Aprende a programar con Minecraft

## **2** **TELETRANSPORTE CON VARIABLES**



¿Estás preparado para controlar tu mundo de Minecraft con el poder de Python? En este capítulo, harás un breve recorrido por los fundamentos de Python. Luego pondrás a prueba tus nuevas habilidades y crearás tu propio recorrido de teletransporte por tu mundo de Minecraft.

Los conceptos descritos en este capítulo no son específicos de Minecraft Python, por lo que podrás utilizarlos en cualquier programa Python que crees.

### **¿Qué es un programa?**

Un *programa* es un conjunto de instrucciones que hace que tu ordenador realice una o varias tareas específicas. Imagina una aplicación de cronómetro en un teléfono móvil. El programa cronómetro tiene instrucciones que le dicen qué hacer cuando pulsas start y stop. También tiene instrucciones que muestran el tiempo en la pantalla mientras se cuenta. Alguien programó ese cronómetro para que funcionara.

Cada día se utilizan millones de programas en todo el mundo. La aplicación de mensajería de un teléfono es un programa, los semáforos están controlados por programas, e incluso los juegos de ordenador como Minecraft son programas.

En este libro, aprenderás los fundamentos de la programación y a escribir programas para que tus ideas cobren vida en Minecraft.

### **Almacenamiento de datos con variables**

Empecemos por aprender a almacenar datos con variables. *Las* variables te permiten almacenar datos para utilizarlos posteriormente en un programa. Los *datos* son cualquier información que quieras registrar, como números, nombres, cualquier tipo de texto, listas de elementos, etc. Por ejemplo, aquí tienes una variable llamada picos que almacena el valor numérico 12:

>>> pickaxes = 12

Las variables pueden almacenar números, palabras e incluso frases completas, como "¡Fuera de aquí, Enredadera!". También puedes cambiar las variables, lo que te permite hacer cosas muy interesantes en Minecraft. De hecho, ¡en breve utilizarás variables para aprovechar el poder del teletransporte!

Para crear una variable en Python, utilizarás un nombre de variable, un signo igual(=) y un valor. Digamos que estás a punto de emprender una gran aventura a través de muchos biomas de Minecraft; querrás llevar mucha comida contigo. Puedes representar la comida como una variable. Por ejemplo, en el siguiente shell de Python, pan es el nombre de la variable y 145 es el valor:

>>> bread = 145

El nombre de la variable siempre está a la izquierda del signo igual, y el valor que quieres almacenar siempre está a la derecha, como se muestra en la [Figura 2-1](ch02.xhtml#ch2fig1). Esta línea de código Python *declara* la variable pan y le *asigna* el valor 145.

image

*Figura 2-1: Partes de una declaración de variable. Debes tener mucha hambre si tienes 145 barras de pan.*

Después de declarar una variable y asignarle un valor, puedes introducir el nombre de la variable en el intérprete de comandos de Python para comprobar lo que contiene:

>>> bread  
145

Puedes utilizar casi cualquier nombre para una variable, pero es mejor utilizar un nombre que describa la finalidad de la variable para que entiendas lo que ocurre en tu programa. Aunque no es una regla, debes empezar los nombres de las variables con una letra minúscula en lugar de mayúscula. Este es un estilo que siguen los programadores de Python, y es una buena práctica que tú también sigas, para que otros puedan leer fácilmente tu código si alguna vez quieres compartirlo.

**NOTA**

*Aunque el* valor de una variable se almacena*, no* se guarda*. El valor de una variable se almacena en la memoria temporal del ordenador, lo que significa que cuando el ordenador se apaga o el programa deja de ejecutarse, el valor de la variable deja de almacenarse. Prueba a cerrar IDLE y a abrirlo de nuevo. Cuando intentes obtener el valor de* pan*, ¿qué ocurre?*

#### **La estructura de los lenguajes de programación**

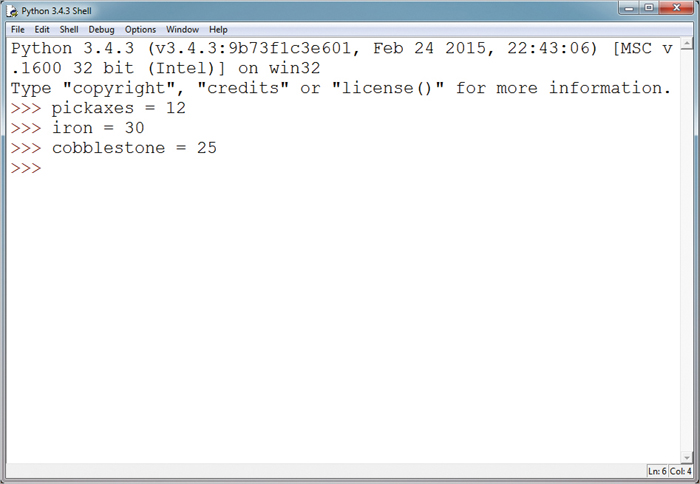
*La sintaxis* es el conjunto de reglas que describen la gramática y la puntuación de un lenguaje de programación, de forma similar a la gramática y la puntuación de un lenguaje humano. Una vez que comprendas la sintaxis de Python, serás más capaz de escribir programas que un ordenador pueda seguir; sin embargo, si no utilizas la sintaxis correcta, el ordenador no entenderá lo que le estás diciendo que haga.

Piensa en una sola instrucción de tu código como si fuera una frase. Para terminar una frase en inglés, se utiliza un punto (llamado punto y aparte en el Reino Unido). En lugar de un punto, Python utiliza una nueva línea para indicar el final de una instrucción y el comienzo de la siguiente. Cada instrucción en una nueva línea se denomina *sentencia*.

Por ejemplo, supongamos que quieres llevar la cuenta de cuántos picos, bloques de mineral de hierro y bloques de adoquines tienes. En el shell de Python, lo escribirías así:

>>> pickaxes = 12  
>>> iron = 30  
>>> cobblestone = 25

[La Figura 2-2](ch02.xhtml#ch2fig2) muestra el aspecto de esto en el intérprete de comandos de Python.



*Figura 2-2: Introducción de código en el intérprete de comandos Python*

Observa que cada sentencia está en su propia línea. Debido a las nuevas líneas, Python entenderá que quieres hacer un seguimiento de tres elementos diferentes. Pero si no pones cada sentencia en una nueva línea, Python se confunde y te da un error de sintaxis:

>>> pickaxes = 12 iron = 30 cobblestone = 25  
SyntaxError: invalid syntax

Un *error de sin* taxis es la forma que tiene Python de decirte que no lo entiende. Python no podrá seguir estas instrucciones porque no sabe dónde acaba una sentencia y empieza otra.

Python tampoco sabrá qué hacer si empiezas las líneas con un espacio:

>>> iron = 30  
SyntaxError: unexpected indent

Si te fijas bien, verás que el código tiene espacios al principio de la línea. Cuando obtengas un error sintáctico de sangría inesperada, como el que aparece aquí, sabrás que tu línea de código empieza con espacios cuando no debería.

Python es muy exigente con la forma en que escribes el código. Si obtienes un error de sintaxis al introducir los ejemplos de este libro, revisa cuidadosamente tu trabajo. Lo más probable es que encuentres un pequeño error.

#### **Reglas sintácticas para las variables**

Necesitas conocer algunas reglas sintácticas para nombrar variables, de modo que Python pueda entenderlas:

- No incluyas símbolos en los nombres de las variables, excepto guiones bajos(\_), o recibirás un error de sintaxis.

- No empieces el nombre de una variable con un número, como en 9pan. Utilizar números en otras partes del nombre de una variable está bien, como en pan9.

- No es necesario que añadas espacios a ambos lados del signo igual: tu programa funcionará bien sin ellos. Pero facilitan la lectura del código, por lo que es una buena idea añadirlos.

Las variables son muy útiles. A continuación, aprenderás a cambiar el valor de las variables, ¡y luego estarás listo para teletransportar a tu jugador!

#### **Cambiar el valor de las variables**

Puedes cambiar el valor de una variable en cualquier momento del mismo modo que declararías una variable. Por ejemplo, supongamos que te encuentras con cinco gatos Minecraft y quieres guardar este valor como variable. Primero declaras una variable, gatos, y le asignas el valor 5, que tendría este aspecto en una shell de Python:

>>> cats = 5  
>>> cats  
5

Más tarde te encuentras con cinco gatos más y decides que quieres actualizar este valor. ¿Qué ocurre si cambias el valor de gatos a 10?

>>> cats = 10  
>>> cats  
10

Cuando pides a Python el nuevo valor de gatos, ¡ya no es 5! Ahora, cuando utilices la variable gatos en un programa, utilizará el nuevo valor de 10.

Hay muchos tipos de datos que puedes almacenar en variables. Los *tipos* de datos indican al ordenador cómo trabajar con un dato concreto. Empezaré hablando de uno de los tipos que utilizarás más a menudo: los números enteros. Más adelante, también presentaré el tipo de datos flotantes.

#### **Enteros**

Los números*enteros* son números enteros positivos o negativos. Valores como 10, 32, -6, 194689 y -5 son números enteros, pero 3,14 y 6,025 no lo son.

Probablemente utilices números enteros todos los días sin pensar en ello, ¡incluso en Minecraft! Por ejemplo, puedes ver 12 vacas en una ladera mientras vas de camino a extraer 5 diamantes con 2 manzanas frescas en tu inventario. Todos esos números son enteros.

Supongamos que tienes 5 cerdos en tu mundo de Minecraft y quieres escribir un programa que utilice el número de cerdos de alguna manera. En Python, declararías una variable entera para representar el número de cerdos:

>>> pigs = 5

También puedes almacenar valores negativos en variables. Por ejemplo, para decir que la temperatura es de cinco grados negativos, establecerías una variable como ésta:

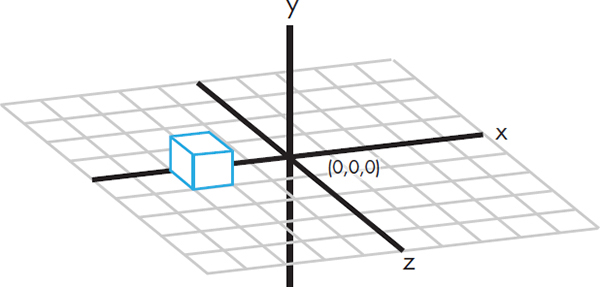
>>> temperature = -5

Para utilizar variables y enteros de Python con Minecraft, completa la primera misión.

#### **Misión nº 1: Teletransportar al Jugador**

En esta misión, explorarás cómo funcionan las variables teletransportando a tu jugador a una nueva ubicación utilizando números enteros.

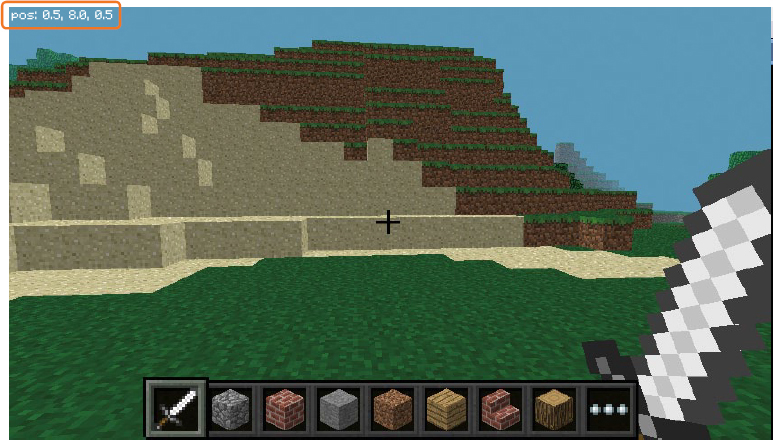
Como se muestra en la [Figura 2-3](ch02.xhtml#ch2fig3), tu jugador tiene una *posición* en el mundo de Minecraft que está representada por tres *coordenadas*: x, y y z. La letra y representa la altura, y x y z representan posiciones horizontales en un plano llano.



*Figura 2-3: Coordenadas 3D*

Si estás utilizando la versión Raspberry Pi del juego, la posición del jugador viene dada por tres números en la esquina superior izquierda de la ventana del juego, que puedes ver en la [Figura 2-4](ch02.xhtml#ch2fig4). Si utilizas la edición de escritorio del juego, puedes ver las coordenadas del jugador pulsando F3 y encontrando la primera línea en el segundo bloque de texto de la izquierda, etiquetado *XYZ*, como se muestra en la Figura [2-5](ch02.xhtml#ch2fig5).

Mueve a tu jugador por el juego y observa cómo cambian los números de posición; las coordenadas deberían actualizarse en tiempo real a medida que el jugador camina. Genial, ¿verdad? Pero caminar largas distancias lleva mucho tiempo. ¿Por qué perder tanto tiempo caminando cuando puedes cambiar de posición instantáneamente utilizando Python? Veamos cómo hacerlo.



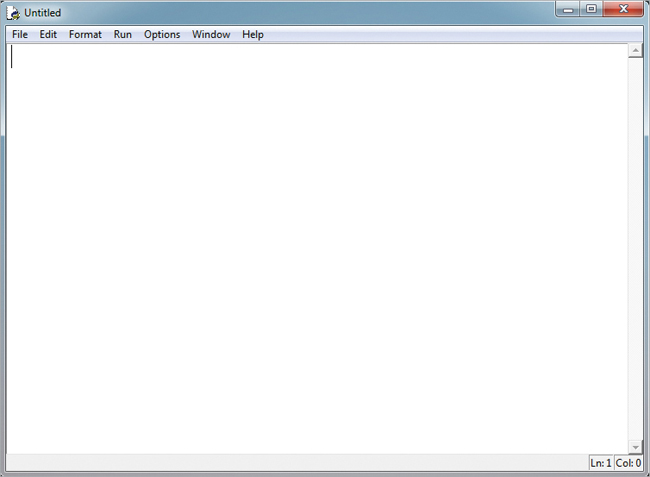
*Figura 2-4: Posición del jugador mostrada en Minecraft: Edición Pi*



*Figura 2-5: La posición del jugador mostrada en la edición de escritorio de Minecraft*

Enciende tu ordenador o tu Raspberry Pi y sigue estos pasos:

1. Abre IDLE y haz clic en **Archivo** image **Nuevo Archivo** (o **Nueva Ventana** en algunos ordenadores). Puedes ver la ventana vacía del editor de texto en la [Figura 2-6](ch02.xhtml#ch2fig6). Si utilizas una Raspberry Pi o tienes más de una versión de Python instalada en tu ordenador, asegúrate de que utilizas Python 3, no Python 2.7.

* 
* *Figura 2-6: Una nueva ventana de editor de texto en IDLE*

1. Cuando aparezca la nueva ventana, haz clic en **Archivo** image **Guardar como**.
2. Crea una nueva carpeta llamada *variables* dentro de la carpeta *Minecraft Python* que creaste en el [Capítulo 1](ch01.xhtml#ch01).
3. Abre la carpeta *variables*, nombra tu archivo *teleport.py*, y haz clic en **Guardar**.

Ahora que estás trabajando en el editor de texto de IDLE, añade las siguientes dos líneas de código en la parte superior de tu programa:

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()

Estas líneas conectan tu programa con Minecraft; las utilizarás en todos los programas que interactúen con Minecraft. A continuación, crea tres variables enteras llamadas x, y y z.

x = 10  
y = 110  
z = 12

Estas variables representan la posición a la que quieres teletransportar a tu jugador. Por ahora, establece estas variables en 10, 110 y 12, como se muestra aquí.

A continuación, introduce la siguiente línea de código, que transportará al jugador:

mc.player.setTilePos(x, y, z)

La parte del programa setTilePos( ) es una *función*, que es un trozo de código preescrito y reutilizable. La función setTilePos(x, y, z) le dice a Minecraft que cambie la posición del jugador utilizando las tres variables que acabas de establecer. Los valores dentro de los paréntesis se llaman *argumentos*. Has *pasado* las variables que acabas de crear a la función como argumentos para que la función pueda utilizar los valores de x, y y z cuando se ejecute.

**ADVERTENCIA**

*Si utilizas una Raspberry Pi, no utilices valores superiores a 127 ni inferiores a -127 para las* variables x *y* z *. El mundo de Minecraft Pi es pequeño, y los números fuera de este rango bloquearán el juego.*

[El Listado 2-1](ch02.xhtml#ch2ex1) contiene el código completo para teletransportar a tu jugador, que también puedes ver en la [Figura 2-7](ch02.xhtml#ch2fig7):

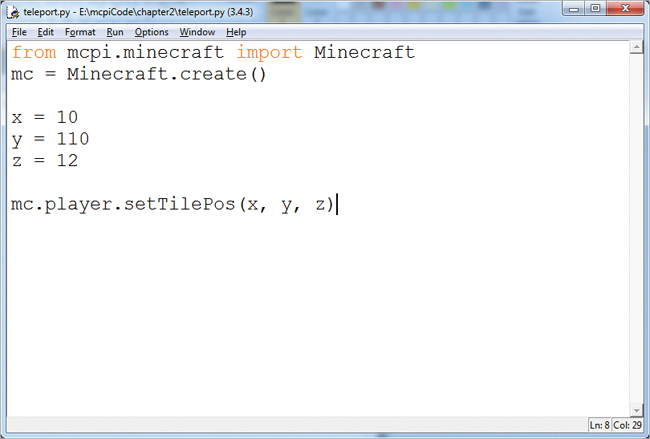
*teleport.py*

➊ # Connect to Minecraft  
from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
# Set x, y, and z variables to represent coordinates  
x = 10  
y = 110  
z = 12  
  
# Change the player's position  
mc.player.setTilePos(x, y, z)

*Listado 2-1: El código de teletransporte terminado*

Para que este programa sea más fácil de entender, he incluido algunos *comentarios* ➊. Los comentarios son declaraciones útiles en el código que describen lo que éste hace, pero que Python ignora. En otras palabras, cuando ejecutas el programa, Python pasa las líneas comentadas sin hacer nada. Un comentario de una sola línea comienza con una almohadilla(#). Mis comentarios describen lo que hace cada parte de *teleport.py*. Es una buena costumbre escribir comentarios en tu código para que puedas recordar lo que hacen las partes de tu programa cuando vuelvas a ellas más tarde.

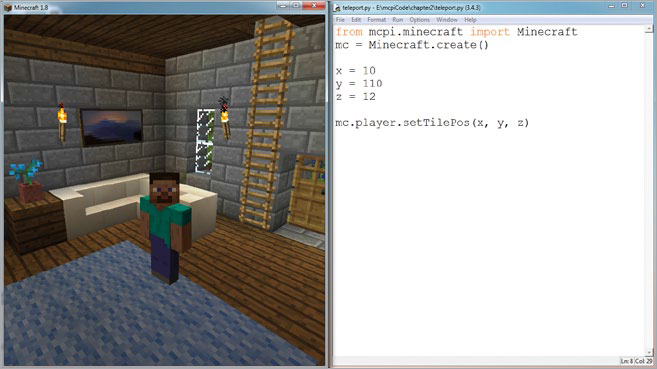
[La Figura 2-7](ch02.xhtml#ch2fig7) muestra el programa completo escrito en el editor de texto de IDLE.



*Figura 2-7: Programa completo en el editor de texto de IDLE*

Ahora, ¡a ejecutar el programa! Sigue estos pasos:

1. Abre Minecraft haciendo clic en el icono del escritorio.
2. Si utilizas una Raspberry Pi, haz clic en **Iniciar Juego** y en **Crear un Nuevo Mundo**. Si utilizas la edición de escritorio de Minecraft, abre el mundo del juego siguiendo las instrucciones de "[Ejecutar Spigot y crear un perfil de juego](ch01.xhtml#ch01lev2sec05)" en la [página 7](ch01.xhtml#page_7) para Windows y en [la página 18](ch01.xhtml#page_18) para Mac.
3. Una vez generado el mundo, pulsa la tecla ESC (o TAB si utilizas una Raspberry Pi) para soltar el ratón. Ahora puedes mover el ratón fuera de la ventana de Minecraft o hacer doble clic en la ventana de Minecraft para volver a seleccionar el juego. [La Figura 2-8](ch02.xhtml#ch2fig8) muestra las ventanas IDLE y Minecraft en mi ordenador.

* 
* *Figura 2-8: Así es como me gusta organizar mis ventanas de Minecraft y del editor de texto IDLE.*

1. Haz clic en la ventana del editor de texto IDLE que tiene tu programa *teleport.py*.
2. Haz clic en **Ejecutar** image **Ejecutar módulo** o pulsa F5. Si no has guardado el programa, IDLE siempre te preguntará si quieres guardarlo antes de ejecutarlo. Pulsa **Aceptar** para guardar el programa. Si haces clic en Cancelar, el programa no se ejecutará.

**NOTA**

*Cuando ejecutes programas desde IDLE en una Raspberry Pi, es posible que aparezca un cuadro de diálogo pidiéndote que guardes el programa y se oculte tras la ventana de Minecraft. Si crees que IDLE se ha congelado, puede ser que el diálogo se esté ocultando. Sólo tienes que minimizar la ventana de Minecraft y hacer clic en Aceptar en el diálogo de IDLE. Tras hacer clic en Aceptar, maximiza la ventana de Minecraft.*

¡Ya está! Tu programa debería ejecutarse ahora, y tras unos segundos, tu jugador debería ser teletransportado a las coordenadas (10, 110, 12), como se muestra en la [Figura 2-9](ch02.xhtml#ch2fig9). Tu mundo no es igual al mío, así que verás algunas diferencias cuando lo ejecutes en tu ordenador.

**OBJETIVO EXTRA: SALTA**

¿Crees que ya le has pillado el truco al teletransporte? ¡Prueba a sustituir x, y y z por otros números enteros para ver dónde acaba tu jugador! ¡Prueba también con valores negativos!



*Figura 2-9: Me he teletransportado desde mi casa hasta la posición (10, 110, 12), que está encima de un pantano. ¡Cuidado abajo!*

#### **Flotantes**

No todos los números son enteros. Los decimales se utilizan para representar valores que no pueden describirse con números enteros. Por ejemplo, puedes tener la mitad (0,5) de una manzana. Los números que utilizan puntos decimales se llaman *números de coma flotante*, o *flotantes*. Éste es otro tipo de datos que utiliza Python. Los números de coma flotante se utilizan en lugar de los enteros cuando quieres ser más preciso. Los flotantes también pueden representar números enteros (como en 3.0), pero los enteros no pueden representar números con decimales.

Te habrás dado cuenta de que las coordenadas de posición de tu jugador (como se muestra en [las Figuras 2-4](ch02.xhtml#ch2fig4) y [2-5](ch02.xhtml#ch2fig5)) incluyen decimales, ¡lo que significa que son flotantes!

En Python, declaras una variable flotante del mismo modo que declaras una variable entera. Por ejemplo, para establecer la variable x en 1,34, escribirías esto:

>>> x = 1.34

Para crear una variable flotante negativa, pon un signo menos(-) delante del valor:

>>> x = -1.34

En la siguiente misión, controlarás aún más tus poderes de teletransporte utilizando variables flotantes para teletransportar al jugador a lugares precisos.

#### **Misión nº 2: Ve exactamente donde quieras**

Aprendiste a fijar la posición del jugador utilizando números enteros, pero puedes fijar la posición del jugador con más precisión si utilizas flotadores. En esta misión, revisaremos el programa de la Misión nº 1 para teletransportar al jugador utilizando un valor flotante:

1. En IDLE, abre el programa *teleport.py*[(página 38](ch02.xhtml#page_38)) haciendo clic en **Archivo** image **Abrir** y seleccionando el archivo de tu carpeta *de variables*.
2. Guarda una copia del programa como teleportPrecise *.* py en tu carpeta *de* variables.
3. En el archivo *teleportPrecise*.py, cambia las variables x, y y z para que utilicen valores flotantes en lugar de enteros. Es decir, cambia los valores de x, y y z de 10, 110 y 12 a 10,0, 110,0 y 12,0.
4. Cambia la última línea de código a mc.player.setPos(x, y, z) eliminando la palabra Mosaico.
5. Guarda el programa.
6. Abre un mundo Minecraft y ejecuta el código.

El resultado final debería ser así:

*teleportPrecise.py*

# Connect to Minecraft  
from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
# Set x, y, and z variables to represent coordinates  
x = 10.0  
y = 110.0  
z = 12.0  
  
# Change the player's position  
mc.player.setPos(x, y, z)

Observa la diferencia entre mc. player.setPos(x, y, z), utilizado aquí, y mc.player.setTilePos(x, y, z), utilizado en [el Listado 2-1](ch02.xhtml#ch2ex1). La función setTilePos () utiliza números enteros para indicar al juego la posición del bloque al que quieres teletransportarte. La función setPos () es un poco diferente: utiliza valores flotantes para indicar al juego la posición del bloque, así como la posición exacta *en* ese bloque a la que quieres teletransportarte. Utilizando mi programa, me teletransporté a la parte superior de mi torre, como se muestra en la [Figura 2-10](ch02.xhtml#ch2fig10).



*Figura 2-10: Me he teletransportado a la parte superior de mi torre, utilizando flotadores para ser muy preciso.*

**OBJETIVO EXTRA: TELETRANSPORTARSE CON PRECISIÓN**

Cambia los valores de las variables x, y y z utilizando una mezcla de flotantes positivos y negativos y ejecuta el programa. Después, cambia ligeramente los valores decimales de tus nuevos valores. ¿Qué ocurre?

### **Ralentizar el teletransporte utilizando el módulo tiempo**

Python ejecuta tu código lo más rápido posible. Pero puedes ralentizar la acción haciendo que tus programas esperen un cierto número de segundos antes de continuar.

Para utilizar el tiempo en tus programas, necesitas el módulo tiempo, que contiene un conjunto de funciones preescritas relacionadas con el tiempo. Para utilizar el módulo tiempo, añade la siguiente línea de código al principio de tus programas:

import time

El orden es muy importante cuando utilices el módulo del tiempo y la función sleep(), que forma parte del módulo del tiempo. La función sleep () hará que un programa espere un número determinado de segundos antes de continuar. Siempre debes importar el módulo del tiempo antes de utilizar la función sleep( ). Si no lo haces, Python no podrá encontrar la función dormir( ) y se confundirá tanto que detendrá la ejecución de tu programa. Por eso es mejor importar cualquier módulo que utilices al principio de tu código. Todas tus declaraciones de importación se agruparán en la parte superior del programa. Por ejemplo, yo suelo incluir primero las líneas de código para conectarme a Minecraft, y luego añado la sentencia import time en la tercera línea.

Aquí tienes un ejemplo de cómo utilizar la función sleep():

time.sleep(5)

Esta línea de código pausa tu programa durante cinco segundos. Puedes utilizar cualquier número, incluidos enteros y flotantes, como muestra el siguiente ejemplo:

time.sleep(0.2)

Cuando tu programa llegue a esta línea de código, esperará 0,2 segundos. Ahora que ya puedes controlar el flujo del tiempo, ¡estás listo para la siguiente misión!

#### **Misión 3: Recorrido de teletransporte**

La belleza del teletransporte en Minecraft es que puedes enviar a tu jugador a cualquier parte. Usando todas las habilidades que has aprendido hasta ahora, ¡enviarás a tu jugador a un tour automatizado por todo tu mundo Minecraft!

En esta misión, practicarás cambiando los valores de las variables modificando el código de la Misión nº 1[(página 35](ch02.xhtml#page_35)) para teletransportar al jugador a varios lugares del mapa. El jugador se teletransportará a un lugar, esperará unos segundos y luego se teletransportará a otro lugar.

1. En IDLE, abre el programa *teleport.py*[(página](ch02.xhtml#page_38) 38) haciendo clic en **Archivo** image **Abrir** y seleccionando el archivo de tu carpeta *de variables*.
2. Guarda una copia del programa como *gira.* py en tu carpeta *de* variables.
3. Justo después del código que conecta tu programa con Minecraft, añade import time.
4. Al final del programa, añade time.sleep(10).
5. Copia las líneas con las variables x, y, y z y la función setTilePos() y pégalas al final del programa, para que esas líneas aparezcan dos veces.
6. Cambia los valores de ambos conjuntos de variables x, y y z por los números que quieras. Puedes encontrar las coordenadas de cualquier posición de tu juego moviéndote hasta allí y anotando las coordenadas del jugador como hiciste anteriormente en este capítulo.
7. Guarda el programa.
8. Abre un mundo Minecraft y ejecuta el código.

El resultado final debería ser así, con las nuevas coordenadas rellenadas:

*tour.py*

# Connect to Minecraft  
from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
import time  
  
# Set x, y, and z variables to represent coordinates  
x = # Fill in  
y = # Fill in  
z = # Fill in  
  
# Change the player's position  
mc.player.setTilePos(x, y, z)  
  
# Wait 10 seconds  
time.sleep(10)  
  
# Set x, y, and z variables to represent coordinates  
x = # Fill in  
y = # Fill in  
z = # Fill in  
  
# Change the player's position  
mc.player.setTilePos(x, y, z)

El jugador debe teletransportarse al primer lugar, esperar 10 segundos y, a continuación, teletransportarse al segundo lugar, como se muestra en la [Figura 2-11](ch02.xhtml#ch2fig11).

**OBJETIVO EXTRA: MÁS TELETRANSPORTE**

Copia el código de *tour.py* para mover al jugador tantas veces como quieras. Sustituye el 10 de la función time.sleep(10) por un valor distinto. Incluso podrías utilizar un número distinto para cada función sleep (), de modo que el jugador espere un tiempo distinto en cada lugar.

Luego edita el código para que entre teletransporte y teletransporte sólo cambie una de las variables x, y y z. ¡No hace falta que cambies todas las variables cada vez! Prueba también a utilizar valores flotantes en lugar de enteros.

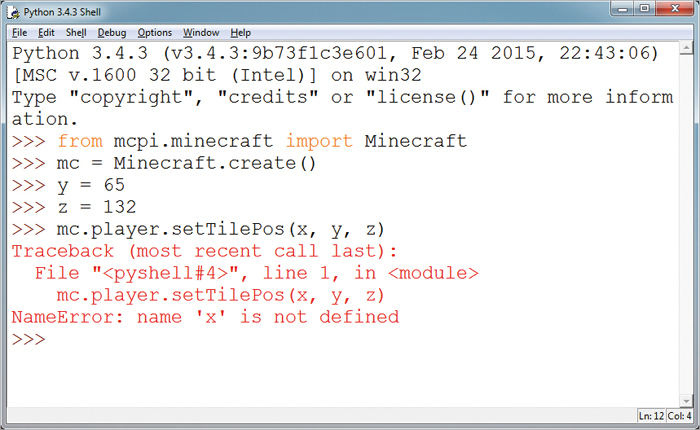


*Figura 2-11: He establecido las coordenadas en mi programa para teletransportarme a mi casa y luego teletransportarme al desierto.*

### **Depurando**

Todo el mundo comete errores; a menudo, ni siquiera los mejores programadores aciertan con su código a la primera. Escribir un programa que funcione es sólo una de las habilidades que necesita un buen programador. Arreglar los programas cuando no funcionan es otra habilidad esencial. Este proceso se denomina *depuración*, y cada problema en un programa que se comporta mal se denomina *error*. En esta sección, aprenderás trucos y consejos para arreglar todos tus futuros programas.

Los fallos pueden impedir completamente que un programa se ejecute, o pueden hacer que el programa se comporte de forma inesperada. Cuando un programa no se ejecuta, Python te mostrará un mensaje de error, como el de la [Figura 2-12](ch02.xhtml#ch2fig12).



*Figura 2-12: Python me da un mensaje de error porque no me he atenido a la sintaxis de Python.*

En la Figura [2-12](ch02.xhtml#ch2fig12) puedes ver que he introducido algo de código en el intérprete de comandos de Python, y me ha devuelto un mensaje de error. En el mensaje de error aparece mucha información, pero basándome en la última línea(NameError: el nombre 'x' no está definido) puedo decir que algo va mal con mi variable x. Concretamente, la variable x no está definida. Para solucionarlo, tengo que añadir una línea de código adicional que defina la variable x, así:

>>> x = 10

Esta línea arreglará el mensaje de error, pero no significa que se arreglen todos los errores.

Los errores que permiten que el programa se ejecute pero hacen que se comporte de forma extraña no mostrarán un mensaje de error, pero sabrás que algo va mal cuando tu programa produzca un resultado inesperado. Por ejemplo, si olvidas escribir una línea de código en tus programas de teletransporte, como setTilePos(), el programa se ejecutará sin errores, pero el jugador no cambiará de posición. ¡Eso no es un programa de teletransporte muy útil!

**ADVERTENCIA**

*Las erratas son una de las causas más comunes de errores. Escribir algo de una forma que el ordenador no espera puede impedir que tu programa se ejecute. Ten cuidado y asegúrate de que la ortografía y las mayúsculas son correctas.*

#### **Misión nº 4: Arreglar el Bug del Teletransporte**

En esta misión, depurarás dos programas. El primer programa, [Listado 2-2](ch02.xhtml#ch2ex2), es similar a *teleport.py*[(página 38](ch02.xhtml#page_38)), pero esta versión tiene cinco errores. Abre un nuevo archivo en el editor de texto IDLE, copia en él el Listado [2-2](ch02.xhtml#ch2ex2) y guárdalo como *teleportBug1.py*.

*teleportBug1.py*

from mcpi.minceraft inport Minecraft  
# mc = Minecraft.create()  
  
x = 10  
y = 11  
z = 12

*Listado 2-2: Una versión rota del programa teletransporte*

Para depurar este programa, sigue los siguientes pasos:

1. Ejecuta teleportBug1 *.*py.
2. Cuando aparezca un mensaje de error, lee la última línea para obtener una pista sobre lo que está mal.
3. Corrige el error y vuelve a ejecutar el código.
4. Sigue corrigiendo los errores hasta que el programa teletransporte al jugador a una nueva ubicación.

**PISTA**

¡*No olvides volver a comprobar que realmente estás llamando a la* *función* setTilePos() *!*

Intentemos depurar otro programa. La versión de *teleport.py* del Listado [2-3](ch02.xhtml#ch2ex3) se ejecuta, pero por alguna razón, el jugador no se teletransporta a la posición especificada. Copia [el Listado 2-3](ch02.xhtml#ch2ex3) en un archivo IDLE y guárdalo como *teleportBug2.py*.

*teleportBug2.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
x = 10  
y = 110  
z = -12  
  
mc.player.setPos(x, z, y)

*Listado 2-3: El programa de teletransporte con errores*

A diferencia de *teleportBug1.py*, no obtendrás ningún mensaje de error cuando ejecutes el programa. Para arreglar este programa, tendrás que leer el código hasta encontrar el error. El programa debería teletransportar al jugador a la posición (10, 110, -12). Ejecuta el programa y comprueba las coordenadas a las que se ha teletransportado el jugador. Esto puede ayudarte a depurar el programa e identificar el problema que tiene.

Cuando hayas eliminado todos los errores de estos dos programas, añade un comentario a cada uno para explicar cuáles eran los problemas. Anotar los problemas que encuentres en la depuración puede ayudarte a recordar que debes estar atento a errores similares en el futuro.

### **Lo que has aprendido**

¡Enhorabuena! Has escrito tus primeros programas en Python para controlar a un jugador de Minecraft mediante el poder de las variables y las funciones. Exploraste dos tipos de datos (enteros y flotantes), controlaste el tiempo y depuraste programas rotos. También aprendiste dos útiles funciones específicas de la API Python de Minecraft: setPos() y setTilePos().

En [el Capítulo 3](ch03.xhtml#ch03), dominarás el arte de la construcción rápida en Minecraft, ¡utilizando operaciones matemáticas y funciones que establecen bloques!