Python Programming

Módulo 1

Operadores

Operadores aritméticos

Los operadores aritméticos son los más comunes que nos podemos encontrar, y nos permiten realizar operaciones matemáticas sencillas.

Los operadores aritméticos tienen orden de prioridad, primero se respeta el () paréntesis, luego los de mayor prioridad en orden:

- Exponente
- Multiplicación, División, División Entera, Módulo
- Suma, Resta

Operador	Nombre	Ejemplo	Resultado
+	Suma	`5 + 6`	`11`
-	Resta	`10 - 2`	`8`
*	Multiplicación	`3 * 10`	`30`
/	División	`5 / 2`	`2.5`
%	Módulo(Resto)	`9 % 2`	`1`
**	Exponente	`10 ** 2`	`100`
//	División Entera	`9 // 2`	`4`

Operadores asignación

Los operadores de asignación nos permiten realizar una operación y almacenar su resultado en la variable inicial.

Podemos ver cómo realmente el más importante es el = igual. El resto son abreviaciones de otros operadores que habíamos visto con anterioridad.

Los operadores de asignación no son más que atajos para escribir de manera más corta, y asignar su resultado a la variable inicial. El operador += en x+=1 es equivalente a x = x + 1.

NO existe el operador ++ como existe en otros lenguajes.

Operador	Ejemplo si x = 7	Equivalente	Resultado
=	`x = 7`	`x = 7`	7
+=	`x += 2`	`x = x + 2`	9
-=	`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

Operadores relacionales

Los operadores relacionales, o también llamados de comparación 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`.

Operador	Nombre	Ejemplo si x=5	Resultado
==	Igual	`x == 6`	`False`
!=	Distinto	`x != 8`	`True`
>	Mayor	`x > 4`	`True`
<	Menor	`x < 3`	`False`
>=	Mayor o igual	`x >= 20`	`False`
<=	Menor o igual	`x <= 5`	`True`

Operadores lógicos

Los operadores relacionales, o también llamados de comparación 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`.

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

Operadores especiales

Operador identidad

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

```
>>> a = 10
>>> b = 10
>>> a is b
True
```

Esto es debido a que Python reutiliza el mismo objeto que almacena el valor 10 para ambas variables. De hecho, si usamos la función `id()`, podemos ver que el objeto es el mismo.

```
>>> id(a)
2618946775632
>>> id(b)
2618946775632
```

Definido el operador `is`, es trivial definir `is not` porque es exactamente lo contrario.

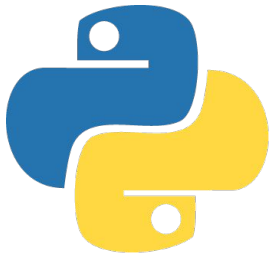
Operador de pertenencia (membresía)

El operador de pertenencia `in` permite saber si un elemento está contenido en una secuencia. Por ejemplo, si un número está contenido en una lista de números. O una key(clave) en un diccionario.

```
>>> datos = [1,2,3]
>>> 3 in datos
True
>>> 10 in datos
False

>>> alumnos = {"Juan":10,"Lorena":3}
>>> "Juan" in alumnos
True
```

Por último, el operador `not in` realiza lo contrario al operador `in`. Verifica que un elemento no está contenido en la secuencia.



Operador de Walrus

El operador `walrus` se introdujo en la versión Python 3.8, y se trata de un operador de asignación con una funcionalidad extra. Se representa con dos puntos seguidos de un igual :=, lo que tiene un parecido a una morsa, siendo ":" los ojos y "=" los colmillos, por lo que de ahí viene su nombre (*walrus* significa *morsa* en inglés).

Ejemplo:

```
>>> dato = "Hola"
>>> print(dato)
Hola
```

Con `walrus` podemos simplificar las dos líneas de código anterior en tan solo una.

(Asignación y salida por pantalla en una sola línea)

```
>>> print(dato:="Hola")
Hola
```

Operador a nivel bit , bitwise

Los operadores a nivel de bit son operadores que actúan sobre números enteros, pero usando su representación binaria. Si no sabes cómo se representa un número en forma binaria, no hace falta que te hagas mucho problema, ya que estos operadores no se usan tan frecuentemente como los otros. Su uso puede ser más empleado en lenguajes de bajo nivel, y Python no es un lenguaje de esas características. Pero no está de más conocerlos.

Operador	Acción
|	OR bit a bit
&	AND bit a bit
~	NOT bit a bit
^	XOR bit a bit
>>	Desplazamiento bit a la derecha
<<	Desplazamiento bit a la izquierda

Atención: *Estos operadores no se emplean para reemplazar al and o al or, los operadores bitwise no son los operadores lógicos que estamos acostumbrados a usar. Son operadores especiales que trabajan a nivel bit. Por eso no se usan habitualmente, dejamos en claro que estos no reemplazan a los otros, son cosas distintas.*

¡Sigamos trabajando!