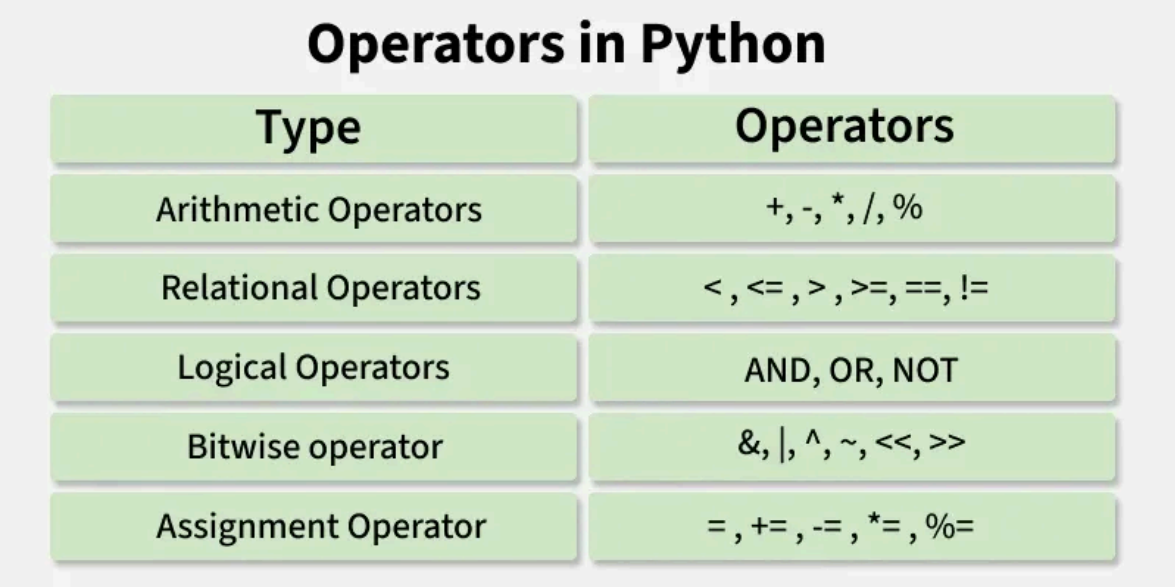
Python Operators

Last Updated : 2 Dec, 2025

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

- **Operators:** Special symbols like -, + , * , /, etc.
- **Operands:** Value on which the operator is applied.

Types of Operators in Python



Operators in Python

Arithmetic Operators

Python Arithmetic operators are used to perform basic mathematical operations like addition, subtraction, multiplication and division.

In Python 3.x the result of division is a floating-point while in Python 2.x division of 2 integers was an integer. To obtain an integer result in Python 3.x floored (// integer) is used.

```
# Variables
a = 15
b = 4

# Addition
print("Addition:", a + b)

# Subtraction
print("Subtraction:", a - b)

# Multiplication
print("Multiplication:", a * b)

# Division
print("Division:", a / b)
```

```
# Floor Division
print("Floor Division:", a // b)

# Modulus
print("Modulus:", a % b)

# Exponentiation
print("Exponentiation:", a ** b)
```

Output

```
Addition: 19
Subtraction: 11
Multiplication: 60
Division: 3.75
Floor Division: 3
Modulus: 3
Exponentiation: 50625
```

Note: *Refer to Differences between / and //.*

Comparison Operators

In Python, Comparison (or Relational) operators compares values. It either returns True or False according to the condition.

```
a = 13
b = 33

print(a > b)
print(a < b)
print(a == b)
print(a != b)
print(a >= b)
print(a <= b)
```

Output

```
False
True
False
True
False
True
```

Logical Operators

Python Logical operators perform **Logical AND**, **Logical OR** and **Logical NOT** operations. It is used to combine conditional statements.

The precedence of Logical Operators in Python is as follows:

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

```
a = True
b = False
```

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

Output

```
False
True
False
```

Bitwise Operators

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

Bitwise Operators in Python are as follows:

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

```
a = 10
b = 4

print(a & b)
print(a | b)
print(~a)
print(a ^ b)
print(a >> 2)
print(a << 2)
```

Output

```
0
14
-11
14
2
40
```

Assignment Operators

Python Assignment operators are used to assign values to the variables. This operator is used to assign the value of the right side of the expression to the left side operand.

Example

```
a = 10
b = a
print(b)
b += a
print(b)
b -= a
print(b)
b *= a
print(b)
```

```
b <<= a
print(b)
```

Output

```
10
20
10
100
102400
```

Identity Operators

In Python, **is** and **is not** are the identity operators both are used to check if two values are located on the same part of the memory. Two variables that are equal do not imply that they are identical.

is	*True if the operands are identical*
is not	*True if the operands are not identical*

```
a = 10
b = 20
c = a

print(a is not b)
print(a is c)
```

Output

```
True
True
```

Membership Operators

In Python, **in** and **not in** are the membership operators that are used to test whether a value or variable is in a sequence.

in	*True if value is found in the sequence*
not in	*True if value is not found in the sequence*

```
x = 24
y = 20
list = [10, 20, 30, 40, 50]

if (x not in list):
    print("x is NOT present in given list")
else:
    print("x is present in given list")

if (y in list):
    print("y is present in given list")
else:
    print("y is NOT present in given list")
```

Output

```
x is NOT present in given list
y is present in given list
```

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

print(min)
```

Output

```
10
```

Precedence and Associativity of Operators

In Python, Operator precedence and associativity determine the priorities of the operator.

Operator Precedence

This is used in an expression with more than one operator with different precedence to determine which operation to perform first.

```
expr = 10 + 20 * 30
print(expr)
name = "Alex"
age = 0

if name == "Alex" or name == "John" and age >= 2:
    print("Hello! Welcome.")
else:
    print("Good Bye!!")
```

Output

```
610
Hello! Welcome.
```

Operator Associativity

If an expression contains two or more operators with the same precedence then Operator Associativity is used to determine. It can either be Left to Right or from Right to Left.

```
print(100 / 10 * 10)
print(5 - 2 + 3)
print(5 - (2 + 3))
```

```
print(2 ** 3 ** 2)
```

Output

```
100.0
6
0
512
```

Related Links:

- Quiz on Python Operators
- Operator Overloading
- Why are there no ++ and - Operator in Python
- Difference between == and is Operator

Recommended Problems:

- Arithmetic Operators
- Logical Operators
- Bitwise Operators
- Modulo Task
- Last Digit of a number
- Sum of N Numbers
- LCM And GCD

Arithmetic Operators in Python

Visit Course

Suggested Quiz 10 Questions

What is the primary purpose of operators in Python programming?

A To format strings

B To perform operations on values and variables

C To manage memory allocation

D To handle exceptions

Comment | S SHARI... + Follow | 370

Article Tags: Misc Python Python-Operators python-basics

GeeksforGeeks
Sanchhaya Education Private Limited

Corporate & Communications Address:
A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

Registered Address:
K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305