educación IT
CODE YOUR FUTURE

Introducción a Python

Módulo 1

Operaciones lógicas

Operaciones lógicas

Las operaciones lógicas son tan comunes como las comparaciones en cualquier programa, puesto que nos permiten crear comparaciones más complejas. Consideremos la siguiente variable:

```
>>> a = 5
```

¿Cómo puedo determinar si **a** es mayor a 1 y menor a 10, o, lo que es lo mismo, si a se encuentra entre 1 y 10?

Necesito combinar dos operaciones de comparación; para ello, empleamos la **palabra reservada and** (palabras que Python reserva para sí y que no pueden usarse para constituir el nombre de ninguna variable).

```
>>> a > 1 and a < 10
True
```

Esta operación se conoce como **conjunción**.

El resultado de una conjunción es verdadero cuando las dos condiciones que se propone conectar son verdaderas.

En este caso, `a > 1 and a < 10` es verdadero únicamente cuando a `> 1 y a < 10` son **ambas** verdaderas. Si alguna de ellas es falsa, el resultado también lo es.

Confirmemos lo que acabamos de decir con el ejemplo que vemos a la derecha.

```
>>> True and True
True

>>> True and False
False

>>> False and True
False
```

Por otro lado, la **disyunción** nos permite saber si alguna de dos comparaciones es verdadera. Una **disyunción** es verdadera cuando al menos una de las dos comparaciones que conecta es verdadera.

Si, por ejemplo, queremos saber si `a` es igual a `1` o bien igual a `10`, podemos seguir la segunda referencia a la derecha: aquí, tanto `a == 1` como `a == 10` son comparaciones falsas, puesto que `a` es `5`.

```
>>> True or True
True

>>> True or False
True

>>> False or False
False
```

```
>>> a == 1 or a == 10
False
```

La negación, que se ejecuta con la palabra reservada **not**, invierte el valor lógico de un booleano.

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

Así, por ejemplo, las siguientes comparaciones son equivalentes:

```
>>> pi != 3.14
False
>>> not pi == 3.14
False
```

La negación, que se ejecuta con la palabra reservada **not**, invierte el valor lógico de un booleano.

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

Así, por ejemplo, las siguientes comparaciones son equivalentes:

```
>>> pi != 3.14
False
>>> not pi == 3.14
False
```

Al igual que en las operaciones aritméticas, podemos **agrupar expresiones lógicas usando paréntesis.**

```
>>> a = 5
>>> pi = 3.14
>>> not pi == 3.14 and a == 1 or a == 10
False
>>> not (pi == 3.14 and a == 1 or a == 10)
True
```

Por último, esta **tabla de verdad** resulta útil para comprender cada una de las operaciones que acabamos de describir:

a	b	a and b	a or b	not a	not b
True	True	True	True	False	False
False	True	False	True	True	False
True	False	False	True	False	True
False	False	False	False	True	True

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!