shell IDLE o haz clic en el shell y pulsa CTRL-C.

**OBJETIVO EXTRA: SOY UN LABRADOR**

Puedes cambiar *midas.* py para que sirva a diversos propósitos. ¿Qué tal si lo cambias para que cambie automáticamente los bloques de tierra por bloques de hierba? ¿Qué tal cambiar los bloques de tierra por bloques de hierba?

[anterior](ch07_3.html)[Subtema 4 de 6: (Ver todo)](ch07.html)[siguiente](ch07_5.html)