Aprende a programar con Minecraft

## **6** **CREAR MINIJUEGOS CON SENTENCIAS IF**



En [el Capítulo 5](ch05.xhtml#ch05) aprendiste a hacer preguntas en Python. Utilizaste operadores de comparación (como ==, !=, >, <, etc.) y operadores lógicos(and, or, not) para averiguar si una condición o un conjunto de condiciones se evaluaban como Verdadero o Falso. En este capítulo, utilizarás las respuestas a estas preguntas -los resultados de las condiciones que pruebes- para decidir qué código ejecutar.

Todos los días tomas decisiones basadas en condiciones. ¿Es de noche? Si es así, te pones tu armadura de diamantes y llevas una espada para luchar contra los monstruos. Si no, puedes dejar todo tu equipo en tu base secreta. ¿Tienes hambre? Si es así, comes un poco de pan o una manzana. Si no, puedes irte a una gran aventura para abrir el apetito. Del mismo modo que tomas decisiones en la vida cotidiana, tus programas deben realizar tareas diferentes en función de una condición.

Utilizaremos un poco de código Python para ayudar a tus programas a tomar decisiones. Las sentencias if indican a tu programa si debe ejecutar o no un determinado fragmento de código . Una sentencia if significa "*Si* esta condición es verdadera, ejecuta este código". Por ejemplo, podrías comprobar si el jugador está en una sala prohibida y convertir el suelo en lava si es así. O podrías comprobar si ha colocado un bloque en un lugar determinado y abrir una puerta oculta si lo ha hecho. Utilizando condiciones y sentencias if, puedes empezar a crear tus propios minijuegos en Minecraft.

### **Uso de sentencias if**

Ser capaz de controlar la ejecución de tu programa es una habilidad muy poderosa; de hecho, ¡es crucial para la codificación! Los programadores a veces llaman a este concepto *control de flujo*. La forma más fácil de añadir este tipo de control es utilizar la simple sentencia if, que ejecuta código cuando una condición es Verdadera.

Una sentencia if tiene tres partes:

- El operador if

- Una condición a comprobar

- Un cuerpo de código que se ejecutará si la condición es Verdadera

Veamos una sentencia if en acción. El siguiente código imprimirá "Son muchos zombis" sólo si hay más de 20 zombis. De lo contrario, no hará nada.

zombies = int(input("Enter the number of zombies: "))  
if zombies > 20:  
print("That's a lot of zombies.")

Aquí, zombis > 20 es la condición que estamos comprobando, y print("Son muchos zombis.") es el cuerpo de la sentencia if; es el código que se ejecuta si zombis > 20 es Verdadero. Los dos puntos (:) al final de la línea if indican a Python que la siguiente línea iniciará el cuerpo de la sentencia if. La sangría indica a Python qué líneas de código forman este cuerpo. La *sangría* significa que hay un espacio extra al principio de una línea de texto. En Python las líneas tienen una sangría de cuatro espacios. Si quisiéramos añadir más líneas de código para ejecutarlas dentro de la sentencia if, pondríamos el mismo número de espacios delante de todas ellas, sangrándolas igual que print("Eso son muchos zombis").

Intenta ejecutar este código unas cuantas veces, probando cada condición, y observa qué ocurre. Por ejemplo, prueba a introducir un número menor que 20, el número 20, y un número mayor que 20. Esto es lo que ocurre si introduces 22:

Enter the number of zombies: 22  
That's a lot of zombies.

Vale, el resultado tiene sentido. Ejecutémoslo otra vez y veamos qué ocurre cuando no se cumple la condición.

Enter the number of zombies: 5

Observa que no ocurre nada si la condición es Falsa. El cuerpo de la sentencia if se ignora por completo. Una sentencia if ejecutará el código de su cuerpo sólo si la condición es Verdadera. Cuando finaliza la sentencia if, el programa continúa en la línea siguiente a la sentencia if.

Veamos otro ejemplo para entender mejor cómo funciona esto. El siguiente código utiliza una sentencia if para comprobar si una contraseña es correcta:

password = "cats"  
attempt = input("Please enter the password: ")  
if attempt == password:  
print("Password is correct")  
print("Program finished")

La expresión después de la sentencia if es la condición: intento == contraseña. La línea sangrada después de if intento == contraseña: es el cuerpo de la sentencia if: print("La contraseña es correcta").

El código imprimirá "La contraseña es correcta " sólo si el valor almacenado en la variable intento es el mismo que el valor de la variable contraseña. Si no son iguales, no imprimirá nada. La última línea se ejecutará e imprimirá " Programa finalizado" tanto si se ejecuta el cuerpo de la sentencia if como si no.

Ahora vamos a intentar algo un poco más explosivo.

#### **Misión 26: Explotar un cráter**

Ya has aprendido a hacer que el jugador se teletransporte y salte alto. Ahora harás desaparecer los bloques que rodean al jugador.

Cuando el programa se ejecute, los bloques situados encima, debajo y alrededor del jugador se convertirán en aire. Este poder es muy destructivo, así que ten cuidado cuando lo utilices. Para mayor seguridad, el programa preguntará al usuario si está seguro de querer destruir los bloques, y sólo lo hará si la respuesta del usuario es afirmativa.

[El Listado 6-1](ch06.xhtml#ch6ex1) crea un cráter alrededor del jugador borrando todos los bloques que hay encima, debajo y a su alrededor. Luego publica "¡Boom!" en el chat. Guarda este programa como *crater.py* en una nueva carpeta llamada *declaracionessi*.

*crater.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
answer = input("Create a crater? Y/N ")  
  
➊ # Add an if statement here  
  
pos = mc.player.getPos()  
➋ mc.setBlocks(pos.x + 1, pos.y + 1, pos.z + 1, pos.x - 1, pos.y - 1, pos.z - 1, 0)  
mc.postToChat("Boom!")

*Listado 6-1: Este código crea un cráter, independientemente de lo que introduzca el usuario.*

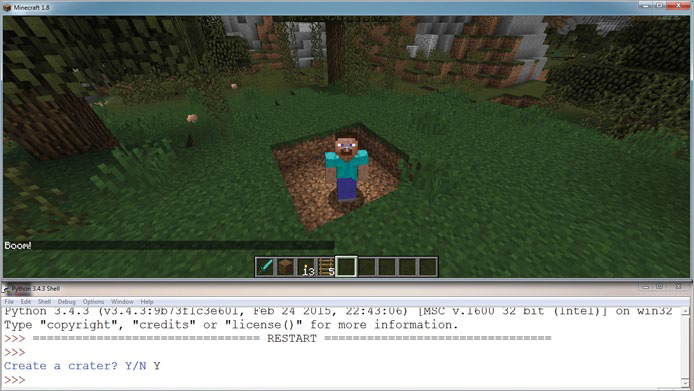
La variable respuesta utiliza la función input() para preguntar al usuario si desea crear un cráter. De momento, sin embargo, el código creará un cráter independientemente de lo que introduzca el usuario: Y, N, otra cosa o nada.

Para completar este programa, tendrás que añadir una sentencia if para comprobar si el usuario ha introducido Y en respuesta a la pregunta. Puedes añadir esa lógica al juego en ➊. Recuerda que la respuesta del usuario se almacena en la variable respuesta, por lo que tu sentencia if debe comprobar la variable respuesta. Después de añadir tu sentencia if, el programa sólo debe ejecutar las tres últimas líneas de código cuando el jugador introduzca Y. Para ello, debes sangrar estas líneas cuatro espacios.

Ten en cuenta que el último argumento de la función setBlocks( ) debe ser el tipo de bloque que quieres establecer. Aquí, el último argumento es 0, el tipo de bloque para el aire. En otras palabras, el cráter se crea utilizando setBlocks( ) para establecer los bloques en aire ➋, haciendo que parezca que todos los bloques alrededor del jugador han sido destruidos. Sumando y restando 1 a los valores de pos .x, pos.y y pos.z, el código coloca bloques de aire alrededor de la posición del jugador como un cubo de 3 por 3. Esto es el cráter.

Cuando hayas hecho los cambios en el programa, guárdalo y ejecútalo. Aparecerá la pregunta ¿Crear un cráter? S/N aparecerá en el intérprete de comandos Python. Introduce Y o N. Asegúrate de que la Y es mayúscula, o el programa no funcionará correctamente.

Cuando el usuario introduzca Y, aparecerá un cráter, como se muestra en la [Figura 6-1](ch06.xhtml#ch6fig1).



*Figura 6-1: ¡Boom! Hay un cráter a mi alrededor.*

**OBJETIVO EXTRA: CONSTRUIR UNA CASA**

¿Qué más puedes hacer con este programa? Prueba a cambiar el programa para que construya una casa alrededor del jugador en lugar de crear un cráter.

#### **sentencias else**

Ahora veremos una sentencia más avanzada que podemos utilizar si queremos ejecutar un fragmento de código *diferente* cuando la condición if sea False. Ahí es donde entra en juego la sentencia else.

Una sentencia else funciona junto con una sentencia if. Primero escribes una sentencia if para ejecutar código si la condición es Verdadera. Después del if, escribes una sentencia else para ejecutar otro código cuando la condición se evalúe como Falso. Es como si dijeras: "Si la condición es verdadera, haz esto. Si no, haz otra cosa".

El siguiente programa imprimirá "¡Ahhhh! Zombis!" si hay más de 20 zombis en una habitación; de lo contrario, imprimirá " Sabes, los zombis no sois tan malos".

zombies = int(input("Enter the number of zombies: "))  
if zombies > 20:  
print("Ahhhh! Zombies!")  
else:  
print("You know, you zombies aren't so bad.")

Al igual que la sentencia if, la sentencia else utiliza dos puntos y sangría para indicar qué código pertenece al cuerpo de la sentencia else. Pero la sentencia else no puede utilizarse por sí sola; debe ir precedida de un if. La sentencia else no tiene su propia condición; el cuerpo de la sentencia else sólo se ejecuta si la condición de la sentencia if(zombis > 20 en este ejemplo) no es Verdadera.

Volviendo al ejemplo anterior de la contraseña, podemos añadir una sentencia else para imprimir un mensaje cuando la contraseña sea incorrecta, así

password = "cats"  
attempt = input("Please enter the password: ")  
if attempt == password:  
print("Password is correct.")  
else:  
print("Password is incorrect.")

Cuando el valor de intento coincida con el valor de contraseña, la condición será Verdadero. El programa ejecuta el código que imprime "La contraseña es correcta ".

Cuando intento no coincida con contraseña, la condición será Falso. El programa ejecuta el código que imprime "La contraseña es incorrecta" .

¿Qué ocurre si se utiliza una sentencia else sin una sentencia if? Por ejemplo, si un programa tuviera sólo estas dos líneas:

else:  
print("Nothing happened.")

Python no entendería lo que está pasando y obtendría un error.

#### **Misión 27: Evitar el aplastamiento, o no**

En la Misión nº 17[(página 86](ch05.xhtml#page_86)), creaste un programa que impedía que el jugador rompiera bloques haciendo que el mundo fuera inmutable mediante mc.setting("mundo\_inmutable", Verdadero). El programa te ayudó a proteger tus preciosas creaciones de accidentes o vándalos. Pero aunque era útil, el programa no era muy flexible. Desactivarlo requería un segundo programa, ¡lo cual es bastante incómodo!

Utilizando una sentencia if, una sentencia else y la entrada de la consola, puedes crear un programa que active y desactive la inmutabilidad. Tu programa te preguntará si quieres que los bloques sean inmutables y, en función de tu respuesta, establecerá inmutable en Verdadero o Falso.

Abre IDLE y crea un nuevo archivo. Guarda el archivo como *inmutableChoice.py* en la carpeta *ifStatements*. Sigue estas instrucciones para completar el programa:

1. El programa debe preguntar al usuario si desea que los bloques sean inmutables:

* "Do you want blocks to be immutable? Y/N"
* Añade esta cadena como argumento dentro de input() y almacena la entrada en una variable llamada respuesta.

1. El programa comprueba si el valor almacenado en la variable respuesta es "Y". Si lo es, ejecuta el siguiente código:

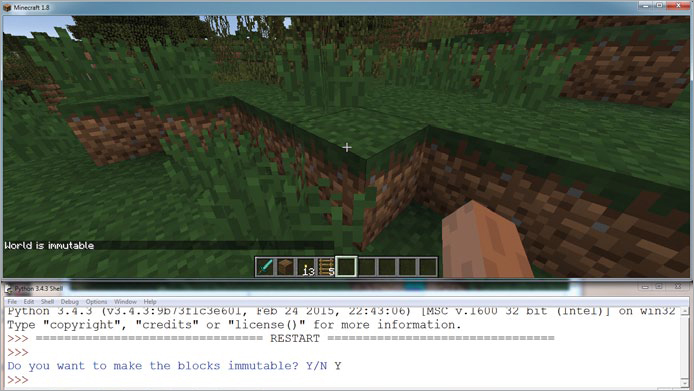
* mc.setting("world\_immutable", True)  
  mc.postToChat("World is immutable")
* Copia este código y ponlo en una sentencia if para que se ejecute sólo si el valor de la variable respuesta es igual a "Y ". ¡No olvides la sangría!

1. El programa ejecuta el siguiente código si la variable de respuesta no es "Y".

* mc.setting("world\_immutable", False)  
  mc.postToChat("World is mutable")
* Copia este código y ponlo en una sentencia else sangrada.

Guarda y ejecuta el programa. Cuando te pregunte si quieres que los bloques sean inmutables, escribe Y o N y pulsa INTRO. Prueba el programa. Si eliges que los bloques sean inmutables, no deberían romperse; en caso contrario, deberían romperse.

[La Figura 6-2](ch06.xhtml#ch6fig2) muestra el mensaje de salida y la pregunta en el intérprete de comandos. Obtendrás el mismo resultado si introduces "N" que si introduces datos sin sentido como "plátano". ¿Por qué crees que ocurre esto?



*Figura 6-2: Puedo elegir que el mundo sea inmutable, y ahora no puedo destruir ninguno de los bloques.*

**OBJETIVO EXTRA: UNA INTERFAZ MEJOR**

Podríamos hacer que este programa fuera más fácil de usar utilizando operadores booleanos para aceptar variaciones de "Sí" y "No", como "sí" en minúsculas, "SÍ" en mayúsculas y una respuesta de un solo carácter "y". ¡Inténtalo!

#### **Sentencias elif**

Utilizando una sentencia if y una sentencia else, tu programa pudo ejecutar cierto código si una condición era Verdadera y código diferente si la condición era Falsa. Pero, ¿y si quieres que se ejecuten más de dos bloques de código?

Para ello, puedes utilizar una sentencia else-if, o elif en Python. Primero escribes una sentencia if, luego escribes una sentencia elif y después escribes una sentencia else. Cuando utilizas estas sentencias juntas, estás diciendo: "Si una condición es Verdadera, ejecuta este código. De lo contrario, si una segunda condición diferente es Verdadera, ejecuta otro código. Por último, si ninguna de esas dos condiciones es Verdadera, ejecuta otro código".

Veámoslo. Supongamos que estás decidiendo qué sabor comprar en la heladería. Podrías decir: "Si queda helado de chocolate, lo compraré. Si no hay de chocolate, pero hay de fresa, cogeré el de fresa. Si no hay ni de chocolate ni de fresa, cogeré de vainilla".

En un programa, este proceso de decisión tiene este aspecto:

hasChocolate = False  
hasStrawberry = True  
if hasChocolate:  
print("Hooray! I'm getting chocolate.")  
elif hasStrawberry:  
print("I'm getting the second best flavor, strawberry.")  
else:  
print("Vanilla is OK too, I guess.")

Las dos primeras líneas sólo preparan el escenario: supondremos que hoy no queda chocolate en la heladería, pero sí fresa. Así que establecemos hasChocolate en Falso y hasFresa en Verdadero.

A continuación está la lógica del proceso de decisión: una sentencia if imprime "¡Hurra! Tengo chocolate" si tieneChocolate es Verdadero. Pero en este ejemplo, es Falso, así que ese mensaje no se imprime. En su lugar, el programa pasa a la sentencia elif y comprueba si tieneFresa es Verdadero. Como lo es, el código en el cuerpo de la sentencia elif se ejecuta e imprime "Tengo el segundo mejor sabor, fresa".

Como puedes ver, esta sentencia elif tiene su propia condición y cuerpo. La sentencia elif sólo se ejecuta cuando la condición de la sentencia if es False y la condición de la sentencia elif es True.

Por último, la sentencia else que sigue a la sentencia elif se ejecuta cuando la condición de la sentencia if es False y la condición de la sentencia elif también es False. En este ejemplo, el código de la sentencia else se ejecutaría si tanto hasChocolate como hasFresa fueran Falsas, imprimiendo "Supongo que la Vainilla también está bien".

Para ver otro ejemplo, volvamos al programa que imprimía "¡Ahhhh! Zombis!" si había más de 20 zombis en una habitación. Podemos añadir una sentencia elif para comprobar otra condición cuando la condición de la sentencia if sea False:

zombies = int(input("Enter the number of zombies: "))  
if zombies > 20:  
print("Ahhhh! Zombies!")  
elif zombies == 0:  
print("No zombies here! Phew!")  
else:  
print("You know, you zombies aren't so bad.")

Añadimos una sentencia elif para comparar zombis y 0. Si zombis == 0 es Verdadero, el código del cuerpo de la sentencia elif imprime " ¡Aquí no hay zombis! Uf!" Si la condición de esta sentencia elif es False, el código pasa a la sentencia else e imprime " Sabéis, los zombis no sois tan malos".

#### **Misión 28: Ofrece un regalo**

Vamos a crear un programa que compruebe si en un bloque determinado se ha colocado un regalo y que envíe al chat respuestas diferentes en función de cuál sea el regalo.

El programa te permitirá colocar uno de dos regalos diferentes. Un regalo es un bloque de diamantes, y como no todo el mundo tiene tantos bloques de diamantes, el otro es un árbol joven.

[El listado 6-2](ch06.xhtml#ch6ex2) comprueba si un bloque en la posición 10, 11, 12 es un bloque de diamante o un árbol joven o si no hay ningún regalo. Sin embargo, el programa no está completo.

*gift.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
x = 10  
y = 11  
z = 12  
gift = mc.getBlock(x, y, z)  
  
# if gift is a diamond block  
➊ if  
  
# else if gift is a sapling  
➋ elif  
  
else:  
mc.postToChat("Bring a gift to " + str(x) + ", " + str(y) + ", " + str(z))

*Listado 6-2: Inicio del código que comprueba si has entregado un regalo*

Crea un nuevo archivo en IDLE y guárdalo como *regalo.py* en la carpeta *ifDeclaraciones*. Copia [el Listado 6-2](ch06.xhtml#ch6ex2) en el archivo. El código que obtiene el tipo de bloque ya está hecho por ti. El tipo de bloque se almacena en la variable regalo. La sentencia else se ejecutará si no se ha colocado ni un bloque de diamante ni un árbol joven, y enviará un mensaje al chat indicando al jugador que lleve un regalo a esas coordenadas. Puedes cambiar las coordenadas de las variables x, y y z a la ubicación que desees.

Para completar el programa, sigue estos pasos

1. Completa la sentencia if en ➊ para que compruebe si la variable regalo contiene el valor de un bloque diamante (57). Si es así, haz que publique este mensaje en el chat "Gracias por el diamante".
2. Añade una sentencia elif bajo el segundo comentario en ➋ que compruebe si la variable de regalo contiene el valor de un árbol joven (6). Si es así, haz que publique este mensaje en el chat "Supongo que los arbolitos son tan buenos como los diamantes...".

Después de hacer los cambios, guarda y ejecuta el programa. Prueba a poner un bloque de diamante en las coordenadas y mira a ver qué pasa. Haz lo mismo con un árbol joven, y prueba también a no dejar nada en las coordenadas. No olvides que el arbolito debe estar plantado en un bloque de tierra o hierba. ¿Obtienes la respuesta correcta en cada situación? Tendrás que volver a ejecutar el programa cada vez para comprobar que funciona. [La Figura 6-3](ch06.xhtml#ch6fig3) muestra mi programa funcionando.



*Figura 6-3: He colocado un retoño de árbol como regalo.*

**OBJETIVO EXTRA: ALABAR AL DIOS MELÓN**

En esta misión puedes utilizar muchos bloques diferentes. Prueba a cambiar el código para que compruebe si has colocado un bloque de oro o un melón como regalo. Prueba a escribir un código que destruya el bloque regalo una vez lo hayas colocado.

#### **Encadenar sentencias elif**

No hay límite en el número de sentencias elif que puedes incluir con una sentencia if. Puedes tener una sentencia elif o 100 sentencias elif. Python simplemente las evalúa una tras otra.

He aquí un ejemplo utilizando el programa "número de zombis":

zombies = int(input("Enter the number of zombies: "))  
if zombies > 20:  
print("Ahhhh! Zombies!")  
➊ elif zombies > 10:  
print("There's just half a Minecraft zombie apocalypse.")  
elif zombies == 0:  
print("No zombies here! Phew!")  
else:  
print("You know, you zombies aren't so bad.")

Aquí hemos añadido una nueva sentencia elif en ➊, justo después de la sentencia if, para comprobar si hay más de 10 zombis en la habitación. Si los hay, imprime "Acaba de haber medio apocalipsis zombi Minecraft"; de lo contrario, el código pasa a la siguiente elif.

El orden de las sentencias if y elif es muy importante. Si las colocas en el orden incorrecto, es posible que nunca se llegue a parte del código, y tu programa no se ejecutará como se espera.

Por ejemplo, si intercambiamos la condición de la sentencia if con la condición de la primera sentencia elif, nos encontramos con un problema:

zombies = int(input("Enter the number of zombies: "))  
if zombies > 10:  
print("There's just half a Minecraft zombie apocalypse.")  
elif zombies > 20:  
print("Ahhhh! Zombies!")  
elif zombies == 0:  
print("No zombies here! Phew!")  
else:  
print("You know, you zombies aren't so bad.")

¿Por qué es un error? Veamos qué ocurre cuando los zombis son, digamos, 22. Como 22 es mayor que 10, la condición de la primera sentencia if, zombies > 10, es Verdadera, y el código de la sentencia if se ejecuta. Una vez que esto ocurre, no se ejecuta ninguna de las demás sentencias elif y else. El programa nunca llega a elif zombies > 20 porque ya ha ejecutado el cuerpo de la sentencia if. Esto es un error.

Si alguna vez obtienes resultados inesperados de tus sentencias if, comprueba siempre que tus sentencias if y elif están en el orden correcto.

#### **Misión nº 29: Teletransportarse al lugar correcto**

Cuando las sentencias if y elif están en el orden incorrecto, el código que esperas que se ejecute no se ejecutará, y el código que no esperas que se ejecute se ejecutará. Esto puede causar fallos extraños en tus programas. Para arreglar el programa, tienes que poner las condiciones en el orden correcto. Vamos a intentarlo.

[El listado 6-3](ch06.xhtml#ch6ex3) no funcionará. Se supone que debe teletransportar al jugador a diferentes lugares dependiendo de cuántos puntos introduzca el usuario. Los puntos coinciden con las ubicaciones correctas, pero parece que las condiciones no están en el orden correcto.

Cuantos más puntos tenga el jugador, mejor será la ubicación. Éste es el código. Las condiciones se establecen utilizando setPos() para cada transporte de ubicación.

*teleportScore.py*

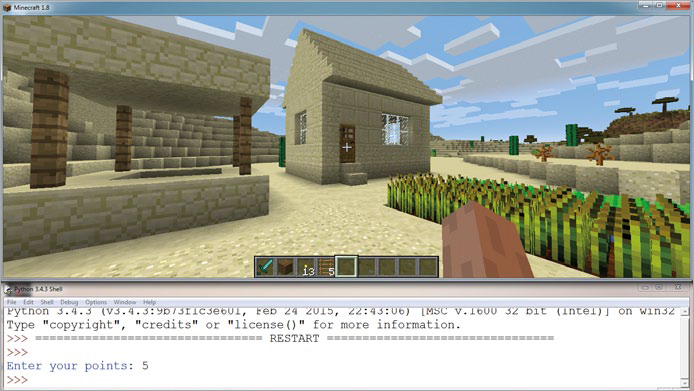
from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
points = int(input("Enter your points: "))  
if points > 2:  
mc.player.setPos(112, 10, 112)  
elif points <= 2:  
mc.player.setPos(0, 12, 20)  
elif points > 4:  
mc.player.setPos(60, 20, 32)  
elif points > 6:  
mc.player.setPos(32, 18, -38)  
else:  
mc.postToChat("I don't know what to do with that information.")

*Listado 6-3: Dependiendo de tus puntos, te teletransportarás a una ubicación diferente.*

Hay una condición distinta para más de 6 puntos, más de 4 puntos, más de 2 puntos, y luego 2 o menos puntos.

La última línea, dentro de la sentencia else, no se ejecutará a menos que el usuario introduzca algo totalmente extraño, como una cadena de texto en lugar de sus puntos, o no introduzca nada en absoluto.

Crea un nuevo archivo en IDLE y guárdalo como *teleportScore.py* en la carpeta *ifStatements*. Cambia el programa para que las condiciones estén en el orden correcto y se pueda llegar a todos los lugares. Prueba el programa con diferentes números de puntos para asegurarte de que el código de cada destino de teletransporte puede ejecutarse. La [Figura 6-4](ch06.xhtml#ch6fig4) muestra que el programa no funciona.



*Figura 6-4: ¡No esperaba acabar aquí!*

Como el programa no funciona en este momento, cuando introduzco 5, me teletransporta al lugar de más de 2 puntos, aunque debería ir al lugar de más de 4 puntos.

**OBJETIVO EXTRA: TRANSPÓRTAME, SCOTTY**

Crea un programa que te permita introducir un lugar al que quieras teletransportarte como una cadena, por ejemplo "castillo". Utiliza sentencias if y elif para elegir a qué lugar quieres teletransportarte. Por ejemplo, "fortaleza marina" te teletransportará a un lugar y "casa del árbol" te teletransportará a otro.

#### **Sentencias if anidadas**

Digamos que tienes una sentencia if, y si su condición es Verdadera, quieres probar otra condición (y ejecutar algún código si esta segunda condición es Verdadera). Por ejemplo, si intentas que la entrada a tu base sea extra secreta, podrías escribir algún código que compruebe si estás sobre un interruptor. Si es cierto, otra línea de código comprueba si estás sujetando el objeto secreto que desbloqueará la puerta. ¿Cómo lo harías?

Puedes poner una sentencia if dentro del cuerpo de otra sentencia if. Esto se conoce como sentencia if *anidada*.

[El listado 6-4](ch06.xhtml#ch6ex4) es un ejemplo de sentencia if anidada. Un simple cajero automático comprueba si tienes suficiente dinero y, en caso afirmativo, te pide que confirmes la retirada. Si lo confirmas, el programa realiza la retirada.

withdraw = int(input("How much do you want to withdraw? "))  
balance = 1000  
  
➊ if balance >= withdraw:  
confirm = input("Are you sure? ")  
➋ if confirm == "Yes":  
print("Here is your money.")  
else:  
print("Sorry, you don't have enough money.")

*Listado 6-4: Un cajero automático imaginario escrito con Python*

Observa que la segunda sentencia if está sangrada dentro de la primera sentencia if. Si la condición de la sentencia if externa ➊ es Verdadera, tienes suficiente dinero en tu cuenta, y se ejecuta la línea confirm = input("¿Estás seguro? "). Entonces, si la condición ➋ de la sentencia if interna es Verdadera, el código imprime "Aquí tienes tu dinero".

#### **Misión 30: Abrir un pasadizo secreto**

En esta misión, ampliarás un poco el ejemplo anterior. Crearás un edificio con un pasadizo secreto que sólo se abre cuando se coloca un bloque de diamante sobre un pedestal. Si se coloca cualquier otro tipo de bloque sobre el pedestal, ¡el suelo se convertirá en lava!

Primero, construye un edificio. Para hacerlo rápidamente, busca el programa *edificio.py*[(página 60](ch03.xhtml#page_60)) en la carpeta *matemáticas* y ejecútalo. No añadas ninguna puerta al edificio. Fuera, donde quieras codificar la entrada al edificio, coloca un solo bloque para representar el pedestal. Cuando coloques un bloque de diamante encima del pedestal, el código abrirá una entrada secreta en el lateral del edificio. [El Listado 6-5](ch06.xhtml#ch6ex5) proporciona un esqueleto de código que puedes utilizar para empezar.

*secretDoor.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
x = 10  
y = 11  
z = 12  
  
gift = mc.getBlock(x, y, z)  
if gift != 0:  
# Add your code here  
else:  
mc.postToChat("Place an offering on the pedestal.")

*Listado 6-5: Inicio del código para abrir una puerta secreta al colocar un regalo sobre un pedestal*

Crea un nuevo archivo en IDLE y guárdalo como puertaSecreta *.py* en la carpeta *ifDeclaraciones*. Cambia las coordenadas de este programa para que coincidan con la ubicación donde tendrás que colocar tu llave de bloque de diamante en tu mundo Minecraft.

Copia [el Listado 6-5](ch06.xhtml#ch6ex5) y añade código para estas tareas:

- Si hay un bloque diamante (57) en el pedestal, abre un pasadizo secreto a la habitación secreta. (Sugerencia: para crear una abertura en el edificio, prueba a poner los bloques al aire).

- Cuando un bloque que no sea diamante esté sobre el pedestal, haz que el suelo bajo el jugador se convierta en lava (10).

Tendrás que utilizar una sentencia if anidada para completar este programa.

Como se trata de un programa más complejo, debes construirlo y probarlo por etapas. Cuando hayas añadido una función, ejecuta el programa y asegúrate de que esa parte funciona antes de seguir adelante. Depurar trozos de código pequeños es más fácil que arreglar trozos de código largos. La [Figura 6-5](ch06.xhtml#ch6fig5) muestra la apertura del pasadizo secreto.



*Figura 6-5: El pasadizo secreto al templo ya está abierto.*

**OBJETIVO EXTRA: ESCALERA MECÁNICA**

¿Qué más podrías hacer cambiando el programa *puertasecreta.py*? ¿Podrías hacer una puerta automática que detectara cuando el jugador está junto a ella o una escalera mecánica que hiciera subir automáticamente al jugador por unas escaleras cuando se encuentra en la parte inferior?

### **Uso de sentencias if para comprobar un rango de valores**

Como aprendiste en [el Capítulo 5](ch05.xhtml#ch05), en Python puedes determinar si un valor se encuentra entre otros dos valores. Como una comprobación de rango se evalúa como Verdadero o Falso, puedes utilizar una comprobación de rango como condición en una sentencia if, igual que utilizarías una simple comparación menor que/mayor que o igual a/no igual a. Cualquier expresión que se evalúe como Verdadero o Falso puede ser una condición de una sentencia if.

Digamos que te has pasado todo el día reuniendo ingredientes para hornear unos deliciosos pasteles de Minecraft. Encuentras suficientes ingredientes para hornear 30 pasteles, y ahora quieres vender los pasteles. La persona que te compre pasteles debe comprar 1 pastel pero menos de 30; de lo contrario, no le venderás pasteles a esa persona. ¡No pueden acaparar todos los pasteles!

Este código representa la situación de los pasteles:

cakes = int(input("Enter the number of cakes to buy: "))  
➊ if 0 < cakes < 30:  
print("Here are your " + str(cakes) + " cakes.")  
➋ elif cakes == 0:  
print("Don't you want some delicious cake?")  
➌ else:  
print("That's too many cakes! Don't be selfish!")

Si la variable pasteles tiene un valor entre 0 y 30, como 15, imprimimos "Aquí tienes tus 15 pasteles". ➊. En caso contrario, imprimimos un mensaje diferente. Si pasteles tiene el valor 0, imprimimos "¿No quieres un delicioso pastel? ". ➋ y si es mayor que 30, imprimimos "¡Son demasiados past eles! No seas egoísta!" ➌.

Podemos probar una expresión más complicada añadiendo un operador booleano. Si fuera realmente raro y no quisiera que la gente comprara una cantidad de pan entre 20 y 30, podría hacerlo utilizando el operador no:

bread = int(input("Enter the amount of bread: "))  
if not 20 <= bread <= 30:  
print("Here are your " + bread + " breads.")  
else:  
print("I don't sell that amount of bread for some reason.")

Aquí utilizo un operador not y las comparaciones mayor que o igual a para comprobar un rango de valores como primera condición. La comprobación del rango determina si la cantidad de pan que la gente quiere comprar está entre 20 y 30. Entonces, el operador not pasa de Verdadero a Falso y de Falso a Verdadero. Por tanto, si el pan está dentro del intervalo, la expresión global se evalúa como Falso, y ejecutamos el código de la sentencia else. Si el pan no está en el rango entre 20 y 30 -digamos que es 40- la expresión global es Verdadera, e imprimimos "Aquí tienes tus 40 panes".

Si alguien intenta comprar 23 panes, no se lo permito. Pero 17 ó 32 está bien.

#### **Misión 31: Restringir las ubicaciones de teletransporte**

¿Recuerdas el programa de teletransporte que creaste en el [Capítulo 2](ch02.xhtml#ch02)? Se llamaba *teleport.py*. En esta misión, utilizarás comprobaciones de rango y sentencias if para limitar a dónde puede teletransportarse el jugador. Si utilizas Minecraft en la Raspberry Pi, hay lugares fuera del mundo del juego que no existen, pero tu programa te permitirá teletransportarte a ellos. Si estás utilizando la edición de escritorio de Minecraft, tu mundo es mucho más grande, por lo que no tienes las mismas restricciones que en la edición Pi del juego, pero este programa sigue siendo útil. Por ejemplo, podrías utilizarlo en un juego del escondite para limitar la zona donde pueden esconderse los jugadores.

[El listado 6-6](ch06.xhtml#ch6ex6) se supone que obtiene las coordenadas x, y y z de la entrada del usuario y las teletransporta a esa posición. Pero el programa no está completo.

*teleportLimit.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
valid = True  
  
x = int(input("Enter x: "))  
y = int(input("Enter y: "))  
z = int(input("Enter z: "))  
  
if not -127 < x < 127:  
valid = False  
  
# check if y is not between -63 and 63  
  
# check if z is not between -127 and 127  
  
if valid:  
mc.player.setPos(x, y, z)  
else:  
mc.postToChat("Please enter a valid location")

*Listado 6-6: Un programa para limitar los lugares a los que puede teletransportarse el jugador*

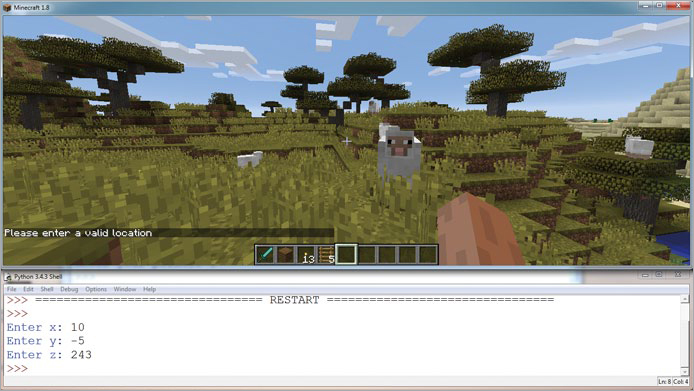
Para limitar a dónde puede teletransportarse el jugador, creamos una variable llamada válido. Esta variable almacenará un Verdadero o un Falso para representar si todas las coordenadas del destino son válidas o no. Pedimos al usuario que introduzca valores para x, y y z. A continuación, hacemos que una sentencia if compruebe si la variable x no está entre -127 y 127. Si no lo está, esta variable x no es válida. Si no lo está, esta coordenada x no es válida, y la variable válida se pone a Falso.

Cuando el programa llegue a la última sentencia if, sólo se llamará a setPos() si valid es True. Y válido será Verdadero sólo si se han cumplido las tres condiciones. De lo contrario, el jugador no podrá teletransportarse, y publicaremos un mensaje de chat indicando al usuario que introduzca una ubicación válida.

Crea un nuevo archivo en IDLE y copia en él [el Listado 6-6](ch06.xhtml#ch6ex6). Guarda el programa como *teleportLimit.py* en la carpeta *ifStatements*.

Completa el programa para que utilice sentencias if y comprobaciones de rango en las variables y y z y ponga valid en False si los valores no son válidos.

Cuando creas que has terminado el programa, ejecútalo. El programa debería teletransportarte cuando introduzcas valores que estén dentro del intervalo de -127 a 127 para las variables x y z y dentro del intervalo de -63 a 63 para la variable y. Cuando introduzcas un valor que no esté dentro de estos rangos, el programa no debería teletransportarte. [La Figura 6-6](ch06.xhtml#ch6fig6) muestra el aspecto que debería tener el juego cuando el usuario introduce un número no válido.



*Figura 6-6: La* variable z *era demasiado grande, por lo que no me teletransporté.*

**OBJETIVO EXTRA: MANTENTE SOBRE EL SUELO**

Uno de los problemas del programa de teletransporte es que puede teletransportarte bajo tierra, atrapándote allí. Puedes hacer cambios en el programa para impedir que el jugador se teletransporte bajo tierra. Compara la coordenada y que introduce el usuario con la función getHeight() para comprobar que el jugador se teletransporta por encima del suelo y detenerlo si se teletransporta bajo tierra.

### **Operadores booleanos y sentencias if**

En la misión anterior, utilizaste not en tus sentencias if. También puedes utilizar and y or. En este caso, la sentencia if actuará igual que con una simple condición: si la expresión global se evalúa como True, se ejecutará el cuerpo de la sentencia. He aquí un programa que pregunta a alguien si tiene pastel y si quiere darnos un poco de pastel:

hasCake = input("Do you have any cake? Y/N")  
wouldShare = input("Would you give me some cake? Y/N")  
if hasCake == "Y" and wouldShare == "Y":  
print("Yay!")  
else:  
print("Boo!")

Este código utiliza el operador y, por lo que Python sólo imprimirá "¡Sí!" si la persona tiene tarta(tieneTarta == "Y " es Verdadero) y está dispuesta a compartirla(compartiría= "Y " es Verdadero). Si alguna de estas comparaciones no es Verdadera, el código dentro de la sentencia else imprimirá "¡Bu! ".

Puedes sustituir y por el operador o para que Python imprima " ¡Sí!" si la persona tiene tarta o está dispuesta a compartirla:

hasCake = input("Do you have any cake? Y/N")  
wouldShare = input("Would you give me some cake? Y/N")  
  
if hasCake == "Y" or wouldShare == "Y":  
print("Yay!")  
else:  
print("Boo!")

Si hasPastel == "Y" o compartiría == "Y" es Verdadero, toda la expresión se evalúa como Verdadero e imprimimos " ¡Sí! La única vez que imprimimos "¡Bu !" es si ninguna de las dos condiciones es Verdadera: la persona no tiene tarta y no la compartiría si la tuviera.

Probemos a utilizar el operador not con una sentencia if:

wearingShoes = input("Are you wearing shoes? Y/N")  
if not wearingShoes == "Y":  
print("You're not wearing shoes.")

Este programa pide al usuario que introduzca Y si lleva zapatos y N si no los lleva. Almacena la entrada en llevandoZapatos. A continuación compara wearingShoes e "Y " para ver si son iguales. El operador not invierte el resultado de una comparación -Verdadero se convierte en Falso y Falso en Verdadero-, de modo quesi el usuario introdujo Y, la comparación empieza siendo Verdadera y not la convierte en Falsa, con lo que la expresión global es Falsa. No imprimimos ningún mensaje. Si el usuario no introdujo Y, la comparación es Falsa y no la convierte en Verdadera. La expresión global se evalúa como Verdadero, e imprimimos " No llevas zapatos".

#### **Misión nº 32: Ducharse**

Las mejores casas de Minecraft prestan mucha atención a los detalles. Mucha gente incluye suelos de madera, chimeneas y cuadros en sus casas para que se sientan más como en casa. Tú irás un paso más allá y harás una ducha que funcione.

Para que la ducha funcione, tendrás que utilizar comprobaciones de rango y operadores booleanos. Crearás una zona de ducha, y cuando el jugador entre en la ducha , el agua se abrirá. En otras palabras, cuando el jugador camine dentro de un rango de coordenadas, el programa deberá crear bloques de agua sobre ellas.

[El listado 6-7](ch06.xhtml#ch6ex7) proporciona la estructura básica del programa con unas pocas líneas de código para ayudarte a empezar. Es tu trabajo completar el resto.

*shower.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
➊ shwrX =  
shwrY =  
shwrZ =  
  
➋ width = 5  
height = 5  
length = 5  
  
pos = mc.player.getTilePos()  
x = pos.x  
y = pos.y  
z = pos.z  
  
➌ if shwrX <= x < shwrX + width and  
mc.setBlocks(shwrX, shwrY + height, shwrZ,  
shwrX + width, shwrY + height, shwrZ + length, 8)  
else:  
mc.setBlocks(shwrX, shwrY + height, shwrZ,  
➍ shwrX + width, shwrY + height, shwrZ + length, 0)

*Listado 6-7: El inicio del programa ducha*

Copia [el Listado 6-7](ch06.xhtml#ch6ex7) y guárdalo como *ducha.py* en la carpeta *declaracionessi*.

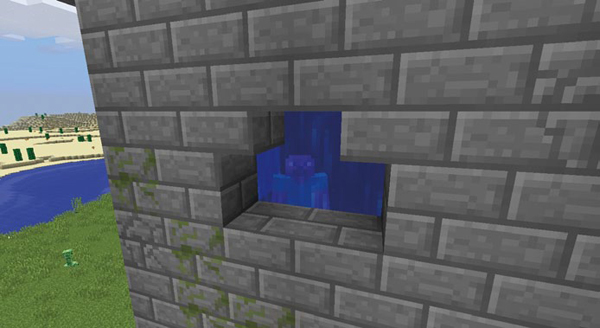
Para terminar el programa, añade las coordenadas de tu ducha en las variables shwrX, shwrY y shwrZ ➊. A continuación, añade el tamaño de la ducha en las variables anchura, altura y longitud ➋. He incluido un valor por defecto de 5, pero deberías cambiarlo para que tu ducha tenga el tamaño que tú quieras.

Termina la sentencia if para que compruebe si las variables y y z están dentro del área de la ducha ➌. He incluido la comprobación de la posición x para ayudarte(shwrX < x < shwrX + anchura). Las expresiones para las posiciones y y z son similares a ésta. Pista: Querrás combinar todas estas comprobaciones utilizando y.

La ducha se enciende y se apaga mediante la función setBlocks() ➍. Los bloques se establecen en agua (bloque ID 8) para encender la ducha y aire (bloque ID 0) para apagarla.

Las funciones setBlocks () de la última declaración if/else están divididas en dos líneas porque sus argumentos son muy largos. Python te permite hacer esto. Podrían escribirse en una sola línea; yo las escribí en dos para que fueran más fáciles de leer.

La Figura[6-7](ch06.xhtml#ch6fig7) muestra mi ducha funcionando.



*Figura 6-7: Aquí estoy, duchándome.*

Cuando ejecutes el programa, creará agua sobre ti si estás de pie en la ducha. El agua no se detendrá hasta que salgas de la ducha y vuelvas a ejecutar el programa. ¡Diviértete!

**OBJETIVO EXTRA: AHORRAR AGUA**

Añade un límite de tiempo que apague la ducha después de un tiempo determinado.

### **Lo que has aprendido**

Ahora tus programas pueden tomar decisiones. En este capítulo has aprendido a utilizar condiciones con las sentencias if, else y elif. En el [Capítulo 7](ch07.xhtml#ch07), aprenderás sobre los bucles *while*. Al igual que las sentencias if, los bucles while ayudan a tu programa a decidir qué hacer y cuándo. Pero a diferencia de las sentencias if y else -que se utilizan para ejecutar código si una condición es verdadera y código diferente si no lo es-, los bucles while ejecutan código *mientras* una condición es verdadera y lo siguen ejecutando repetidamente hasta que la condición se convierte en falsa.