**Python 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. In this article, we will look into different types of Python operators.

## **Arithmetic Operators**

Arithmetic operators are used to performing mathematical operations like addition, subtraction, multiplication, and division.

| 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 |

### **Example: Arithmetic operators in Python**

* Python3

| # Examples of Arithmetic Operator  a = 9  b = 4    # Addition of numbers  add = a + b    # Subtraction of numbers  sub = a - b    # Multiplication of number  mul = a \* b    # Division(float) of number  div1 = a / b    # Division(floor) of number  div2 = a // b    # Modulo of both number  mod = a % b    # Power  p = a \*\* b    # print results  print(add)  print(sub)  print(mul)  print(div1)  print(div2)  print(mod)  print(p) |
| --- |

**Output**

13

5

36

2.25

2

1

6561

## **Comparison Operators**

Comparisonof Relational operators compares the values. It either returns **True** or **False** according to the condition.

| Operator | Description | Syntax |
| --- | --- | --- |
| > | Greater than: True if the left operand is greater than the right | x > y |
| < | Less than: True if the left operand is less than the right | x < y |
| == | Equal to: True if both operands are equal | x == y |
| != | Not equal to – True if operands are not equal | x != y |
| >= | Greater than or equal to True if the left operand is greater than or equal to the right | x >= y |
| <= | Less than or equal to True if the left operand is less than or equal to the right | x <= y |

### **Example: Comparison Operators in Python**

* Python3

| # Examples of Relational Operators  a = 13  b = 33    # a > b is False  print(a > b)    # a < b is True  print(a < b)    # a == b is False  print(a == b)    # a != b is True  print(a != b)    # a >= b is False  print(a >= b)    # a <= b is True  print(a <= b) |
| --- |

**Output**

False

True

False

True

False

True

## **Logical Operators**

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

| 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 |

### **Example: Logical Operators in Python**

* Python3

| # Examples of Logical Operator  a = True  b = False    # Print a and b is False  print(a and b)    # Print a or b is True  print(a or b)    # Print not a is False  print(not a) |
| --- |

**Output**

False

True

False

## **Assignment Operators**

[Assignment operators](https://www.geeksforgeeks.org/assignment-operators-in-python/) are used to assigning values to the variables.

| Operator | Description | Syntax |
| --- | --- | --- |
| = | Assign value of right side of expression to left side operand | x = y + z |
| += | Add AND: Add right-side operand with left side operand and then assign to left operand | a+=b a=a+b |
| -= | Subtract AND: Subtract right operand from left operand and then assign to left operand | a-=b a=a-b |
| \*= | Multiply AND: Multiply right operand with left operand and then assign to left operand | a\*=b a=a\*b |
| /= | Divide AND: Divide left operand with right operand and then assign to left operand | a/=b a=a/b |
| %= | Modulus AND: Takes modulus using left and right operands and assign the result to left operand | a%=b a=a%b |
| //= | Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand | a//=b a=a//b |
| \*\*= | Exponent AND: Calculate exponent(raise power) value using operands and assign value to left operand | a\*\*=b a=a\*\*b |
| &= | Performs Bitwise AND on operands and assign value to left operand | a&=b a=a&b |
| |= | Performs Bitwise OR on operands and assign value to left operand | a|=b a=a|b |
| ^= | Performs Bitwise xOR on operands and assign value to left operand | a^=b a=a^b |
| >>= | Performs Bitwise right shift on operands and assign value to left operand | a>>=b a=a>>b |
| <<= | Performs Bitwise left shift on operands and assign value to left operand | a <<= b a= a << b |

### **Example: Assignment Operators in Python**

* Python3

| # Examples of Assignment Operators  a = 10    # Assign value  b = a  print(b)    # Add and assign value  b += a  print(b)    # Subtract and assign value  b -= a  print(b)    # multiply and assign  b \*= a  print(b)    # bitwise lishift operator  b <<= a  print(b) |
| --- |

**Output**

10

20

10

100

102400

## **Identity Operators**

**is** and **is not** are the [identity operators](https://www.geeksforgeeks.org/python-membership-identity-operators-not-not/) 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

### **Example: Identity Operator**

* Python3

| a = 10  b = 20  c = a    print(a is not b)  print(a is c) |
| --- |

**Output**

True

True

## **Membership Operators**

**in** and **not in** are the membership operators; 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-------x--------x

**\*It should be explained that the ‘in’ operator traverses through the entire sequence to check if the value is present or not. So whilst it is useful, it should be avoided if the sequence to be checked is large because it will take a lot of time to run. ----- x ------ x --------x**

### **Example: Membership Operator**

* Python3

| # Python program to illustrate  # not 'in' operator  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

## **Precedence and Associativity of Operators**

**Precedence and Associativity of Operators:** 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.

### **Example: Operator Precedence**

* Python3

| # Examples of Operator Precedence    # Precedence of '+' & '\*'  expr = 10 + 20 \* 30  print(expr)    # Precedence of 'or' & 'and'  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.

### **Example: Operator Associativity**

* Python3

| # Examples of Operator Associativity    # Left-right associativity  # 100 / 10 \* 10 is calculated as  # (100 / 10) \* 10 and not  # as 100 / (10 \* 10)  print(100 / 10 \* 10)    # Left-right associativity  # 5 - 2 + 3 is calculated as  # (5 - 2) + 3 and not  # as 5 - (2 + 3)  print(5 - 2 + 3)    # left-right associativity  print(5 - (2 + 3))    # right-left associativity  # 2 \*\* 3 \*\* 2 is calculated as  # 2 \*\* (3 \*\* 2) and not  # as (2 \*\* 3) \*\* 2  print(2 \*\* 3 \*\* 2) |
| --- |

**Output**

100.0

6

0

512

**\*The knowledge of the bitwise operators is not very useful in this course but it holds an important role in python programming and dsa. So, it is upto the discretion of the instructor to cover this topic or not ---- x ------- x**

## 

## **Bitwise Operators**

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

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

### **Example: Bitwise Operators in Python**

* Python3

| # Examples of Bitwise operators  a = 10  b = 4    # Print bitwise AND operation  print(a & b)    # Print bitwise OR operation  print(a | b)    # Print bitwise NOT operation  print(~a)    # print bitwise XOR operation  print(a ^ b)    # print bitwise right shift operation  print(a >> 2)    # print bitwise left shift operation  print(a << 2) |
| --- |

**Output**

0

14

-11

14

2

40