Operators

- The operator is a symbol that performs a specific operation between two operands, according to one definition.
- Operators serve as the foundation upon which logic is constructed in a program in a particular programming language.
- In every programming language, some operators perform several tasks. Same as other languages, Python also has some operators, and these are given below -

  - **Arithmetic operators**
  - **Comparison operators**
  - **Assignment Operators**
  - **Logical Operators**
  - **Bitwise Operators**
  - **Membership Operators**
  - **Identity Operators**

❖ Arithmetic Operators in Python

Python Arithmetic operators are used to perform basic 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

Operator	Description	Syntax
//	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:

```
a = 9
b = 4
# Addition of numbers
add = a + b
# Subtraction of numbers
sub = a - b
# Multiplication of number
mul = a * b
# Modulo of both number
mod = a % b
print(add)
print(sub)
print(mul)
print(mod)
```

Output:

```
13
5
36
1
6561
```

❖ Comparison Operators in Python

In Python Comparison_of 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:

```
# 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)
```

Output:

```
False
True
False
True
False
```

❖ Assignment Operators in Python

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

Operator	Description	Syntax
=	Assign the value of the right side of the expression to the 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

Operator	Description	Syntax
|=	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:

```
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)
```

Output:

```
10
20
10
100
```

❖ Logical Operators in Python

Python 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:

```
# 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
```

❖ Bitwise Operators in Python

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 | y
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

Example:

```
# Examples of Bitwise operators
a = 10
b = 4
# Print bitwise AND operation
print(a & b)
# Print bitwise OR operation
print(a | b)
```

Output:

```
0
14
```

❖ Membership Operators in Python

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

Example:

```
# 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
```

❖ Identity Operators in Python

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

Example:

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


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

Output:

```
True
True
```