Tipos booleanos

En esta lección se tratan los tipos de datos lógicos, también llamados boobleanos.

 Los ejemplos de esta lección muestran el resultado de ejecutar las operaciones en el prompt de IDLE.

Tipos booleanos: True y False

Una variable booleana es una variable que sólo puede tomar dos posibles valores: True (verdadero) o False (falso).

En Python cualquier variable (en general, cualquier objeto) puede considerarse como una variable booleana. En general los elementos nulos o vacíos se consideran False y el resto se consideran True.

Para comprobar si un elemento se considera True o False, se puede convertir a su valor booleano mediante la función bool().

```
>>> bool(0)
False
>>> bool(0.0)
False
>>> bool("")
False
>>> bool(None)
False
>>> bool(())
False
>>> bool([])
False
>>> bool({})
False
```

```
>>> bool(25)
True
>>> bool(-9.5)
True
>>> bool("abc")
True
>>> bool((1, 2, 3))
True
>>> bool([27, "octubre", 1997])
True
>>> bool({27, "octubre", 1997})
True
```

Operadores lógicos

Los **operadores lógicos** son unas operaciones que trabajan con valores booleanos.

- and: "y" lógico. Este operador da como resultado True si y sólo si sus dos operandos son True:

```
>>> True and True
True
>>> True and False
False
>>> False and True
False
>>> False and False
False
```

- or: "o" lógico. Este operador da como resultado True si algún operando es True:

```
>>> True or True
True
>>> True or False
True
>>> False or True
True
>>> False or False
False
```

  Nota: En el lenguaje cotidiano, el "o" se utiliza a menudo en situaciones en las que sólo puede darse una de las dos alternativas. Por ejemplo, en un menú de restaurante se puede elegir "postre o café", pero no las dos cosas (salvo que se pague aparte, claro). En lógica, ese tipo de "o" se denomina "o exclusivo" (xor).

- not: negación. Este operador da como resultado True si y sólo si su argumento es False:

```
>>> not True
False
>>> not False
True
```

Expresiones compuestas

Si no se está acostumbrado a evaluar expresiones lógicas compuestas, se recomienda utilizar paréntesis para asegurar el orden de las operaciones.

Al componer expresiones más complejas hay que tener en cuenta que Python evalúa primero los not, luego los and y por último los or, como puede comprobarse en los ejemplos siguientes:

- El operador not se evalúa antes que el operador and:

```
>>>  not True and False
False
>>> (not True) and False
False
>>> not (True and False)
True
```

- El operador not se evalúa antes que el operador or:

```
>>>  not False or True
True
>>> (not False) or True
```

```
True
>>> not (False or True)
False
```

- El operador and se evalúa antes que el operador or:

```
>>> False and True or True
True
>>> (False and True) or True
True
>>> False and (True or True)
False
```

```
>>> True or True and False
True
>>> (True or True) and False
False
>>> True or (True and False)
True
```

Si en las expresiones lógicas se utilizan valores distintos de True o False, Python utiliza esos valores en vez de True o False.

```
>>> 3 or 4
3
```

Si se quieren mostrar valores booleanos, se puede convertir el resultado a un valor booleano:

```
>>> 3 or 4
3
>>> bool(3 or 4)     # Verdadero porque 3 es diferente de 0
True
```

Nota: No suele ser muy habitual tener que utilizar este tipo de expresiones. En la lección Detalles de Python se comenta con más detalle cómo Python elige la respuesta a estas expresiones.

Comparaciones

Las comparaciones también dan como resultado valores booleanos:

- **>** Mayor que; **<** Menor que;

```
>>> 3 > 2
True
>>> 3 < 2
False
```

- **>=** Mayor o igual que; **<=** Menor o igual que;

```
>>> 2 >= 1 + 1
True
>>> 4 - 2 <= 1
False
```

- **==** Igual que; **!=** Distinto de;

```
>>> 2 == 1 + 1
True
```

```
>>> 6 / 2 != 3
False
```

Es importante señalar que en matemáticas el signo igual se utiliza tanto en las asignaciones como en las comparaciones, mientras que en Python (y en otros muchos lenguajes de programación):

- un signo igual (=) significa asignación, es decir, almacenar un valor en una variable
- mientras que dos signos iguales seguidos (==) significa comparación, es decir, decir si es verdad o mentira que dos expresiones son iguales

Cuando se aprende a programar es habitual confundir una cosa con la otra (el error más frecuente es escribir una sola igualdad en las comparaciones), por lo que se recomienda prestar atención a este detalle.

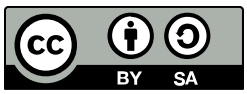
Pyhon permite encadenar varias comparaciones y el resultado será verdadero si y sólo si todas las comparaciones lo son.

```
>>> 4 == 3 + 1 > 2
True
>>> 2 != 1 + 1 > 0
False
```

Encadenar comparaciones no está permitido en otros lenguajes como PHP, en los que las comparaciones deben combinarse mediante operadores lógicos and, lo que también puede hacerse en Python:

```
>>> 4 == 3 + 1  and 3 + 1 > 2
True
>>> 2 != 1 + 1 and 1 + 1 > 0
False
```

Última modificación de esta página: 20 de febrero de 2017



Esta página forma parte del curso **Introducción a la programación con Python** por Bartolomé Sintes Marco
que se distribuye bajo una Licencia Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional (CC BY-SA 4.0).