OPERADORES DE PYTHON GRATIS

Sara Díaz-P.
Desarrolladora de
Algoritmos de Trading

SaraSTEM
Trading Algorítmico

Los **operadores en Python** son símbolos especiales que permiten realizar tareas específicas en un programa. Los operadores más utilizados en Python son los operadores aritméticos, de asignación, de comparación, lógicos, de pertenencia y de identidad.

Operadores aritméticos

Los operadores aritméticos se utilizan para realizar operaciones matemáticas como la adición, sustracción, multiplicación, división y modulo.

- **Adición (+)**

El operador de adición se utiliza para sumar dos valores.

- **Sustracción (-)**

El operador de sustracción se utiliza para restar dos valores.

- **Multiplicación (*)**

El operador de multiplicación se utiliza para multiplicar dos valores.

- **División (/)**

El operador de división se utiliza para dividir dos valores.

- **Módulo (%)**

El operador de módulo se utiliza para obtener el resto de una división entre dos valores.

- **División entera (//)**

El operador de división entera se utiliza para obtener la parte entera de una división entre dos valores.

- **Potencia (**)**

El operador de potencia se utiliza para elevar un valor a una potencia específica.

A continuación, vemos un ejemplo de cada uno:

```python
'''
Operadores ariméticos
'''
a = 10
b = 4

print(a + b) # 14
print(a - b) # 6
print(a * b) # 40
print(a / b) # 2.5
print(a % b) # 2
print(a // b) # 2
print(a ** b) # 10000
```

www.sarastem.com
hola@sarastem.com

Operadores de asignación

Los operadores de asignación se utilizan para asignar un valor a una variable.

- **Asignación (=)**

El operador de asignación se utiliza para asignar un valor a una variable.

- **Asignación de adición (+=)**

El operador de asignación de adición se utiliza para agregar un valor a una variable existente.

- **Asignación de sustracción (-=)**

El operador de asignación de sustracción se utiliza para restar un valor a una variable existente.

- **Asignación de multiplicación (*=)**

El operador de asignación de multiplicación se utiliza para multiplicar un valor a una variable existente.

- **Asignación de división (/=)**

El operador de asignación de división se utiliza para dividir un valor a una variable existente.

- **Asignación de módulo (%=)**

El operador de asignación de módulo se utiliza para calcular el resto de una división y asignar el resultado a una variable existente.

- **Asignación de división entera (//=)**

El operador de asignación de división entera se utiliza para calcular la parte entera de una división y asignar el resultado a una variable existente.

- **Asignación de potencia (**=)**

El operador de asignación de potencia se utiliza para elevar un valor a una potencia y asignar el resultado a una variable existente.

A continuación, vemos un ejemplo de cada uno:

```
'''
Operadores de asignación
'''
a = 5 # asignamos el valor inicial
b = 2

a += b # ahora a = 7
a -= b # a = 5
a *= b # a = 10
a /= b # a = 5
a %= b # a = 1
a //= b # a = 2
a **= b # a = 4
```

www.sarastem.com
hola@sarastem.com

Operadores de comparación

Los operadores de comparación se utilizan para comparar dos valores y devolver un valor verdadero o falso.

- **Igual a (==)**

El operador de igual a se utiliza para comparar si dos valores son iguales.

- **No igual a (!=)**

El operador de no igual a se utiliza para comparar si dos valores son diferentes.

- **Mayor que (>)**

El operador de mayor que se utiliza para comparar si un valor es mayor que otro.

- **Menor que (<)**

El operador de menor que se utiliza para comparar si un valor es menor que otro.

- **Mayor o igual a (>=)**

El operador de mayor o igual a se utiliza para comparar si un valor es mayor o igual a otro.

- **Menor o igual a (<=)**

El operador de menor o igual a se utiliza para comparar si un valor es menor o igual a otro.

A continuación, vemos un ejemplo de cada uno:

```
'''
Operadores de comparación
'''
a = 8
b = 4

print(a == b)  # False
print(a != b)  # True
print(a > b)  # True
print(a < b)  # False
print(a >= b)  # True
print(a <= b)  # False
```

Operadores lógicos

Los operadores lógicos se utilizan para combinar varias expresiones booleanas y devolver un valor verdadero o falso.

- **Operador AND (and)**

El operador AND se utiliza para evaluar dos o más condiciones y devolver verdadero solo si todas las condiciones son verdaderas.

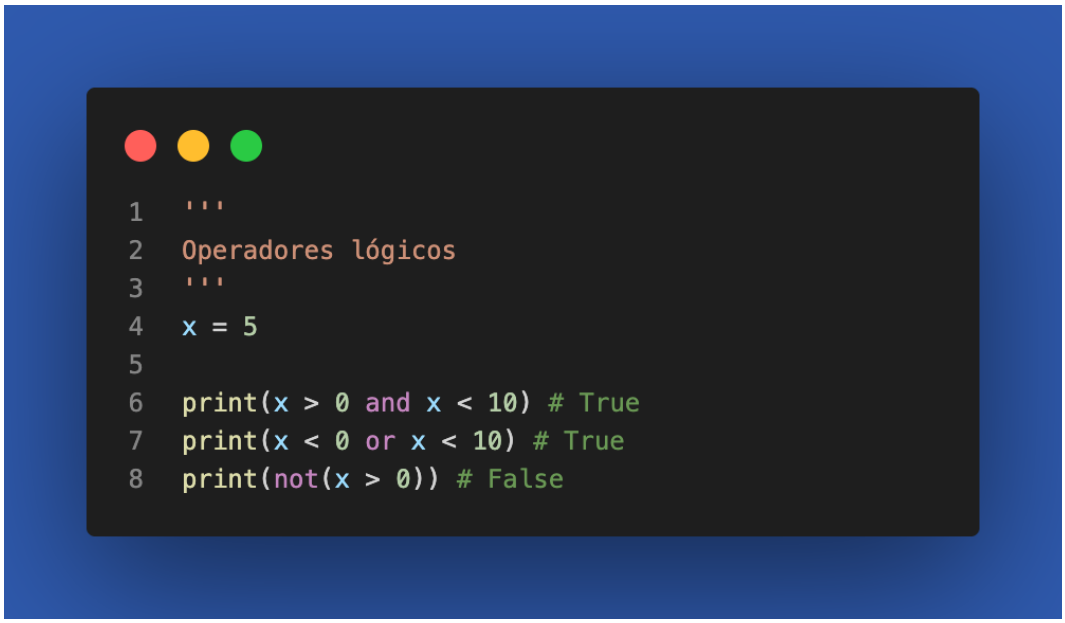
- **Operador OR (or)**

El operador OR se utiliza para evaluar dos o más condiciones y devolver verdadero si al menos una de las condiciones es verdadera.

- **Operador NOT (not)**

El operador NOT invierte el valor booleano de una condición.

A continuación, vemos un ejemplo de cada uno:



```
1  '''
2  Operadores lógicos
3  '''
4  x = 5
5
6  print(x > 0 and x < 10) # True
7  print(x < 0 or x < 10) # True
8  print(not(x > 0)) # False
```

Estos operadores lógicos son muy útiles para combinar múltiples condiciones y controlar el flujo de un programa. Por ejemplo, pueden utilizarse para evaluar múltiples condiciones en una estructura if y tomar una acción en consecuencia.

Operadores de pertenencia

Los operadores de pertenencia se utilizan para determinar si un objeto se encuentra en una secuencia, como una lista o una cadena.

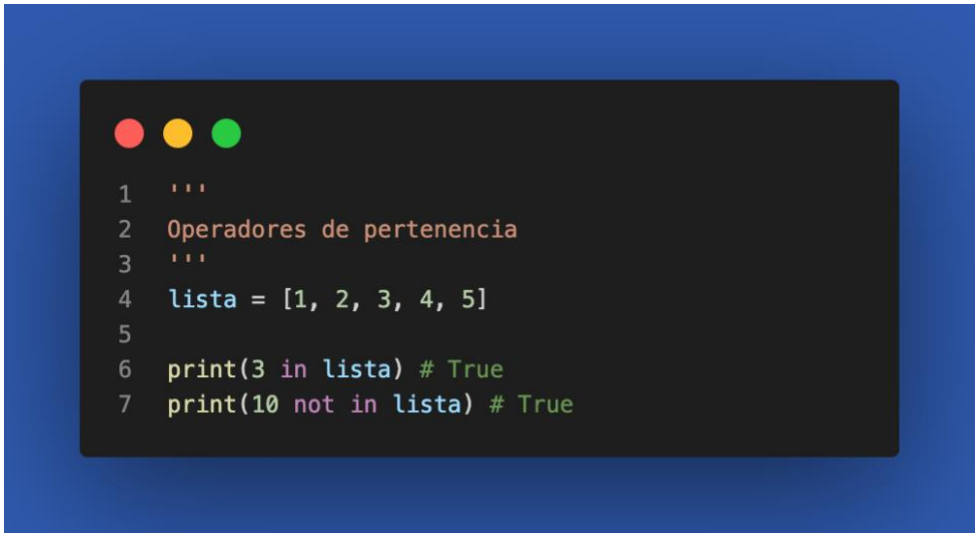
- **Operador in**

El operador in se utiliza para determinar si un elemento se encuentra en una secuencia, como una lista o una cadena.

- **Operador not in**

El operador not in se utiliza para determinar si un elemento no se encuentra en una secuencia.

A continuación, vemos un ejemplo de cada uno:



```
1  '''
2  Operadores de pertenencia
3  '''
4  lista = [1, 2, 3, 4, 5]
5
6  print(3 in lista) # True
7  print(10 not in lista) # True
```

Operadores de identidad

Los operadores de identidad se utilizan para determinar si dos objetos son el mismo objeto en la memoria.

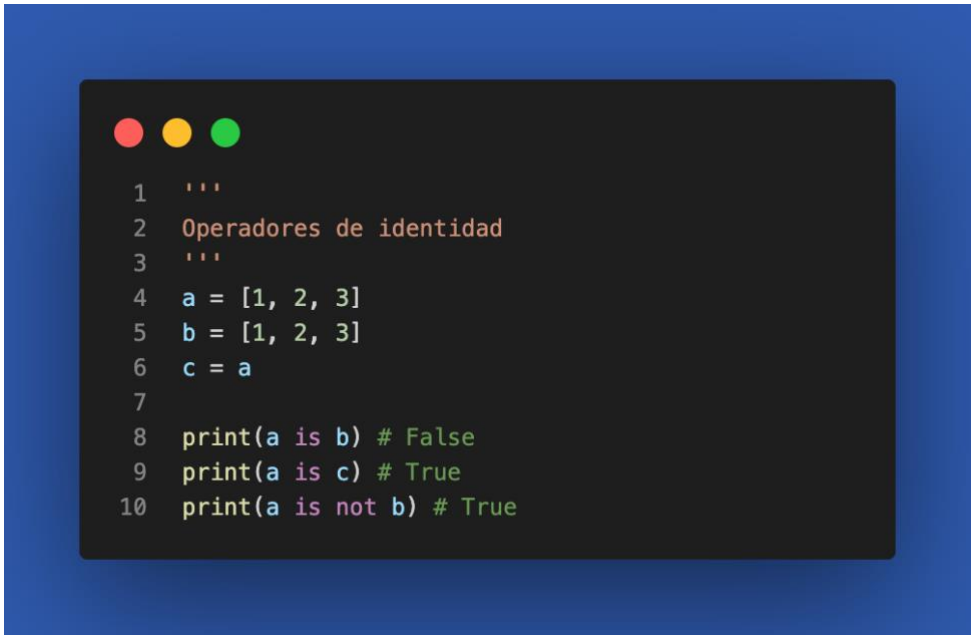
- **Operador is**

El operador is se utiliza para comparar si dos objetos son idénticos en la memoria.

- **Operador is not**

El operador is not se utiliza para comparar si dos objetos no son idénticos en la memoria.

A continuación, vemos un ejemplo de cada uno:



```
'''
Operadores de identidad
'''
a = [1, 2, 3]
b = [1, 2, 3]
c = a

print(a is b) # False
print(a is c) # True
print(a is not b) # True
```