# **Operadores y expresiones**

Sabemos que la programación gira en torno a la información, en torno a los datos, ¿pero con qué propósito?

# **El tipo lógico**

Representa la mínima expresión racional:

* Verdadero (True)
* Falso (False)

Podemos preguntarle a Python algunas igualdades numéricas:

1 + 1 == 3

False

1 + 1 == 2

True

# **Operadores relacionales**

Sirven para comparar dos valores, dependiendo del resultado de la comparación se devolverá:

* Verdadero (True), si es cierta
* Falso (False), si no es cierta

## **Igual que**

3 == 2

False

## **Distinto de**

3 != 2

True

## **Mayor que**

3 > 2

True

## **Menor que**

3 < 2

False

## **Mayor o igual que**

3 >= 4

False

## **Menor o igual que**

3 <= 4

True

También podemos comparar variables:

a = 10

b = 5

a > b

True

b != a

True

a == b\*2

True

Y otros tipos, como cadenas, listas, el resultado de algunas funciones o los propios tipos lógicos:

"Hola" == "Hola"

True

"Hola" != "Hola"

False

c = "Hola"

c[0] == "H"

True

c = "Hola"

c[-1] == "a"

True

l1 = [0,1,2]

l2 = [2,3,4]

l1 == l2

False

len(l1) == len(l2)

True

l1[-1] == l2[0]

True

True == True

True

False == True

False

False != True

True

True > False

True

False > True

False

La representación aritmética de True y False equivale a 1 y 0 respectivamente:

True \* 3

3

False / 5

0.0

True \* False

0

# **Operadores lógicos**

Encontramos 3 operadores especiales para realizar operaciones lógicas. Normalmente se utilizan para agrupar, excluir y negar expresiones. Puede ayudar echar un vistazo a [esta explicación](https://es.wikipedia.org/wiki/Tabla_de_verdad) sobre las tablas de la verdad:

* Not
* And
* Or

## **Not (Negación lógica)**

Niega un valor o expresión lógica:

not True

False

not False

True

not True == False

True

## **And (Conjunción lógica)**

Devuelve verdadero sólo si se cumplen todas las condiciones:

True and True

True

True and False

False

False and True

False

False and False

False

Fa = 45

a > 10 and a < 20

False

c = "Hola Mundo"

len(c) >= 20 and c[0] == "H"

False

## **Or (Disyunción lógica)**

Devuelve verdadero si se cumple como mínimo una condición:

True or True

True

True or False

True

False or True

True

False or False

False

c = "OTRA COSA"

c == "EXIT" or c == "FIN" or c == "SALIR"

False

c = "Lector"

c[0] == "H" or c[0] == "h"

False

# **Expresiones anidadas**

Se pueden solucionar empleando las reglas de precedencia:

1. Primero los paréntesis que indican prioridad.
2. Segundo, las expresiones aritméticas por sus propias reglas.
3. Tercero, las expresiones relacionales.
4. Cuarto, las expresiones lógicas.

a = 10

b = 5

a \* b - 2\*\*b >= 20 and not (a % b) != 0

False

# **Operadores en asignación**

Actúan directamente sobre la variable actual modificando su valor.

## **Suma en asignación**

a = 5

a += 5

a

10

## **Resta en asignación**

a = 5

a -= 10

a

-5

## **Producto en asignación**

a = 5

a \*= 2

a

10

## **División en asignación**

a = 5

a /= 2

a

2.5

## **Módulo en asignación**

a = 5

a %= 2

a

1

## **Potencia en asignación**

a = 5

a \*\*= 3

a

125