OPERADORES

Operadores de asignación

Anteriormente hemos visto los operadores aritméticos, que usaban dos números para calcular una operación aritmética (como suma o resta) y devolver su resultado. En este caso, los operadores de asignación o assignment operators nos permiten realizar una operación y almacenar su resultado en la variable inicial. Podemos ver como realmente el único operador nuevo es el =. El resto son abreviaciones de otros operadores que habíamos visto con anterioridad. Ponemos un ejemplo con x=7

Operador	Ejemplo	Equivalente
=	x=7	x=7
+=	x+=2	x=x+2 = 7
-=	x-=2	x=x-2 = 5
*=	x*=2	x=x*2 = 14
/=	x/=2	x=x/2 = 3.5
%=	x%=2	x=x%2 = 1
//=	x//=2	x=x//2 = 3
**=	x**=2	x=x**2 = 49
&=	x&=2	x=x&2 = 2
|=	x|=2	x=x|2 = 7
^=	x^=2	x=x^2 = 5
>>=	x>>=2	x=x>>2 = 1
&lt&lt=	x&lt&lt=2	x=x&lt&lt2 = 28

Operadores aritméticos

Los operadores aritméticos o *arithmetic operators* son los más comunes que nos podemos encontrar, y nos permiten realizar operaciones aritméticas sencillas, como pueden ser la suma, resta o exponente. A continuación, condensamos en la siguiente tabla todos ellos con un ejemplo, donde `x=10` y `y=3`.

Operador	Nombre	Ejemplo
+	Suma	x + y = 13
-	Resta	x - y = 7
*	Multiplicación	x * y = 30
/	División	x/y = 3.333
%	Módulo	x%y = 1
**	Exponente	x ** y = 1000
//	Cociente	3

Operadores relacionales

Los operadores relacionales, o también llamados *comparison operators* nos permiten saber la **relación existente entre dos variables**. Se usan para saber si por ejemplo un número es mayor o menor que otro. Dado que estos operadores indican si se cumple o no una operación, el valor que devuelven es `True` o `False`. Veamos un ejemplo con `x=2` e `y=3`

Operador	Nombre	Ejemplo
==	Igual	x == y = False
!=	Distinto	x != y = True
>	Mayor	x > y = False
&lt	Menor	x &lt y = True
>=	Mayor o igual	x >= y = False
&lt=	Menor o igual	x &lt y = True

Operadores lógicos

Los operadores lógicos o logical operators nos permiten trabajar con valores de tipo booleano. Un valor booleano o bool es un tipo que solo puede tomar valores True o False. Por lo tanto, estos operadores nos permiten realizar diferentes operaciones con estos tipos, y su resultado será otro booleano. Por ejemplo, True and True usa el operador and, y su resultado será True.

Operador	Nombre	Ejemplo
and	Devuelve True si ambos elementos son True	True and True = True
or	Devuelve True si al menos un elemento es True	True or False = True
not	Devuelve el contrario, True si es Falso y viceversa	not True = False

Operadores bitwise

Los operadores a nivel de bit o **bitwise operators** son operadores que actúan sobre números enteros pero usando su representación binaria.

Operador	Nombre
|	And bit a bit
&	Or bit a bit
~	Not bit a bit
^	Xor bit a bit
>>	Desplazamiento a la derecha
&lt&lt	Desplazamiento a la izquierda

Para entender los operadores de bit, es importante antes entender cómo se representa un número de manera binaria. Todos estamos acostumbrados a lo que se denomina representación `decimal`. Se llama así porque se usan diez números, del `0` al `9` para representar todos los valores. Nuestro decimal, es también posicional, ya que no es lo mismo un `9` en `19` que en `98`. En el primer caso, el valor es de simplemente `9`, pero en el segundo, en realidad ese 9 vale `90`.

Lo mismo pasa con la representación binaria pero con algunas diferencias. La primera diferencia es que sólo existen dos posibles números, el `0` y el `1`. La segunda diferencia es que a pesar de ser un sistema que también es posicional, los valores no son como en el caso decimal, donde el valor se multiplica por `1` en la primera posición, `10` en la segunda, `100` en la tercera, y así sucesivamente. En el caso binario, los valores son potencias de 2, es decir `1, 2, 4, 8, 16` o lo que es lo mismo `$$2^0, 2^1, 2^2, 2^3, 2^4$$`

Operadores de Identidad

El operador de identidad o *identity operator* `is` nos indica si dos variables hacen referencia al mismo objeto. Esto implica que si dos variables distintas tienen el mismo `id()`, el resultado de aplicar el operador `is` sobre ellas será `True`.

Operador	Nombre
is	Devuelve True si hacen referencia a el mismo objeto
is not	Devuelve False si no hacen referencia a el mismo objeto

Operadores de membresía

Los operadores de membresía o *membership operators* son operadores que nos **permiten saber si un elemento está contenido en una secuencia**. Por ejemplo si un número está contenido en una lista de números.

Operador	Nombre	Ejemplo
in	True si el elemento esta contenido	xxx
not in	False si el elemento no esta contenido	xxx

Operador Walrus

El operador walrus o walrus operator se introdujo con la PEP572 en Python 3.8, y se trata de un operador de asignación con una funcionalidad extra que veremos a continuación. El operador se representa con dos puntos seguidos de un igual :=, lo que tiene cierto parecido a una morsa, siendo : los ojos y = los colmillos, por lo que de ahí viene su nombre (walrus significa morsa en Inglés).

El problema que dicho operador intenta resolver es el siguiente. Podemos simplificar las siguientes tres líneas de código.

```
x = "Python"
print(x)
print(type(x))

# Salida:
# Python
# <class 'str'>
```

En sólo dos líneas. Como podemos ver, el uso de := asigna y devuelve el contenido de la variable.

```
print(x := "Python")
print(type(x))
# Python
# <class 'str'>
```