PYTHON-OPERATORS

OPERATORS IN PYTHON

In Python programming, Operators in general are used to perform operations on values and variables.

These are standard symbols used for the purpose of logical and arithmetic operations.

OPERATORS: These are the special symbols. Eg- + , * , /, etc.

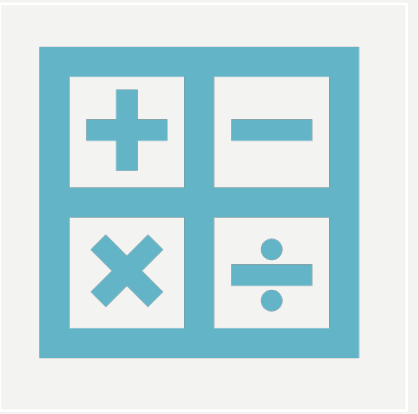
OPERAND: It is the value on which the operator is applied.

TYPES OF PYTHON OPERATORS

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

ARITHMETIC OPERATORS

Python arithmetic operators are used to perform mathematical operations on numerical values.



These operations are Addition, Subtraction, Multiplication, Division, Modulus, Exponents and Floor Division.

ARITHMETIC OPERATORS

Operator	Description	Syntax
+	Addition: adds two operands	x + y
–	Subtraction: subtracts two operands	x – y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when the first operand is divided by the second	x % y
**	Power: Returns first raised to power second	x ** y

PRECEDENCE OF ARITHMETIC OPERATORS IN PYTHON

The precedence of Arithmetic Operators in python is as follows:

- P – Parentheses
- E – Exponentiation
- M – Multiplication (Multiplication and division have the same precedence)
- D – Division
- A – Addition (Addition and subtraction have the same precedence)
- S – Subtraction

COMPARISON OPERATORS

- Python comparison operators compare the values on either sides of them and decide the relation among them.
- They are also called relational operators.
- These operators are equal, not equal, greater than, less than, greater than or equal to and less than or equal to.
- In python, the comparison operators have lower precedence than the arithmetic operators. All the operators within comparison operators have same precedence order.

COMPARISON OPERATORS

Operator	Name	Example
==	Equal	4 == 5 is not true.
!=	Not Equal	4 != 5 is true.
>	Greater Than	4 > 5 is not true.
<	Less Than	4 < 5 is true.
>=	Greater than or Equal to	4 >= 5 is not true.
<=	Less than or Equal to	4 <= 5 is true.

ASSIGNMENT OPERATORS

- Python assignment operators are used to assign values to variables.
- These operators include simple assignment operator, addition assign, subtraction assign, multiplication assign, division and assign operators etc.

ASSIGNMENT OPERATORS

Operator	Name	Example
=	Assignment Operator	a = 10
+=	Addition Assignment	a += 5 (Same as a = a + 5)
-=	Subtraction Assignment	a -= 5 (Same as a = a - 5)
*=	Multiplication Assignment	a *= 5 (Same as a = a * 5)
/=	Division Assignment	a /= 5 (Same as a = a / 5)
%=	Remainder Assignment	a %= 5 (Same as a = a % 5)
**=	Exponent Assignment	a **= 2 (Same as a = a ** 2)
//=	Floor Division Assignment	a //= 3 (Same as a = a // 3)

LOGICAL OPERATORS

Logical Operators in Python are used to perform logical operations on the values of variables.

The value is either true or false. We can figure out the conditions by the result of the truth values.

There are mainly three types of logical operators in python: logical AND, logical OR and logical NOT.

Operators are represented by keywords or special characters

Operator	Description	Syntax
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if the operand is false	not x

PRECEDENCE OF LOGICAL OPERATORS

The precedence of Logical Operators in python is as follows:

1. Logical not
2. logical and
3. logical or

Bitwise Operators

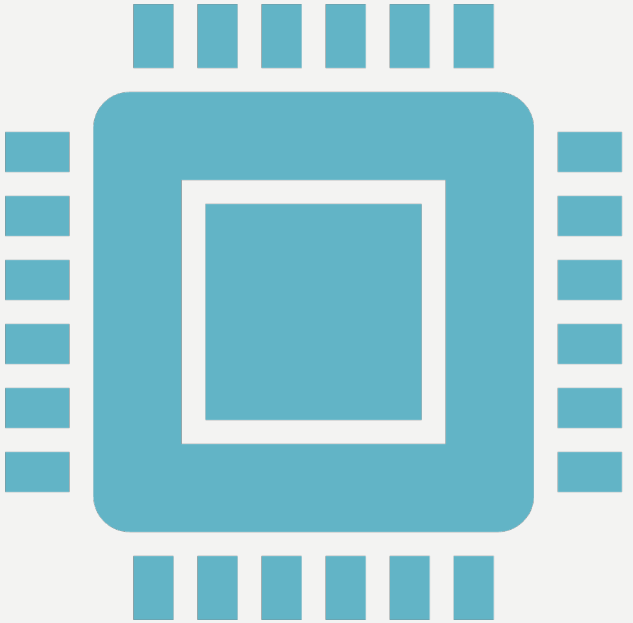
Python Bitwise operators act on bits and perform bit-by-bit operations. These are used to operate on binary numbers.

Operator	Description	Syntax
&	Bitwise AND	x & y
|	Bitwise OR	x | ys
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

PRECEDENCE OF BITWISE OPERATORS

The precedence of Bitwise Operators in python is as follows:

1. Bitwise NOT
2. Bitwise Shift
3. Bitwise AND
4. Bitwise XOR
5. Bitwise OR



Membership Operators

Python's membership operators test for membership in a sequence, such as strings, lists, or tuples.

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Identity Operators

Identity operators compare the memory locations of two objects.

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here **is** results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here **is not** results in 1 if id(x) is not equal to id(y).

TERNARY OPERATOR

- In Python, Ternary operators also known as conditional expressions are operators that evaluate something based on a condition being true or false. It was added to Python in version 2.5.
- It simply allows testing a condition in a **single line** replacing the multiline if-else making the code compact.

Syntax : [on_true] if [expression] else [on_false]

```
a, b = 10, 20
min = a if a < b else b # Copy value of a in min if a < b else copy b
print(min)
```