Operadores y expresiones

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

Ejercicios « Operadores y expresiones

Ejercicio 1

Realiza un programa que lea 2 números por teclado y determine los siguientes aspectos (es suficiene con mostrar True o False):

* Si los dos números son iguales
* Si los dos números son diferentes
* Si el primero es mayor que el segundo
* Si el segundo es mayor o igual que el primero

Ejercicio 2

Utilizando operadores lógicos, determina si una cadena de texto introducida por el usuario tiene una longitud mayor o igual que 3 y a su vez es menor que 10 (es suficiene con mostrar True o False).

Ejercicio 3

Realiza un programa que cumpla el siguiente algoritmo utilizando siempre que sea posible operadores en asignación:

* Guarda en una variable numero\_magico el valor 12345679 (sin el 8)
* Lee por pantalla otro numero\_usuario, especifica que sea entre 1 y 9
* Multiplica el numero\_usuario por 9 en sí mismo
* Multiplica el numero\_magico por el numero\_usuario en sí mismo
* Finalmente muestra el valor final del numero\_magico por pantalla

# Condiciones

Condicionar permite dividir el flujo de un programa en diferentes caminos.

## Sentencia if (si)

El if se ejecuta siempre que la expresión que comprueba devuelva True:

if True: # equivale a if not False

print("Se cumple la condición")

print("También se muestre este print")

Se cumple la condición

También se muestre este print

Podemos encadenar diferentes If:

a = 5

if a == 2:

print("a vale 2")

if a == 5:

print("a vale 5")

a vale 5

O también anidar If dentro de If:

a = 5

b = 10

if a == 5:

print("a vale",a)

if b == 10:

print("y b vale",b)

a vale 5

y b vale 10

Como condición podemos evaluar múltiples expresiones, siempre que éstas devuelvan True o False:

if a==5 and b == 10:

print("a vale 5 y b vale 10")

a vale 5 y b vale 10

## Sentencia else (sino)

Se encadena a un If para comprobar el caso contrario (en el que no se cumple la condición):

n = 11

if n % 2 == 0:

print(n,"es un número par")

else:

print(n,"es un número impar")

11 es un número impar

## Sentencia elif (sino si)

Se encadena a un if u otro elif para comprobar múltiples condiciones, siempre que las anteriores no se ejecuten:

comando = "OTRA COSA"

if comando == "ENTRAR":

print("Bienvenido al sistema")

elif comando == "SALUDAR":

print("Hola, espero que te lo estés pasando bien aprendiendo Python")

elif comando == "SALIR":

print("Saliendo del sistema...")

else:

print("Este comando no se reconoce")

Este comando no se reconoce

nota = float(input("Introduce una nota: "))

if nota >= 9:

print("Sobresaliente")

elif nota >= 7:

print("Notable")

elif nota >= 6:

print("Bien")

elif nota >= 5:

print("Suficiente")

else:

print("Insuficiente")

Introduce una nota: 10

Sobresaliente

Es posible simular el funcionamiento de elif con if utilizando expresiones condicionales:

nota = float(input("Introduce una nota: "))

if nota >= 9:

print("Sobresaliente")

if nota >= 7 and nota < 9:

print("Notable")

if nota >= 6 and nota < 7:

print("Bien")

if nota >= 5 and nota < 6:

print("Suficiente")

if nota < 5:

print("Insuficiente")

Introduce una nota: 8

Notable

## Instrucción pass

Sirve para como instrucción de paso para utilizar en un bloque de código vacío, no finaliza el código. No tiene ningún efecto pero sirve para crear estructuras pendientes de ser programadas:

if True:

pass