Python Operators

Operators are special symbols in Python that carry out arithmetic or logical computation. The value that the operator operates on is called the operand.

For Example:

```
In [1]: 2 + 3
Out[1]: 5
```

Here, + is the operator that performs addition. 2 and 3 are the operands and 5 is the output of the operation.

Operator Type

1. Arithmetic operators
2. Comparison (Relational) operators
3. Logical (Boolean) operators
4. Bitwise operators
5. Assignment operators
6. Special operators

1. Arithmetic operators

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication etc.

Operator	Meaning
+	Add two operands or unary plus
-	Subtract right operand from the left or unary minus
*	Multiply two operands
/	Divide left operand by the right one (always results into float)
%	Modulus - remainder of the division of left operand by the right
//	Floor division - division that results into whole number adjusted to the left in the number line
**	Exponent - left operand raised to the power of right

Example:

In [2]:
```
x, y = 10, 20
#addition(+)
print(x + y )
#subtracton(-)

#multiplication(*)

#division(/)

#modulo division(%)

#Floor division(//)

#Exponent(**)
```

```
30
```

2. Comparison operators

Comparison operators are used to compare values. It either returns True or False according to the condition.

Operator	Meaning
>	Greater that - True if left operand is greater than the right
<	Less that - True if left operand is less than the right
==	Equal to - True if both operands are equal
!=	Not equal to - True if operands are not equal
>=	Greater than or equal to - True if left operand is greater than or equal to the right
<=	Less than or equal to - True if left operand is less than or equal to the right

Example:

In [3]:
```
a, b = 10, 20

#check a is less than b
a < b
#check a is greater than b

#check a is equal to b

#check a is not equal to b(!=)

#check a is greater than or equal to b

#check a is less than or equal to b
```

Out[3]: True

3. Logical Operators

Logical operators are the and, or, not operators.

Operator	Meaning
and	True if both the operands are true
or	True if either of the operands is true
not	True if operand is false (complements the operand)

Example:

In [4]:
```
a, b = True, False

#print(a and b)
#print(a or b)
#print(not b)
```

4. Bitwise operators

Bitwise operators act on operands as if they were string of binary digits. It operates bit by bit, hence the name.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
~	Bitwise NOT
^	Bitwise XOR
>>	Bitwise right shift
<<	Bitwise left shift

Example:

In [12]:
```
a, b = 10,4
#Bitwise AND

#Bitwise OR

#Bitwise NOT

#Bitwise XOR

#Bitwise RightShift

#Bitwise LeftShift
```

5. Assignment Operators

Assignment operators are used in Python to assign values to variables.

-, +=, -=, *=, /=, %=, //=, **=, &=, |=, >>=, <<= are Assignment Operator

In [11]:
```
a = 10

#Add AND(+=)

#Multiply AND(*=)

#Divide AND(/=)

#Modulus AND(%=)

#Floor Division AND(//=)

#Exponent AND(**=)
```

6. Special Operators

- Identity Operator

"is" and "is not" are the identity operators in Python. They are used to check if two values (or variables) are located on the same part of the memory. Two variables that are equal does not imply that they are identical.

Operator	Meaning
is True	if the operands are identical (refer to the same object)
is not	True if the operands are not identical (do not refer to the same object)

Example:

In [10]:
```
x= 5;y=5
x is y
```

Out[10]: True

- Membership Operators

"in" and "not in" are the membership operators in Python. They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).

Operator	Meaning
in	True if value/variable is found in the sequence
not in	True if value/variable is not found in the sequence

– .

```
In [8]: l = ['a','e','i','o','u']
        j = 'e'
        j in l
```

```
Out[8]: True
```