Desconocido

## 5 Formular preguntas con if y else



En programación, a menudo hacemos preguntas de sí o no, y hacemos algo en función de la respuesta. Por ejemplo, podemos preguntar: "¿Tienes más de 20 años?" y, si la respuesta es afirmativa, responder "¡Eres demasiado mayor!". Este tipo de preguntas se denominan *condiciones* , y combinamos las condiciones y sus respuestas en afirmaciones if . Las condiciones pueden ser más complicadas que una sola pregunta, y las sentencias if pueden combinarse con varias preguntas y diferentes respuestas en función de la respuesta a cada pregunta. En este capítulo, aprenderás a utilizar las sentencias if para construir programas.

### Sentencias If

Podríamos escribir una sentencia if en Python de la siguiente manera:

>>> age = 13  
>>> if age > 20:  
 print('You are too old!')

Una sentencia if está formada por la palabra clave if , seguida de una condición y dos puntos ( : ), como en esta sentencia if age > 20: . Las líneas que siguen a los dos puntos deben estar en un bloque; si la respuesta a la pregunta es sí (o True ), se ejecutarán los comandos del bloque. Ahora, vamos a explorar cómo escribir bloques y condiciones.



**NOTA**

*True*  *es un* *valor* booleano *, llamado así por el matemático George Boole. Los booleanos sólo pueden tener uno de estos dos valores: True o False .*

### Un bloque es un grupo de sentencias de programación

Un *bloque* de código es un conjunto agrupado de sentencias de programación. Por ejemplo, cuando if age > 20: es True , puede que quieras hacer algo más que imprimir "¡Eres demasiado viejo!". Tal vez quieras imprimir más sentencias, como ésta:

>>> age = 25  
>>> if age > 20:  
 print('You are too old!')  
 print('Why are you here?')  
 print('Why aren\'t you mowing a lawn or sorting papers?')

Este bloque de código está formado por tres llamadas a print que se ejecutan sólo si la condición age > 20 resulta ser True . Cada línea del bloque tiene cuatro espacios al principio. Veamos de nuevo ese código, con espacios visibles:

>>> age = 25  
>>> if age > 20:  
 .... print('You are too old!')  
 .... print('Why are you here?')  
 .... print('Why aren\'t you mowing a lawn or sorting papers?')

En Python, los espacios en *blanco -como* un tabulador (que se inserta al pulsar la tecla TAB) o un espacio (que se inserta al pulsar la barra espaciadora)- tienen significado. El código que está en la misma posición (o sangrado el mismo número de espacios desde el margen izquierdo) se agrupa en un bloque. Siempre que empieces una nueva línea con más espacios que la anterior, estarás iniciando un nuevo bloque. Este nuevo bloque también forma parte del bloque anterior, como en la Figura 5-1 [.](ch05.xhtml#ch05fig01)

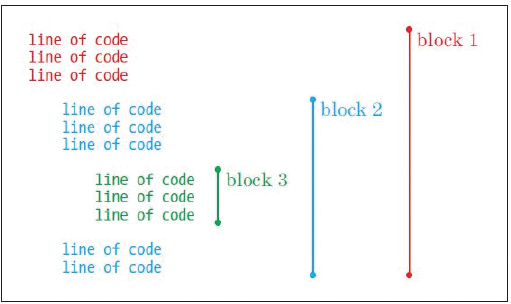
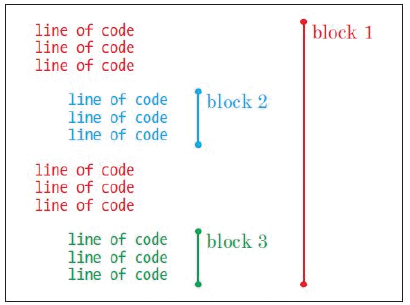


Figura 5-1*: Funcionamiento de los bloques de código*

Agrupamos las sentencias en bloques porque están relacionadas y deben ejecutarse juntas. Cuando cambias la sangría del código, generalmente estás creando nuevos bloques. La Figura [5-2](ch05.xhtml#ch05fig02) muestra cómo se crean tres bloques con sólo cambiar la sangría.



Figura*5-2: Un segundo ejemplo que muestra cómo funcionan los bloques de* código

Aquí, aunque los bloques 2 y 3 tienen la misma sangría, se consideran bloques diferentes porque hay un bloque con menos sangría (menos espacios) entre ellos.

Un bloque con cuatro espacios en una línea y seis espacios en la siguiente producirá un *error de indentación* cuando lo ejecutes. Esto se debe a que Python espera que utilices el mismo número de espacios para todas las líneas de un bloque. Aquí tienes un ejemplo:

>>> if age > 20:  
 .... print('You are too old!')  
 ...... print('Why are you here?')

He hecho visibles los espacios para que puedas ver las diferencias. Observa que la segunda línea print tiene seis espacios en lugar de cuatro. Cuando intentamos ejecutar este código, Python resalta la línea problemática con un bloque rojo y muestra un mensaje explicativo SyntaxError :

>>> age = 25  
>>> if age > 20:  
 print('You are too old!')  
 print('Why are you here?')  
SyntaxError: unexpected indent

Python no esperaba ver dos espacios de más al principio de la segunda línea print .

**NOTA**

*Utiliza un espaciado coherente para que tu código sea más fácil de leer. Si empiezas a escribir un programa y pones cuatro espacios al principio de un bloque, sigue utilizando cuatro espacios al principio de los demás bloques de tu programa. Asegúrate de sangrar cada línea de un bloque con el mismo número de espacios.*

### Las condiciones nos ayudan a comparar cosas

Una *condición* es una expresión de programación que compara cosas y nos dice si los criterios establecidos por la comparación son True (sí) o False (no). Por ejemplo, age > 10 es una condición que pregunta: "¿Es el valor de la variable age mayor que 10?". Otra condición podría ser hair\_color == ’mauve’ , o, "¿Es el valor de la variable hair\_color malva?".

En Python utilizamos símbolos -llamados *operadores-* para crear nuestras condiciones, como *igual a* , *mayor que* y *menor que* . En [la](ch05.xhtml#ch05tab01) Tabla 5-1 se enumeran los operadores más comunes.

**Tabla 5-1:** Símbolos para condiciones

| **Símbolo** | **Definición** |
| --- | --- |
| == | Igual a |
| != | No igual a |
| > | Mayor que |
| < | Menor que |
| >= | Mayor o igual que |
| <= | Menor o igual que |

Por ejemplo, si tienes 10 años, la condición your\_age == 10 devolvería True ; en caso contrario, devolvería False . Si tienes 12 años, la condición your\_age > 10 devolvería True .

**NOTA**

*Asegúrate de utilizar un doble signo igual ( == ) al definir una condición de igual a.*

Probemos algunos ejemplos. Aquí, establecemos nuestra edad como igual a 10 y, a continuación, escribimos una sentencia condicional que imprimirá "¡Eres demasiado mayor para mis chistes!" si age es mayor que 10:

>>> age = 10  
>>> if age > 10:  
 print('You are too old for my jokes!')

¿Qué ocurre cuando escribimos esto en el intérprete de comandos de Python y pulsamos ENTER?

Nada.

Como el valor devuelto por age no es mayor que 10, Python no ejecuta el bloque print . Sin embargo, si hubiéramos puesto age a 20, se imprimiría el mensaje.

Ahora cambiemos el ejemplo anterior para utilizar una condición mayor que o igual a ( >= ):



>>> age = 10  
>>> if age >= 10:  
 print('You are too old for my jokes!')

Deberías ver impreso en pantalla "¡Eres demasiado viejo para mis chistes!" porque el valor de age es igual a 10.

A continuación, probemos a utilizar una condición igual a ( == ):

>>> age = 10  
>>> if age == 10:  
 print("What's brown and sticky? A stick!!")

Deberías ver el mensaje "¿Qué es marrón y pegajoso? Un palo!" impreso en la pantalla.

### Sentencias If-Then-Else

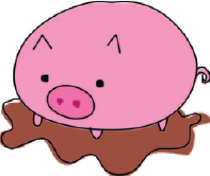
Además de utilizar las sentencias if para hacer algo cuando se cumple una condición ( True ), también podemos utilizar las sentencias if para hacer algo cuando una condición no es cierta. Por ejemplo, podríamos imprimir un mensaje en la pantalla si tu edad es 12 y otro si no lo es.

El truco aquí es utilizar una sentencia if-then-else, que esencialmente dice: "Si algo es cierto, entonces haz esto; si no, haz aquello".

Creemos una sentencia if-then-else. Introduce lo siguiente en el intérprete de comandos de Python:

>>> print('Want to hear a dirty joke?')  
Want to hear a dirty joke?  
>>> age = 12  
>>> if age == 12:  
 print('A pig fell in the mud!')  
 else:  
 print("Shh. It's a secret.")  
  
A pig fell in the mud!

Como hemos puesto age a 12, y la condición pregunta si age es igual a 12, deberías ver el primer mensaje print en la pantalla. Ahora prueba a cambiar el valor de age por un número distinto de 12, así:



>>> print('Want to hear a dirty joke?')  
Want to hear a dirty joke?  
>>> age = 8  
>>> if age == 12:  
 print('A pig fell in the mud!')  
 else:  
 print("Shh. It's a secret.")  
  
Shh. It's a secret.

Esta vez, deberías ver el segundo mensaje print .

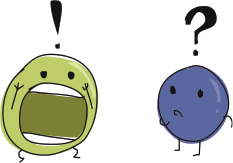
### SENTENCIAS IF Y ELIF

Podemos ampliar aún más una sentencia if con elif , que es la abreviatura de *else-if .* Estas sentencias difieren de las if-then-else en que puede haber más de una elif en la misma sentencia. Por ejemplo, podemos comprobar si la edad de una persona es 10, 11 o 12 (y así sucesivamente) y hacer que nuestro programa haga algo diferente en función de la respuesta:

>>> age = 12  
>>> if age == 10:  
 print('What do you call an unhappy cranberry?')  
 print('A blueberry!')  
 elif age == 11:  
 print('What did the green grape say to the blue grape?')  
 print('Breathe! Breathe!')  
 elif age == 12:  
 print('What did 0 say to 8?')  
 print('Hi guys!')  
 elif age == 13:  
 print("Why wasn't 10 afraid of 7?")  
 print('Because rather than eating 9, 7 8 pi.')  
 else:  
 print('Huh?')  
  
What did 0 say to 8?  
Hi guys!

En este ejemplo, la sentencia if de la segunda línea comprueba si el valor de age es igual a 10. Si es así, se ejecuta la función print que viene a continuación. Sin embargo, como hemos puesto age igual a 12, el ordenador salta a la siguiente parte de la sentencia if (la primera elif o *else if* ) y comprueba si el valor de age es igual a 11. No lo es, por lo que el ordenador ejecuta la función . No lo es, así que el ordenador salta a la siguiente elif para ver si age es igual a 12. Lo es, así que esta vez el ordenador ejecuta la siguiente llamada a print .

Cuando introduzcas este código en IDLE, se sangrará automáticamente, así que asegúrate de pulsar la tecla RETROCESO (o la tecla SUPR si utilizas un Mac) una vez que hayas escrito cada sentencia print . De este modo, tus frases if , elif y else comenzarán en el margen izquierdo. Esta es la misma posición en la que estaría la sentencia if , si no existiera el prompt ( >>> ).



### Combinar condiciones

Puedes combinar condiciones utilizando las palabras clave and y or , lo que produce un código más corto y sencillo. Aquí tienes un ejemplo utilizando or :

>>> if age == 10 or age == 11 or age == 12 or age == 13:  
 print('What is 13 + 49 + 84 + 155 + 97? A headache!')  
 else:  
 print('Huh?')

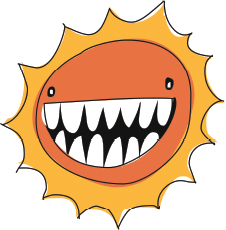
En este código, si alguna de las condiciones de la primera línea es verdadera (si age es 10, 11, 12 ó 13), se ejecutará la sentencia print de la línea siguiente.

Si las condiciones de la primera línea no son verdaderas ( else ), Python muestra Huh? en la pantalla.

Para reducir aún más este ejemplo, podríamos utilizar la palabra clave and , junto con el operador mayor que o igual a ( >= ) y el operador menor que o igual a ( <= ), como se indica a continuación:

>>> if age >= 10 and age <= 13:  
 print('What is 13 + 49 + 84 + 155 + 97? A headache!')  
 else:  
 print('Huh?')

Aquí, si age es mayor o igual que 10 y menor o igual que 13, se ejecutará la sentencia print de la línea siguiente. Por ejemplo, si el valor de age es 12, se imprimirá en pantalla What is 13 + 49 + 84 + 155 + 97? A headache! porque 12 es mayor que 10 y menor que 13.



### Variables sin valor-Ninguna

Igual que podemos asignar números, cadenas y listas a una variable, también podemos asignarle nada, o un valor vacío. En Python, un valor vacío se denomina None . Es importante tener en cuenta que el valor None es diferente del valor 0 porque es la *ausencia* de un valor en lugar de un número con valor 0. He aquí un ejemplo en el que asignamos a una variable el valor None :

>>> myval = None  
>>> print(myval)  
None

Asignar None a una variable indica a Python que la variable ya no tiene ningún valor (o mejor dicho, que ya no etiqueta un valor). También es una forma de definir una variable sin establecer su valor. Esto puede ser útil cuando sabes que necesitarás una variable más adelante en tu programa, pero quieres definir todas las variables al principio.

**NOTA**

*Los programadores suelen definir sus variables al principio de un programa (o de una función) para tener una referencia rápida de todas las variables utilizadas en un trozo de código.*

También puedes buscar None en una sentencia if , como en el siguiente ejemplo:

>>> myval = None  
>>> if myval is None:  
 print("The variable myval doesn't have a value")  
  
The variable myval doesn't have a value

Esto es útil cuando quieres calcular el valor de una variable sólo si aún no se ha calculado. En este caso, la comprobación de None indica a Python que realice el cálculo sólo si la variable no tiene ningún valor.

### Diferencia entre cadenas y números

La entrada del*usuario* es lo que una persona introduce en el teclado, ya sea un carácter, una flecha pulsada o la tecla ENTER, o cualquier otra cosa. En Python, la entrada del usuario es una cadena, lo que significa que cuando escribes el número 10 en el teclado, Python guarda el número 10 como una cadena, no como un número.

Compara el número 10 y la cadena '10' . Aunque para nosotros la única diferencia entre ambos es que uno está rodeado de comillas, para un ordenador son completamente distintos.

Por ejemplo, comparemos el valor de la variable age con un número en una sentencia if , así

>>> if age == 10:  
 print("What's the best way to speak to a monster?")  
 print("From as far away as possible!")

Si primero establecemos la variable age con el número 10 :

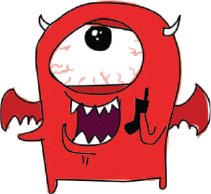
>>> age = 10  
>>> if age == 10:  
 print("What's the best way to speak to a monster?")  
 print("From as far away as possible!")  
   
What's the best way to speak to a monster?  
From as far away as possible!

Como puedes ver, se ejecuta la sentencia print .

A continuación, establecemos age a la cadena ‘10’ (con comillas), así:

>>> age = '10'  
>>> if age == 10:  
 print("What's the best way to speak to a monster?")  
 print("From as far away as possible!")  
  
>>>

Aquí, la sentencia print no se ejecuta porque Python no ve la cadena como un número.



Afortunadamente, Python tiene funciones que pueden convertir cadenas en números y números en cadenas. Por ejemplo, puedes convertir la cadena ‘10’ en un número con la función int :

>>> age = '10'  
>>> converted\_age = int(age)

La variable converted\_age contiene ahora el número 10 (en lugar de una cadena).

Para convertir un número en una cadena, utiliza str del siguiente modo:

>>> age = 10  
>>> converted\_age = str(age)

Ahora converted\_age contiene la cadena ‘10’ en lugar del número 10.

¿Recuerdas la sentencia if age == 10 que no imprimía nada cuando la variable se convertía en una cadena ( age = ’10’ )? Si convertimos primero la variable, obtendremos un resultado totalmente distinto:

>>> age = '10'  
>>> converted\_age = int(age)  
>>> if converted\_age == 10:  
 print("What's the best way to speak to a monster?")  
 print("From as far away as possible!")  
  
What's the best way to speak to a monster?  
From as far away as possible!

Pero debes saber esto: si intentas convertir un número con punto decimal (también llamados *números de coma flotante* , porque el punto puede "moverse" en un número), obtendrás un error porque la función int espera un *número entero* (un número sin punto decimal):

>>> age = '10.5'  
>>> converted\_age = int(age)  
Traceback (most recent call last):  
 File "<pyshell#35>", line 1, in <module>  
 converted\_age = int(age)  
ValueError: invalid literal for int() with base 10: '10.5'

Python envía un ValueError para indicarte que el valor que has intentado utilizar no es apropiado. Para solucionarlo, utiliza float en lugar de int , ya que la función float puede manejar números que no sean enteros:

>>> age = '10.5'  
>>> converted\_age = float(age)  
>>> print(converted\_age)  
10.5

También obtendrás un ValueError si intentas convertir una cadena que no contiene un número en dígitos:

>>> age = 'ten'  
>>> converted\_age = int(age)  
Traceback (most recent call last):  
 File "<pyshell#1>", line 1, in <module>  
 converted\_age = int(age)}  
ValueError: invalid literal for int() with base 10: 'ten'

Como hemos utilizado la palabra ten en lugar del número 10 , Python se echa las manos a la cabeza.

### Lo que has aprendido

En este capítulo, has aprendido a utilizar las sentencias if para crear bloques de código que se ejecutan sólo cuando se cumplen determinadas condiciones. Viste cómo ampliar las sentencias if utilizando elif para que diferentes secciones de código se ejecuten como resultado de diferentes condiciones, y cómo utilizar la palabra clave else para ejecutar código si ninguna de las condiciones resulta ser cierta.

Combinaste condiciones utilizando las palabras clave and y or para comprobar si los números caen dentro de un rango, y cambiaste cadenas por números con las funciones int , str y float . También descubriste que None puede restablecer variables a su estado inicial, vacío.

### Rompecabezas de programación

Prueba los siguientes puzzles utilizando if y condiciones. Puedes descargarte las soluciones en http://python-for-kids.com [*.*](http://python-for-kids.com)

#### #nº 1: ¿Eres rico?

¿Qué crees que hará el siguiente código? Intenta averiguar la respuesta sin escribirla en el intérprete de comandos de Python y luego comprueba tu trabajo.

>>> money = 2000  
>>> if money > 1000:  
 print("I'm rich!!")  
 else:  
 print("I'm not rich!!")  
 print("But I might be later...")

#### #2: ¡Twinkies!

Crea una sentencia if que compruebe si el número de Twink-ies (en la variable twinkies ) es inferior a 100 o superior a 500. Tu programa debe imprimir el mensaje "Demasiado pocos o demasiados" si la condición es True .

#### #3: El número justo

Crea una sentencia if que compruebe si la cantidad de dinero contenida en la variable money está entre 100 y 500 o entre 1.000 y 5.000.

#### #4: Puedo luchar contra esos ninjas

Crea una sentencia if que imprima "Son demasiados" si la variable ninjas contiene un número inferior a 50; imprima "Será una lucha, pero puedo con ellos" si es inferior a 30; e imprima "¡Puedo luchar contra esos ninjas!" si es inferior a 10. Puedes probar tu código con esto

>>> ninjas = 5