Aprende a programar con Minecraft

## **4** **HABLAR CON CADENAS**



En los [Capítulos 2](ch02.xhtml#ch02) y [3](ch03.xhtml#ch03), trabajaste con enteros y flotantes, que son tipos de números. En este capítulo, utilizarás otro tipo de datos llamado *cadenas*. Puedes utilizar cadenas para trabajar con letras y símbolos, además de con números.

Las cadenas te ayudan a mostrar datos a las personas que utilizan tus programas, una parte importante de la programación. Usando cadenas, puedes decirle a Python que *envíe* datos a la pantalla, que muestre y comunique información al usuario.

En Minecraft puedes utilizar cadenas en varios lugares, como para enviar mensajes al chat, que es una forma de comunicarte con otros jugadores cuando estás en modo multijugador. Aunque enviar mensajes es una función estándar en otras versiones de Minecraft, es una función oculta en la versión Rasp-berry Pi. Pero puedes acceder a esta función mediante el poder de la programación. ¡Podrás compartir información secreta con tus amigos y presumir de tus tesoros!

También aprenderás sobre las funciones en este capítulo. Si tienes ojo de lince, te habrás dado cuenta de que ya has visto algunas funciones. setPos(), setTilePos(), setBlock(), setBlocks(), getPos() y getTilePos() son todas ellas *funciones:*bloques de código *reutilizables*que te facilitan la realización de tareas. Muy chulo, ¿eh?

En las misiones de este capítulo, ampliarás los conocimientos que has aprendido hasta ahora. Imprimirás mensajes en el chat de Minecraft utilizando cadenas y practicarás la introducción de datos para crear objetos en tu mundo Minecraft.

### **¿Qué son las cadenas?**

Un tipo de dato *cadena* incluye cualquier cantidad de texto, desde una sola letra o símbolo -como "a" o "&"- hasta un gran bloque de texto. Cada letra, número o símbolo de una cadena se denomina *carácter*. Cuando quieras incluir letras, símbolos, palabras, frases o una combinación de estas cosas en tu programa, utilizarás cadenas.

Con el tipo de datos cadena, puedes almacenar letras, números y símbolos. Todas las cadenas van entre comillas. Por ejemplo, esto es una cadena:

"Look out! There's a zombie behind you!"

Lo siguiente también es una cadena:

'Welcome to my secret base!'

¿Has notado la ligera diferencia en la forma de escribir estos ejemplos? Al escribir una cadena, puedes utilizar comillas simples o dobles: ' o ". ¡Ten cuidado de no mezclar las comillas! Si empiezas una cadena con una comilla simple, debes terminarla con una comilla simple. Si empiezas con comillas dobles, termina con comillas dobles. Hay una buena razón para incluir ambas opciones en el lenguaje de programación Python; por ejemplo, si quieres utilizar un apóstrofo en tu cadena, puedes incluirlo con seguridad si encierras tu cadena entre comillas dobles.

### **La función print()**

Mostrar texto y otra información al usuario es importante para la interacción con él; de lo contrario, no sabrá lo que ocurre en tus programas. La información que muestras al usuario se llama *salida*. Para mostrar datos en la pantalla del usuario, utiliza la función print().

Para mostrar un mensaje, pasa una cadena a la función print( ) como argumento:

>>> print("String")

Esto indica a Python que quieres mostrar al usuario la palabra Cadena. Así, para imprimir chocolate en el shell de Python, escribes:

>>> print("chocolate")

Y la salida sería

chocolate

También puedes utilizar print() para imprimir los valores de las variables. Por ejemplo, si tienes una variable llamada nombre que almacena un nombre como cadena, y quieres mostrarlo en la pantalla, puedes hacer lo siguiente:

>>> name = "Steve the Miner"

Después de almacenar la cadena "Steve el minero" en nombre, puedes escribir simplemente print(nombre) para mostrar esta salida:

>>> print(name)  
Steve the Miner

Ahora que ya conoces los conceptos básicos de las cadenas, ¡completa la misión para saludar a tu mundo Minecraft!

#### **Misión nº 11: Hola, Mundo Minecraft**

Si quieres chatear con otros jugadores en Minecraft Pi, la API Python de Minecraft te permite enviar mensajes al chat utilizando la función postToChat (). La función postToChat () toma una cadena como argumento y publica esa cadena en la ventana de chat de Minecraft. Por ejemplo, [el Listado 4-1](ch04.xhtml#ch4ex1) envía "Hola, Mundo Minecraft" al chat.

*message.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
mc.postToChat("Hello, Minecraft World")

*Listado 4-1: Utiliza Python para enviar un saludo a través del chat de Minecraft.*

Recuerda que un argumento es información que pasas a una función cuando la llamas. La función necesita esta información para hacer su trabajo. Por ejemplo, en el capítulo anterior, necesitábamos pasar números a nuestras funciones para definir lo que queríamos que hicieran. En este caso, postToChat( ) necesita una cadena, como "Hola, Mundo Minecraft".

La función postToChat () es similar a la función print(). Ambas pueden mostrar cadenas en la pantalla, y ambas pueden tomar como argumento una variable que almacene una cadena. La diferencia es que la función print () envía cadenas al shell de Python, mientras que postToChat () muestra la salida en el chat de Minecraft.

Copia el código [del Listado 4-1](ch04.xhtml#ch4ex1) y guárdalo como *mensaje.py* en una nueva carpeta llamada *cadenas*. Cuando ejecutes el programa, deberías ver el mensaje publicado en el chat, como se muestra en la [Figura 4](ch04.xhtml#ch4fig1)-1.



*Figura 4-1: Mi mensaje se ha publicado en el chat.*

Prueba a pasar una cadena diferente a postToChat() para que muestre un mensaje de chat diferente.

**OBJETIVO EXTRA: ¿DÓNDE ESTÁS?**

Puedes publicar todo tipo de información en el chat utilizando la función mc.postToChat(). Prueba a mostrar la posición x actual del jugador o el tipo de bloque sobre el que se encuentra. Recuerda que la función mc. player.getTilePos () obtiene la posición actual del jugador y la función mc.getBlock() te indica el tipo de bloque en determinadas coordenadas.

### **La función input()**

Hasta ahora, todas tus variables se han establecido en tus programas, o se han *codificado*. Para cambiar el valor de una variable, tendrías que editar el programa. Sería práctico poder cambiar estas variables mientras el programa se está ejecutando o aceptar la *entrada* del usuario por parte del jugador.

Una forma de añadir este tipo de interactividad a tu programa es utilizar la función input(). Imprime una cadena en la consola (para indicar al usuario qué tipo de información debe introducir) y luego espera a que el usuario escriba una respuesta. Intenta introducir este código en el intérprete de comandos de Python para ver qué ocurre:

>>> input("What is your name? ")

Verás la cadena que pasaste a input(), y podrás escribir una respuesta.

What is your name?

Cuando introduzcas una respuesta, deberías ver algo así:

What is your name? Craig  
'Craig'

¡Genial! Pero si quieres utilizar esta entrada en algún lugar de tu programa, tendrás que guardarla en una variable. A diferencia del shell de Python, los programas creados en el editor de texto no muestran automáticamente los resultados de las sentencias. Por ejemplo:

>>> name = input("What is your name? ")  
What is your name? Craig

Observa cómo esta vez, después de escribir tu nombre y pulsar INTRO, el programa no muestra automáticamente tu entrada. Para ver la entrada guardada, sólo tienes que pasar el nombre de la variable como argumento a la función print():

>>> print(name)  
Craig

¡Genial! Ahora has guardado tu entrada en una variable y has impreso el valor de la variable. Esto es muy útil porque te permite obtener información del usuario para utilizarla en cualquier parte de tu programa. ¡Utilicemos esta técnica para escribir mensajes de chat en el chat de Minecraft!

#### **Misión nº 12: Escribe tu propio mensaje de chat**

¡Hagamos el chat más interactivo! Puedes utilizar el intérprete de comandos Python para escribir un mensaje en el chat de Minecraft, como hiciste en la Misión nº 11. En esta misión, escribiremos un programa ligeramente diferente que guarde la cadena que quieras publicar en el chat en una variable llamada mensaje.

[El listado 4-2](ch04.xhtml#ch4ex2) te ayudará a empezar. Cópialo en un nuevo archivo en IDLE y guárdalo como *mensajeInput.* py en tu carpeta de *cadenas*.

*messageInput.py*

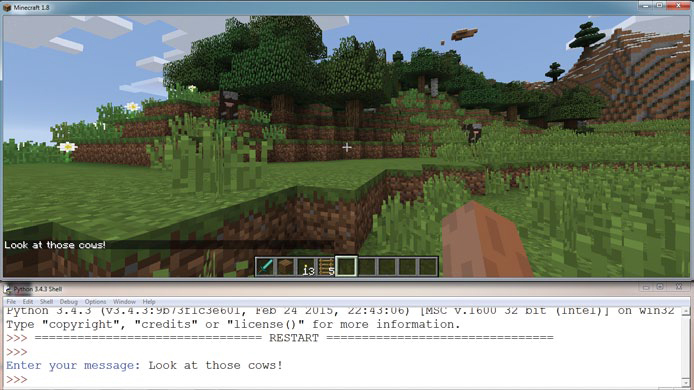
from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
➊ message = "This is the default message."  
➋ mc.postToChat(message)

*Listado 4-2: Cómo enviar cadenas al chat de Minecraft*

El programa almacena el mensaje que quieres enviar al chat en la variable mensaje ➊. En este caso, la variable es una cadena que dice "Este es el mensaje por defecto". Luego el programa pasa el mensaje a la función postToChat( ) ➋, que emite la cadena al chat de Minecraft.

En este programa, la cadena está codificada, lo que significa que será la misma cada vez que se ejecute el programa. Pero con un simple cambio, ¡puedes hacer que imprima lo que el usuario escriba! En otras palabras, puedes escribir tus propios mensajes personalizados cada vez que ejecutes el programa. Crearás tu propio programa de chat.

Para que el programa acepte entradas, sustituye la cadena " Este es el mensaje por defecto". ➊ por la función input( ). Dale a la función input( ) un argumento, como "Introduce tu mensaje: ". Recuerda poner esta cadena dentro de los paréntesis de la función input( ). Cuando hayas hecho los cambios en el programa, ejecútalo. Deberías ver un mensaje en el intérprete de comandos de Python que diga "Introduce tu mensaje : ". Escribe tu mensaje y pulsa INTRO. El mensaje aparecerá en el intérprete de comandos y en el chat de Minecraft, como se muestra en la [Figura 4-2](ch04.xhtml#ch4fig2).



*Figura 4-2: Cuando introduzco un mensaje en el intérprete de comandos IDLE, se publica en el chat de Minecraft.*

Ahora tu programa te permite escribir un mensaje para mostrarlo en el chat, en lugar de tener que escribir tú el mensaje en tu programa. ¿Ves cuánto más fácil es chatear utilizando la entrada?

**OBJETIVO EXTRA: MÁS MENSAJES**

El programa pide un mensaje, pero ¿puedes averiguar cómo hacer que pida un mensaje, espere unos segundos (utilizando la función sleep() ), y luego pida un segundo mensaje?

### **Unir cadenas**

A menudo, necesitarás imprimir una combinación de cadenas. Esto se llama unir, o *concatenar*, cadenas, y Python lo hace fácil.

En el [Capítulo 3](ch03.xhtml#ch03) utilizamos el operador de suma(+) para sumar números, pero también puedes utilizarlo para concatenar cadenas. Por ejemplo:

firstName = "Charles"  
lastName = "Christopher"  
print(firstName + lastName)

La salida de print( ) será "CarlosCristóbal". Si quieres un carácter de espacio entre los valores, puedes añadir un espacio utilizando el operador de suma de esta forma:

print(firstName + " " + lastName)

Python suele ofrecer varias formas de conseguir el mismo resultado. En este caso, podrías utilizar una coma para crear el espacio:

print(firstName, lastName)

Ambas sentencias darán como resultado "Charles Christopher". También puedes concatenar cadenas codificadas con variables que sean cadenas. Sólo tienes que escribir el valor como escribirías cualquier otra cadena:

print("His name is " + firstName + " " + lastName)

Esto dará como resultado "Se llama Carlos Cristóbal".

Unir bloques de texto es útil, pero a veces querrás unir cadenas con otro tipo de datos, como un entero. Python no te permitirá concatenar una cadena con un entero; en este caso, tienes que decirle a Python que primero convierta el entero en una cadena. Vamos a intentarlo.

#### **Convertir números en cadenas**

Convertir un tipo de variable en otro es muy práctico. Por ejemplo, imagina que almacenas el número de manzanas de oro que tienes, que es un número entero, en una variable llamada misManzanasDeOro. Quieres presumir ante tus amigos de cuántas manzanas doradas tienes, porque son raras y te gusta presumir. Podrías imprimir un mensaje como " Mi alijo no tan secreto de manzanas doradas: ", seguido del valor almacenado en misManzanasDoradas. Pero antes de poder incluir el valor de misManzanasDoradas en el mensaje impreso, tienes que decirle a Python que cambie el número entero de misManzanasDoradas por una cadena.

La función str( ) convierte tipos de datos que no son cadenas, como enteros y flotantes, en cadenas. Para convertir a cadena, pon el valor que quieres convertir dentro de los paréntesis de la función str( ).

Volvamos a tu almacén de manzanas de oro. Digamos que has puesto misManzanasDoradas a 2, y quieres que Python trate ese 2 como una cadena en lugar de como un entero. Así es como imprimirías tu mensaje:

print("My not-so-secret golden apple stash: " + str(myGoldenApples))

Esta sentencia muestra la cadena "Mi alijo no tan secreto de manzanas de oro: 2". También puedes convertir valores flotantes en cadenas. Supongamos que te has comido media manzana golden, y ahora misManzanasGolden almacenan 1,5 manzanas. str(misManzanasGolden) funciona igual con el 1,5 que con el 2. Convierte el 1,5 en una cadena para que puedas incluirlo en tu mensaje.

Una vez convertidos los números en cadenas, puedes concatenarlos como quieras. ¡Vamos a divertirnos convirtiendo números en cadenas y concatenándolas!

#### **Concatenar números enteros y flotantes**

Si quieres concatenar dos datos, deben ser cadenas. Pero el signo más se utiliza tanto para sumar como para concatenar, así que si concatenas enteros, flotantes y otros números, Python intentará sumarlos en su lugar. Debes cambiar los valores numéricos a cadenas para poder unirlos utilizando la concatenación.

Para unir dos números en lugar de sumarlos, sólo tienes que utilizar el método str():

print(str(19) + str(84))

Como le has dicho a Python que trate los números 19 y 84 como cadenas y los concatene, esta sentencia produce 1984 en lugar de 103, la suma de 19 y 84.

Puedes utilizar la concatenación tantas veces como quieras dentro de una sentencia. Por ejemplo:

print("The year is " + str(19) + str(84))

Esta línea de código da como resultado El año es 1984.

Ahora que ya tienes un poco de práctica en el uso de la concatenación, ¡pon a prueba tus nuevas habilidades en la siguiente misión!

#### **Misión nº 13: Añadir nombres de usuario al chat**

Cuando juegas una partida con más de dos personas, puede ser confuso averiguar quién está escribiendo un mensaje en el chat de Minecraft. La solución obvia es incluir el nombre del usuario al principio de su mensaje. En esta misión, modificarás el programa de la Misión nº 12 para incluir un nombre de usuario en todos los mensajes enviados al chat.

Abre *messageInput.* py en IDLE y guárdalo como un nuevo archivo llamado *userChat.py* en la carpeta *strings*. A continuación, añade código para introducir el nombre del usuario antes de introducir su mensaje. El mensaje enviado al chat debe tener en el siguiente formato: "Anna: Necesito TNT". Tendrás que utilizar la concatenación para cumplir esta misión.

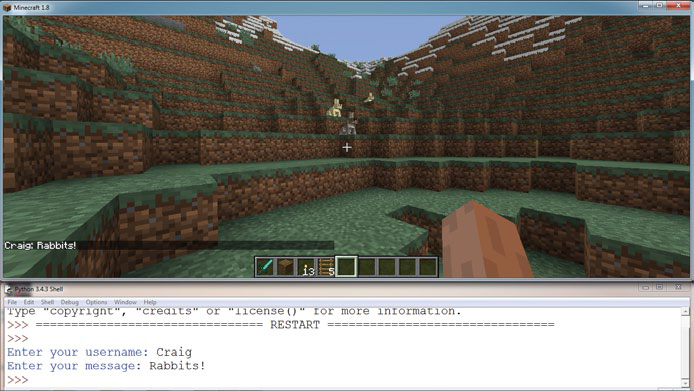
En el programa, busca esta línea de código:

message = input("Enter your message: ")

En la línea anterior, tienes que añadir otra variable llamada nombre de usuario y establecer su valor en input("Por favor, introduce un nombre de usuario: "). Después de añadir la variable nombredeusuario, busca esta línea:

mc.postToChat(message)

Usando la concatenación, une las cadenas nombredeusuario y mensaje dentro de la función postToChat(). Añade ": " entre las dos cadenas para que la salida tenga dos puntos y un espacio entre la variable nombre de usuario y la variable mensaje. [La Figura 4-3](ch04.xhtml#ch4fig3) muestra el aspecto que debería tener la salida del programa terminado.



*Figura 4-3: Ahora, cuando publico en el chat utilizando mi programa, aparece mi nombre de usuario.*

Guarda tu programa actualizado y ejecútalo. En el intérprete de comandos de Python se te pedirá que introduzcas un nombre de usuario. Escribe tu nombre y pulsa INTRO. A continuación se te pedirá que escribas un mensaje, así que hazlo también. Tu nombre de usuario y tu mensaje deberían aparecer en el chat de Minecraft.

**OBJETIVO EXTRA: UN USUARIO SIN NOMBRE**

¿Qué ocurre si dejas el nombre de usuario en blanco y pulsas ENTER? ¿A qué crees que se debe?

### **Convertir cadenas en enteros con int()**

Al igual que la función str (), que convierte tipos de datos que no son cadenas en cadenas, la función int( ) convierte tipos de datos que no son enteros en enteros.

La función int () es útil cuando se utiliza con la función input (). La función input () devuelve la entrada del usuario como una cadena, pero a menudo querrás utilizar esta entrada en operaciones matemáticas. Para ello, primero tendrás que convertir la entrada a un tipo entero utilizando int( ).

Funciona así. Supongamos que ya hemos asignado un valor entero a una variable llamada latasDeAtúnPorGato, y queremos un programa que nos diga cuánto atún se come, en función del número de gatos que tenga el usuario. Aquí tienes un ejemplo de programa que podríamos escribir:

cansOfTunaPerCat = 4  
cats = input("How many cats do you have? ")  
cats = int(cats)  
dailyTunaEaten = cats \* cansOfTunaPerCat

Puedes hacer lo mismo en una sola línea poniendo una función dentro de la otra:

cats = int(input("How many cats do you have? "))  
dailyTunaEaten = cats \* cansOfTunaPerCat

Ahora que sabes cómo convertir la entrada en un número entero, puedes utilizarlo para introducir tipos de bloque en los programas de Minecraft.

#### **Misión nº 14: Crear un bloque con entrada**

Hay montones de tipos de bloques en Minecraft. Aunque puedes elegir muchos bloques en el modo creativo, muchos otros no se pueden utilizar. Sin embargo, la API de Python para Minecraft te da acceso a todos los tipos de bloque y te permite establecerlos mediante la función setBlocks().

Ya has utilizado antes la función setBlocks (), pero tenías que codificar el tipo de bloque en tu programa. Esto significa que no podías cambiarlo mientras se ejecutaba el programa. Ahora puedes utilizar la función input(). Si escribes un programa que acepte entradas, cada vez que ejecutes el programa podrás elegir el tipo de bloque que quieres crear. Podrías crear un bloque de lana la primera vez que ejecutes el programa, y luego crear mineral de hierro la segunda vez.

En esta misión, escribirás un programa que permita al usuario decidir qué tipo de bloque quiere crear. Copia [el Listado 4-3](ch04.xhtml#ch4ex3) en un nuevo archivo y guárdalo como *blockInput.py* en tu carpeta de *cadenas*.

*blockInput.py*

from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
➊ blockType = # Add input() function here  
pos = mc.player.getTilePos()  
x = pos.x  
y = pos.y  
z = pos.z  
mc.setBlock(x, y, z, blockType)

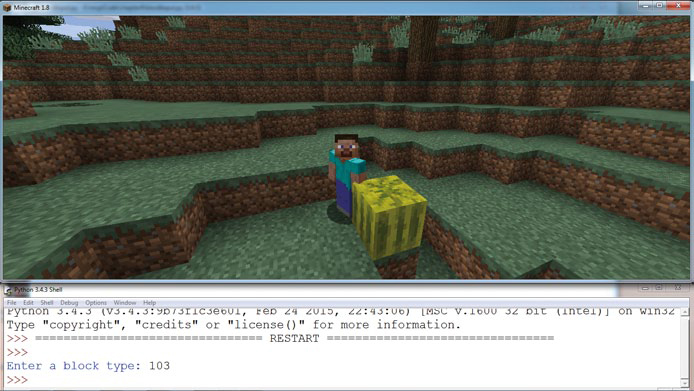
*Listado 4-3: Código para establecer un bloque en la posición del jugador*

Este programa establece un bloque en la posición actual del jugador. Cámbialo para que la variable blockType se establezca mediante la función input () ➊. Te sugiero que incluyas una pregunta u otro aviso de texto para que el usuario sepa que debe escribir un número de bloque, y no otro tipo de entrada. Si no incluyes una pregunta, IDLE se limitará a esperar en una línea en blanco hasta que el usuario introduzca algo, y tú quieres dejar claro que el programa necesita un número del usuario.

Recuerda que input() devuelve la entrada como una cadena, y para que introduzca el valor como un número entero, tienes que utilizar la función int(). La expresión para recoger la entrada para el tipo de bloque debería tener este aspecto:

blockType = int(input("Enter a block type: "))

Guarda tu programa modificado, ejecútalo e introduce el número de bloque que quieras. [La Figura 4-4](ch04.xhtml#ch4fig4) muestra el resultado del programa.



*Figura 4-4: ¡Ahora puedo crear el bloque que quiera!*

**OBJETIVO EXTRA: JUEGOS MÁS INTERACTIVOS**

Puedes utilizar tantas entradas en el programa como quieras. De momento, el programa crea el bloque en la posición actual del jugador. Piensa cómo establecerías las variables x, y y z utilizando entradas. Si te sientes realmente aventurero, intenta teletransportar al jugador a unas coordenadas concretas utilizando entradas.

### **Recuperarse de los errores**

Python utiliza el manejo de *excepciones* para asegurarse de que tu programa pueda recuperarse de los errores y continuar ejecutándose cuando se produzcan. Por ejemplo, el manejo de excepciones es una forma de gestionar las entradas incorrectas del usuario.

Supongamos que tu programa solicita un número entero, pero el usuario introduce una cadena. Normalmente, el programa mostraría un mensaje de error, lo que también se llama *lanzar una* excepción, y dejaría de ejecutarse.

Con el manejo de excepciones, puedes gestionar tú mismo ese error: puedes mantener el programa funcionando sin problemas, mostrar un mensaje de error útil al usuario -como "Por favor, introduzca un número en su lugar"- y darle la oportunidad de solucionar su problema sin reiniciar el programa.

Una sentencia try-except es una herramienta que puedes utilizar para gestionar errores. Es especialmente buena para proporcionar información útil al usuario cuando introduce datos incorrectos, y puede evitar que tu programa se detenga cuando se produce un error.

La sentencia se compone de dos partes: el try y el except. La primera parte, try, es el código que quieres ejecutar si no se produce ningún error. Este código puede tomar una entrada o imprimir una cadena. La parte except de la sentencia se ejecutará sólo si se produce un error en la parte try.

Imagina un código que te pregunta cuántos pares de gafas de sol tienes (tengo tres pares):

try:  
➊ noOfSunglasses = int(input("How many sunglasses do you own? "))  
except:  
➋ print("Invalid input: please enter a number")

Este programa requiere un número. Si introduces letras o símbolos, imprimirá "Entrada no válida: introduce un número". El error se produce porque la función int( ) sólo puede convertir cadenas que contengan sólo números enteros ➊. Si introduces un número, el código funcionará, pero si introduces algo que no es un número, como muchas gafas de sol, esta entrada provocará un error en la función int( ).

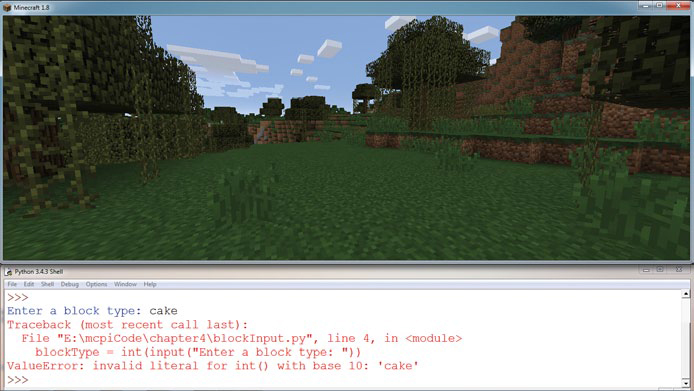
Por cierto, ¿has notado algo diferente en este trozo de código? Es la primera vez que utilizamos una sentencia que requiere *sangría*, que es cuando escribes varios espacios *antes de* escribir cualquier texto. Hablaré más de la sangría cuando trate las sentencias if en [el Capítulo 6](ch06.xhtml#ch06) y los bucles for en [los Capítulos 7](ch07.xhtml#ch07) y [9](ch09.xhtml#ch09). Por ahora, asegúrate de que escribes tu código exactamente como aparece en este libro.

Normalmente, cuando se produce un error, Python muestra un mensaje difícil de entender y no indica claramente al usuario cómo solucionar el problema. Pero con try-except, puedes evitar que los mensajes de error de Python se muestren al usuario cuando introduzca un tipo de datos incorrecto; en su lugar, puedes darle instrucciones sencillas y útiles sobre lo que debe hacer. A veces el usuario simplemente pulsa INTRO en lugar de introducir datos. Normalmente, esto crea un error, pero con el código dentro de la sentencia try-except ➊, en su lugar se imprimirá la cadena " Entrada no válida: por favor, introduzca un número ➋.

Puedes poner casi cualquier código dentro de una sentencia try-except, incluso otras sentencias try-except. ¡Pruébalo en la siguiente misión!

#### **Misión nº 15: Sólo se permiten números**

¿Recuerdas el programa que escribiste en la Misión 14? Cuando introducías un valor entero, el programa funcionaba exactamente como debía y creaba un bloque. Pero si introducías una cadena, el programa dejaba de funcionar y mostraba un error, como se muestra en la [Figura 4-5](ch04.xhtml#ch4fig5).



*Figura4-5:* cake *no es un número, por lo que el programa no creó un bloque.*

Este mensaje de error tiene sentido cuando estás acostumbrado a Python. Pero, ¿qué pasaría si alguien que nunca antes hubiera utilizado Python intentara introducir una cadena en lugar de un número entero? Recibiría un mensaje de error que no entendería. Tu misión es utilizar el tratamiento de errores para escribir un mensaje que sea fácil de entender.

Abre el programa bloqueEntrada *.py* que creaste en la Misión 14. Guarda el programa como *blockInputFix.py* en la carpeta *strings*.

Vas a cambiar el programa para que utilice una sentencia try-except cuando te pida un número de bloque. Busca la última línea de código en el programa. Debe tener este aspecto:

mc.setBlock(x, y, z, blockType)

Añade una sentencia try en la línea anterior a ésta y añade cuatro espacios al principio de la línea antes de la función mc.setBlock(). Después, en la línea anterior a setBlock( ), añade este código para obtener una entrada del usuario: blockType = int(input("Introduce un tipo de bloque: ")).

A continuación, en la línea justo después de la función setBlock (), escribe una sentencia except. Dentro de la sentencia except añade una línea de código que publique un mensaje en el chat de Minecraft para decir que el tipo de bloque debe ser un número; por ejemplo, " ¡No has introducido un número! Introduce un número la próxima vez". Este es el aspecto que debería tener el código modificado (fíjate en los cuatro espacios, o sangría, al principio de las líneas ➊ y ➋):

try:  
➊ blockType = int(input("Enter a block type: "))  
mc.setBlock(x, y, z, blockType)  
except:  
➋ mc.postToChat("You did not enter a number! Enter a number next time.")

La función int( ) espera convertir la entrada introducida por el usuario en un número entero ➊. Como hemos añadido la sentencia try-except al programa, si el usuario introduce algo que no es un número (como letras o símbolos), se producirá un error. En lugar de mostrar el mensaje de error normal de Python, el programa enviará una cadena al chat pidiendo al usuario que introduzca sólo un número ➋. Quizá quieras cambiar el mensaje del chat para que sea un poco más educado.

Cuando hayas terminado de introducir un mensaje de error más amable, guarda el archivo *blockInputFix.py* y ejecútalo para admirar tu obra. El resultado debería parecerse a [la Figura 4-6](ch04.xhtml#ch4fig6).



*Figura 4-6: El mensaje de error mostrado en el chat es mucho más fácil de entender.*

#### **Misión nº 16: Sprint Record**

La misión final de este capítulo combina todo lo que has aprendido sobre variables[(Capítulo 2](ch02.xhtml#ch02)) y operadores matemáticos[(Capítulo 3](ch03.xhtml#ch03)) con la publicación de mensajes en el chat. Tu tarea consiste en crear un registrador: el programa calculará la distancia que recorre el jugador en 10 segundos y mostrará los resultados en el chat.

Recuerda que puedes utilizar el siguiente código para hacer que tus programas esperen, o duerman, un determinado número de segundos:

import time # Place this somewhere near the top of your program  
time.sleep(30) # Makes the program wait 30 seconds

Utiliza este ejemplo de sleep() y escribe el siguiente código para empezar con este nuevo programa:

*sprint.py*

import time  
from mcpi.minecraft import Minecraft  
mc = Minecraft.create()  
  
➊ pos1 = mc.player.getTilePos()  
x1 = pos1.x  
y1 = pos1.y  
z1 = pos1.z  
  
time.sleep(10)  
  
➋ pos2 = mc.player.getTilePos()  
x2 = pos2.x  
y2 = pos2.y  
z2 = pos2.z  
  
# Compare the difference between the starting position and ending position  
➌ xDistance = x2 – x1  
yDistance =  
zDistance =  
  
# Post the results to the chat  
➍ mc.postToChat("")

Desglosemos este código. El programa obtiene la posición inicial del jugador ➊, espera 10 segundos y luego obtiene la posición final del jugador ➋. Para terminar el programa, tienes que calcular la diferencia entre las posiciones inicial y final. Para ello, establece los valores de las variables yDistancia y zDistancia, que comienzan en ➌. Para ayudarte, he incluido el valor de la variable xDistancia, que debe ser x2 - x1. Los valores de las variables yDistancia y zDistancia deben ser similares a éste, pero utilizando variables distintas en lugar de x1 y x2.

En la última línea, envía los resultados al chat de Minecraft ➍. Los resultados deben tener el siguiente formato "El jugador se ha movido x: 10, y: 6, y z: -3". Utiliza cadenas, concatenación y los valores de las variables xDistancia, yDistancia y zDistancia para hacerlo.

Guarda este programa en la carpeta *cadenas* como *sprint.py* y ejecútalo. [La Figura 4-7](ch04.xhtml#ch4fig7) muestra el resultado del programa.



*Figura 4-7: Al finalizar el programa se muestra la distancia recorrida.*

Si tienes el programa en marcha pero te resulta difícil cambiar entre la línea de comandos y Minecraft lo suficientemente rápido, prueba a añadir una cuenta atrás de tres segundos antes del paso 2. Publica esta cuenta atrás en el chat.

**OBJETIVO EXTRA: A VUELO DE PÁJARO**

En este punto, el programa muestra la distancia recorrida a lo largo de cada eje por separado: la dirección x, la dirección y y la dirección z. ¿Cómo crearías y mostrarías la distancia recorrida como un único número entero, es decir, a vuelo de pájaro? Pista: Tendrás que utilizar la trigonometría, concretamente el teorema de Pitágoras. Si no estás seguro de cómo hacerlo ahora, no te preocupes; verás un programa similar con código para calcular la distancia recorrida como un único número entero en la Misión 40[(página 145](ch07.xhtml#page_145)).

### **Lo que has aprendido**

¡Enhorabuena! Has aprendido mucho en este capítulo. Has creado cadenas, has mostrado cadenas mediante sentencias print y las has unido mediante concatenación. Has escrito programas que aceptan entradas del usuario, han cambiado los tipos de datos de los valores y han manejado excepciones. Por el camino, aplicaste tus conocimientos de Python para hacer más animado tu chat de Minecraft.

En [el Capítulo 5](ch05.xhtml#ch05), aprenderás a controlar el flujo del programa y a indicar a tus programas cómo tomar decisiones.