Aprende a programar con Minecraft

### **Comparadores**

Se te da muy bien comparar valores. Sabes que 5 es mayor que 2, que 8 y 8 son el mismo número y que 6 y 12 no son el mismo número. Un ordenador también es bueno comparando valores; sólo tienes que decirle exactamente qué tipo de comparación quieres escribiendo un símbolo llamado *comparador*. Por ejemplo, ¿quieres que compruebe si un valor es mayor que el otro, o quieres que compruebe si es menor?

Los comparadores (u *operadores de comparación*) de Python te permiten comparar datos. Python utiliza seis comparadores:

- Igual a(==)

- No igual a (!=)

- Menor que (<)

- Menor o igual que (<=)

- Mayor que(>)

- Mayor o igual que(>=)

Cada comparador devuelve un valor booleano(Verdadero o Falso) que indica si se ha cumplido la condición. Veamos estos comparadores y exploremos cómo utilizarlos.

#### **Igual a**

Cuando quieras averiguar si dos valores son iguales, puedes utilizar el comparador igual a(==). Cuando los valores son iguales, la comparación devuelve el valor booleano Verdadero. Cuando los valores son diferentes, la comparación devuelve Falso.

Por ejemplo, asignemos el valor de dos variables y utilicemos el operador igual a para compararlas:

>>> length = 2  
>>> width = 2  
>>> length == width  
True

El resultado es Verdadero porque los valores de las variables longitud y anchura son iguales.

Pero si son diferentes, el resultado es Falso:

>>> length = 4  
>>> width = 1  
>>> length == width  
False

Puedes utilizar el comparador igual a en todos los tipos de variables: cadenas, enteros, flotantes y booleanos.

Fíjate en que he utilizado == para comparar la longitud y la anchura en lugar de =, que se utiliza para establecer una variable. Python utiliza el operador == para diferenciar entre una comparación (preguntar si dos valores son iguales) y establecer una variable (hacer que una variable sea igual a algún valor). Intenta recordar esta diferencia para evitar errores en tu código. No te preocupes; ¡incluso yo cometo el error de utilizar = en lugar de == de vez en cuando!

#### **Misión nº 18: ¿Estoy nadando?**

Ahora utilizarás comparadores para crear un programa que indique si estás o no en el agua. Los resultados se publicarán en el chat de Minecraft.

Para averiguar el tipo de bloque en determinadas coordenadas, utilizarás la función getBlock(). Esta función toma las coordenadas como tres argumentos y devuelve el tipo de bloque como un número entero. Por ejemplo:

blockType = mc.getBlock(10, 18, 13)

Aquí he almacenado el resultado de mc.getBlock(10, 18, 13) en una variable llamada tipoBloque. Si el tipo de bloque en las coordenadas (10, 18, 13) es melón (valor de bloque 103), la variable blockType tendrá el valor 103.

Pongamos a trabajar la función getBlock(). El listado [5-2](ch05.xhtml#ch5ex2) comprueba si el jugador está en tierra firme.

*swimming.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
pos = mc.player.getPos()  
x = pos.x  
y = pos.y  
z = pos.z  
  
blockType = mc.getBlock(x, y, z)  
mc.postToChat(blockType == 0)

*Listado 5-2: Este código comprueba el tipo de bloque donde están las piernas del jugador.*

Aquí, obtengo las tres coordenadas de la posición del jugador y paso esas coordenadas como argumentos a getBlock (). Almaceno el resultado de mc.getBlock(x, y, z) en blockType. La expresión blockType == 0 comprueba si el bloque es aire; si es aire, sabes que estás parado en algún lugar de tu mundo Minecraft, la expresión es True, y True se publica en el chat. Si no es aire, se envía False al chat, ¡así que debes estar bajo el agua o quizás ahogándote en la arena!

Copia [el Listado 5-2](ch05.xhtml#ch5ex2) y guárdalo como *nadando.py* en el directorio [*del Capítulo 5*](ch05.xhtml#ch05). Luego cambia el código para que compruebe si el jugador está de pie en el agua (bloque tipo 9) y ejecútalo.

Prueba a ponerte de pie en el agua y a ejecutar el programa. Asegúrate de que cuando el jugador está en el agua, el chat muestra Verdadero. Cuando el jugador no esté en el agua, el chat debe mostrar Falso.

La salida del programa debe parecerse a [la Figura 5-2](ch05.xhtml#ch5fig2).

**NOTA**

*En este punto, no podrás ejecutar este programa de forma continua. Deberás ejecutar el programa cada vez que quieras comprobar el bloque situado debajo del jugador. Esto se aplica también a todas las demás misiones de este capítulo.*



*Figura 5-2: Aunque veo que estoy en el agua, Python me lo confirma amablemente.*

**OBJETIVO EXTRA: ¡ESTOY VOLANDO!**

Con un par de cambios en el código, puedes comprobar si el bloque que *hay debajo de* ti es aire. Esto te indicaría que estás volando o saltando. ¿Cómo lo harías?

#### **No igual a**

El comparador no igual a es lo contrario del comparador igual a. En lugar de comprobar si dos valores son iguales, comprueba si son diferentes. Cuando los dos valores son diferentes, la comparación se evalúa como Verdadero. Si son iguales, se evalúa como Falso.

Supongamos que quieres asegurarte de que un objeto es un rectángulo pero no un cuadrado. Como un rectángulo no cuadrado tiene una longitud y una anchura diferentes, podrías escribir una comparación para comprobar que la longitud y la anchura no son iguales:

>>> width = 3  
>>> length = 2  
>>> width != length  
True

La expresión anchura != longitud pregunta si los valores de anchura y longitud son diferentes.

El resultado de esta comparación es Verdadero porque la variable anchura y la variable longitud tienen valores diferentes.

Pero si estos valores son iguales, la comparación devuelve Falso:

>>> width = 3  
>>> length = 3  
>>> width != length  
False

El comparador no igual a también funciona con cadenas, enteros, flotantes y booleanos, igual que el comparador igual a.

#### **Misión 19: ¿Estoy parado en algo más que aire?**

Supongamos que quieres comprobar si estás sobre algo distinto del aire, como agua, lava, tierra, grava o cualquier otro tipo de bloque. En la Misión nº 18, comprobaste si el bloque en tu posición actual era aire, y resolviste cómo comprobar si estabas en el agua. Podrías copiar y pegar el programa muchas veces, cambiándolo ligeramente cada vez para comprobar si hay lava, tierra, grava, etc., uno por uno. Pero eso sería muy aburrido. En lugar de eso, utiliza el comparador no igual a para comprobar si estás bajo tierra, atrapado en arena, en el fondo del océano, ¡o incluso ahogándote en lava!

Abre el programa de la Misión 18*(nadando.py*) y guárdalo como *notAir.py* en la carpeta de *booleanos*. Elimina la última línea del programa y sustitúyela por el Listado [5-3](ch05.xhtml#ch5ex3).

*notAir.py*

➊ notAir = blockType == 0  
mc.postToChat("The player is not standing in air: " + str(notAir))

*Listado 5-3: Cambios en el programa de natación*

La última línea de este código imprimirá si no estás en el aire. El resultado de la comparación se almacena en la variable noAire ➊. Cuando la comparación se evalúe como Verdadero, el valor de la variable noAire será Verdadero, y cuando la comparación se evalúe como Falso, el valor de la variable noAire será Falso.

Pero la comparación de la primera línea no es del todo correcta ➊. Actualmente comprueba si el bloque Tipo es igual a aire utilizando el comparador igual a(==). En lugar de eso, debería comprobar si la variable bloqueTipo no es igual a aire utilizando el comparador no igual a (!=). Cambia la primera línea para utilizar el comparador no igual a en lugar del comparador igual a. Esto comprobará si el bloque en la posición actual del jugador no es igual a aire.

Cuando ejecutes el programa, asegúrate de que funciona cuando estás en el aire y cuando estás bajo el agua, en la lava, en la grava, en la arena o teletransportado al suelo. El mensaje enviado al chat cuando la condición es Verdadera se muestra en la [Figura 5-3](ch05.xhtml#ch5fig3).



*Figura 5-3: Dándome un agradable y relajante baño en el agua, que no es aire.*

#### **Mayor que y menor que**

Cuando necesites averiguar si un valor es mayor que otro, utiliza el comparador mayor que. El comparador mayor que devolverá Verdadero cuando el valor de la izquierda sea mayor que el de la derecha. Si el valor de la izquierda es menor o igual que el de la derecha, la comparación devolverá Falso.

Supongamos que tenemos una vagoneta minera que no puede levantar más de 99 bloques de obsidiana. Mientras el límite de elevación de nuestro minicarro sea mayor que el número de bloques de obsidiana que está intentando levantar, los bloques se podrán levantar:

>>> limit = 100  
>>> obsidian = 99  
>>> limit > obsidian  
True

¡Brillante! Nuestro minicarro puede transportar cualquier número de bloques de obsidiana que sea menor que 100, y 99 es menor que 100, por lo que límite > obsidiana se evalúa como Verdadero. Pero, ¿y si alguien añade otro bloque de obsidiana a la pila?

>>> limit = 100  
>>> obsidian = 100  
>>> canLift = limit > obsidian  
False

Oh, no, ¡ahora se ha alcanzado el límite! El resultado ahora es Falso: 100 no es mayor que 100; es lo mismo. Nuestro minicarro no puede levantar la obsidiana.

El comparador menor que funciona de la misma manera.

Una furgoneta que pasa por debajo de un puente necesita saber si es lo bastante pequeña para caber debajo de él:

>>> vanHeight = 8  
>>> bridgeHeight = 12  
>>> vanHeight < bridgeHeight  
True

En este caso, la furgoneta cabrá porque es más pequeña que la altura del puente: 8 es menor que 12. Más adelante, la misma furgoneta puede encontrarse con otro puente demasiado bajo para pasar por debajo:

>>> vanHeight = 8  
>>> bridgeHeight = 7  
>>> vanHeight < bridgeHeight  
False

Como 8 no es menor que 7, el resultado es Falso.

#### **Mayor o igual que y Menor o igual que**

Al igual que el comparador mayor que, el comparador mayor o igual que determina si un valor es mayor que otro. A diferencia del comparador mayor que, también se evaluará como Verdadero si los valores son iguales.

Supongamos que voy a dar pegatinas a todas las personas que vinieron a ver la presentación de mi increíble programa. Necesito comprobar si tengo suficientes pegatinas para todos:

>>> stickers = 30  
>>> people = 30  
>>> stickers >= people  
True

Tengo suficientes pegatinas: 30 es lo mismo que 30, por lo que pegatinas >= personas evalúa a Verdadero. Pero supongamos que a uno de mis amigos le parecen chulas las pegatinas y quiere una. Ahora, 31 personas quieren pegatinas:

>>> stickers = 30  
>>> people = 31  
>>> stickers >= people  
False

No tengo suficientes pegatinas: 30 no es mayor o igual que 31. Parece que mi amigo no puede tener una pegatina.

A estas alturas, ya estás preparado para enfrentarte a casi cualquier comparación. Mientras estás en IDLE, prueba también el comparador menor o igual que(<=) para ver cómo funciona.

**NOTA**

*Los comparadores mayor que, mayor o igual que, menor que y menor o igual que no funcionan con cadenas, aunque sí con enteros, flotantes y booleanos.*

#### **Misión nº 20: ¿Estoy por encima del suelo?**

La coordenada y de un jugador en Minecraft muestra la altura a la que se encuentra en el juego. Los bloques también se almacenan utilizando coordenadas, lo que te permite obtener los tipos de bloque en coordenadas concretas utilizando getBlock () y crear bloques en coordenadas concretas utilizando setBlocks().

Para obtener el bloque más alto de Minecraft, puedes utilizar la función getHeight(). La función toma una coordenada x y z y devuelve la coordenada y del bloque más alto en esa posición, como se muestra en el Listado [5-4](ch05.xhtml#ch5ex4).

*aboveGround.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
pos = mc.player.getTilePos()  
x = pos.x  
y = pos.y  
z = pos.z  
highestBlockY = mc.getHeight(x, z)  
mc.postToChat(highestBlockY)

*Listado 5-4: Código para encontrar la coordenada y del bloque más alto en la posición actual del jugador*

Este programa obtiene la posición actual del jugador, obtiene la coordenada y del bloque más alto en la posición del jugador, y luego publica este valor en el chat de Minecraft.

Combinando este programa con un comparador mayor que o igual a, puedes comprobar si el jugador está o no por encima del suelo. Hagámoslo ahora.

Copia el programa del [Listado 5-4](ch05.xhtml#ch5ex4) y guárdalo como *aboveGround.py*. Cambia el programa para comprobar si la coordenada y del jugador es mayor que la variable bloqueYmayor. Después, añade código para enviar el resultado al chat con el formato "El jugador está por encima del suelo: Verdadero/Falso".

**CONSEJO**

*Recuerda que puedes almacenar el resultado de una comparación en una variable. Por ejemplo, si quisiera comprobar si* y *es mayor o igual que 10 y almacenar la respuesta en una variable llamada* altoSuficiente*, utilizaría la siguiente sentencia*: altoSuficiente = y >= 10.

Ejecuta el programa cuando hayas hecho estos cambios. La salida de los resultados Falso del programa se muestran en la [Figura 5-4](ch05.xhtml#ch5fig4).



*Figura 5-4: Ahora estoy en una cueva, así que Python tiene mucha razón en que no estoy en la superficie.*

#### **Misión nº 21: ¿Estoy cerca de casa?**

Mientras vagas por el mundo de Minecraft, podrías perderte y olvidar dónde está tu casa. Podrías vagar durante horas sólo para descubrir que estabas cerca de casa cuando te perdiste por primera vez.

Con una sola línea de código, puedes comprobar a qué distancia estás de cualquier coordenada del juego. Por ejemplo, podrías utilizar las coordenadas de tu casa y tu posición actual para calcular a qué distancia estás. Añadiendo un comparador, también puedes comprobar si estás o no a un determinado número de manzanas de tu casa. Diremos que estás cerca de casa si estás a sólo 40 manzanas.

¡Escribamos un programa Python que compruebe esto por ti! El código de esta misión debe comprobar a qué distancia estás de tu casa, como se muestra en el Listado [5-5](ch05.xhtml#ch5ex5).

*farFromHome.py*

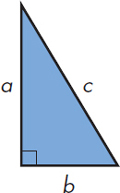
from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
import math  
➊ homeX = 10  
homeZ = 10  
pos = mc.player.getTilePos()  
x = pos.x  
z = pos.z  
➋ distance = math.sqrt((homeX - x) \*\* 2 + (homeZ - z) \*\* 2)  
➌ mc.postToChat(distance)

*Listado 5-5: Código que calcula la distancia a tu casa*

Este código supone que tu casa está en las coordenadas x = 10 y z = 10, que se establecen con las variables homeX y homeZ ➊. En este caso, no necesitamos conocer la coordenada y. Utilizo la función getTilePos() para obtener la posición del jugador y establecer los valores x y z.

Para calcular la variable distancia, utilizamos una fórmula llamada *teorema de Pitágoras*. Calcula la longitud de un lado de un triángulo rectángulo, y puedes utilizarla en Minecraft para calcular la distancia entre dos puntos. Es posible que hayas visto esta fórmula escrita en clase de matemáticas como a2 + b2 = c2, donde *a* y *b* son los dos catetos de un triángulo rectángulo, y *c* es la hipotenusa del triángulo, como se muestra en [la Figura 5-5](ch05.xhtml#ch5fig5). En ➋, estamos resolviendo para *c*, que está representada por la variable distancia.

Guarda [el listado 5-5](ch05.xhtml#ch5ex5) como *distanciaFromHome.py* en la carpeta de *booleanos*.



*Figura 5-5: Un triángulo rectángulo*

Para terminar el programa, utiliza un comparador menor o igual que para comprobar si el valor de la variable distancia es menor o igual que 40 y envía el resultado al chat con el formato "Tu casa está cerca: Verdadero/Falso" ➌. Utiliza la concatenación para combinar la cadena con el resultado de la comparación. Actualiza el contenido de la función postToChat() ➌ para mostrar la cadena.

Prueba el programa. Cuando estés a menos de 40 manzanas de tu casa, deberías recibir un mensaje Verdadero; cuando no estés a menos de 40 manzanas, deberías ver un mensaje Falso. [La Figura 5-6](ch05.xhtml#ch5fig6) muestra el programa en funcionamiento.



*Figura 5-6: Definitivamente, estoy a menos de 40 manzanas de mi casa. De hecho, ¡ahí está la puerta principal!*

[anterior](ch05_3.html)[Subtema 4 de 6: (Ver todo)](ch05.html)[siguiente](ch05_5.html)