# Operators in Python

⇨ Operator is a symbol that performs certain operations.

⇨ Python provides the following set of operators

1. Arithmetic operators
2. Relational Operator / Comparison operators
3. Assignment Operators
4. Logical Operators
5. Bitwise Operators
6. Special Operators:
   a. Membership Operators
   b. Identity Operators
   c. Conditional operator

Note: python does not support increment and decrement operator.

## Arithmetic Operators:

| Operator | Description |
|---|---|
| **+ (Addition)** | It is used to add two operands. For example, if a = 20, b = 10 => a+b = 30 |
| **- (Subtraction)** | It is used to subtract the second operand from the first operand. If the first operand is less than the second operand, the value results negative. For example, if a = 20, b = 10 => a - b = 10 |
| **/ (divide)** | It returns the quotient after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a/b = 2.0 |
| *** (Multiplication)** | It is used to multiply one operand with the other. For example, if a = 20, b = 10 => a * b = 200 |
| **% (reminder)** | It returns the reminder after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a%b = 0 |
| **** (Exponent)** | It is an exponent operator represented as it calculates the first operand power to the second operand. |
| **// (Floor division)** | It gives the floor value of the quotient produced by dividing the two operands. |

**Sample Program:**

**Write a python program for arithmetic operators for dynamic values.**

```
"""
Write a python program for All arithmatic Operators:
"""

x = int (input("Enter Frist Integer"))
y = int (input("Enter Second Ingeger"))

print("The Addition of ",x," and ",y," is ",(x+y))
print("The Subtraction of ",x," and ",y," is ",(x-y))
print("The Multiplication of ",x," and ",y," is ",(x*y))
print("The Division of ",x," and ",y," is ",(x/y))
print("The Modulus of ",x," and ",y," is ",(x%y))
print("The Floor-Division of ",x," and ",y," is ",(x//y))
print("The Power of ",x," and ",y," is ",(x**y))
```

## Comparison or Relational Operator

| Operator | Description |
|---|---|
| == | If the value of two operands is equal, then the condition becomes true. |
| != | If the value of two operands is not equal, then the condition becomes true. |
| <= | If the first operand is less than or equal to the second operand, then the condition becomes true. |
| >= | If the first operand is greater than or equal to the second operand, then the condition becomes true. |
| > | If the first operand is greater than the second operand, then the condition becomes true. |
| **<** | If the first operand is less than the second operand, then the condition becomes true. |

Relational Operator / comparison operator example

```
>>> a=10
>>> b=20
>>> print(a>b)
False
>>> print(a>=b)
False
>>> print(a<b)
```

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

## Bitwise Operators

| Operator | Description |
|---|---|
| & (binary and) | If both the bits at the same place in two operands are 1, then 1 is copied to the result. Otherwise, 0 is copied. |
| \| (binary or) | The resulting bit will be 0 if both the bits are zero; otherwise, the resulting bit will be 1. |
| ^ (binary xor) | The resulting bit will be 1 if both the bits are different; otherwise, the resulting bit will be 0. |
| ~ (negation) | It calculates the negation of each bit of the operand, i.e., if the bit is 0, the resulting bit will be 1 and vice versa. |
| << (left shift) | The left operand value is moved left by the number of bits present in the right operand. |
| >> (right shift) | The left operand is moved right by the number of bits present in the right operand. |

Example for : Bitwise operator

```
>>> a=95
>>> b=45
>>> print(a&b)
13
>>> print(a|b)
127
>>> print(~a)
-96
>>> print(a^b)
114
>>> print(a<<3)
760
>>> print(a>>3)
11
```

Python program for – Bitwise Operators

```
x = int(input("Enter First Integer"))
```

```
y=int(input("Enter Second Integer"))

print("The Bitwise - and of ",x," and ",y," is :",(x&y))
print("The Bitwise - or of ",x," and ",y," is :",(x|y))
print("The Bitwise - Xor of ",x," and ",y," is :",(x^y))
print("The Bitwise - Negation of  is :",(~x))
print("The Bitwise - leftshift of ",x," and 3 is :",(x << 3))
print("The Bitwise - Right shift of ",x," and 3 is :",(x >> 3))
```

## Assignment Operators

| Operator | Description |
|---|---|
| = | It assigns the value of the right expression to the left operand. |
| += | It increases the value of the left operand by the value of the right operand and assigns the modified value back to left operand. For example, if a = 10, b = 20 => a+ = b will be equal to a = a+ b and therefore, a = 30. |
| -= | It decreases the value of the left operand by the value of the right operand and assigns the modified value back to left operand. For example, if a = 20, b = 10 => a- = b will be equal to a = a- b and therefore, a = 10. |
| *= | It multiplies the value of the left operand by the value of the right operand and assigns the modified value back to then the left operand. For example, if a = 10, b = 20 => a* = b will be equal to a = a* b and therefore, a = 200. |
| %= | It divides the value of the left operand by the value of the right operand and assigns the reminder back to the left operand. For example, if a = 20, b = 10 => a % = b will be equal to a = a % b and therefore, a = 0. |
| **= | a**=b will be equal to a=a**b, for example, if a = 4, b =2, a**=b will assign 4**2 = 16 to a. |
| //= | A//=b will be equal to a = a// b, for example, if a = 4, b = 3, a//=b will assign 4//3 = 1 to a. |
| &= | A&=B will be equal to A = A& B |
| \|= | A\|=B will be equal to A=A\|B |
| ^= | A^=B will be equl to A = A^B |
| <<= | A<<=B will be equal to A = A<<B |
| >>= | A>>=B will be equal to A=A>>B |

**Example :**

1. If A =10 , B=20

    A+=B  # it evaluates  A+B and result will copies to A variable.

   A = 30 , B=20

## Logical Operators

The logical operators are used primarily in the expression evaluation to make a decision. Python supports the following logical operators.

| Operator | Description |
|---|---|
| and | If both the expression are true, then the condition will be true. If a and b are the two expressions, a → true, b → true => a and b → true. |
| or | If one of the expressions is true, then the condition will be true. If a and b are the two expressions, a → true, b → false => a or b → true. |
| not | If an expression **a** is true, then not (a) will be false and vice versa. |

## Membership Operators

Python membership operators are used to check the membership of value inside a Python data structure. If the value is present in the data structure, then the resulting value is true otherwise it returns false.

| Operator | Description |
|---|---|
| in | It is evaluated to be true if the first operand is found in the second operand (list, tuple, or dictionary). |
| not in | It is evaluated to be true if the first operand is not found in the second operand (list, tuple, or dictionary). |

**Eg:**

```
1) x="hello learning Python is very easy!!!"
2) print('h' in x) True
3) print('d' in x) False
4) print('d' not in x) True
5) print('Python' in x) True
```

## Identity Operators

The identity operators are used to decide whether an element certain class or type.

| Operator | Description |
|---|---|
| is | It is evaluated to be true if the reference present at both sides point to the same object. |
| is not | It is evaluated to be true if the reference present at both sides do not point to the same object. |

**Eg:**

```
a=10
b=10
print(a is b) True
x=True
y=True
print( x is y) True
```

**Eg:**

```
a="sru"
b="sru"
print(id(a))
print(id(b))
print(a is b)
```

## Ternary Operator:

Syntax:

```
x = TrueValue if condition else FalseValue
```

If condition is True then TrueValue will be considered else FalseValue will be considered.

Eg 1: write a python program to find the smallest number between two integer values

```
>>> A=10,20
>>>Small=A if A<B else B
>>> print(Small)
```

Eg 2: Read two numbers from the keyboard and print minimum value

```
a=int(input("Enter First Number:"))
b=int(input("Enter Second Number:"))
min=a if a<b else b
print("Minimum Value:",min)
```

```
Output:
Enter First Number:10
Enter Second Number:30
Minimum Value: 10
```
Note: Nesting of ternary operator is possible.

## Operator Precedence

The precedence of the operators is essential to find out since it enables us to know which operator should be evaluated first. The precedence table of the operators in Python is given below.

| Operator | Description |
|---|---|
| ** | The exponent operator is given priority over all the others used in the expression. |
| ~ + - | The negation, unary plus, and minus. |
| * / % // | The multiplication, divide, modules, reminder, and floor division. |
| + - | Binary plus, and minus |
| >> << | Left shift. and right shift |
| & | Binary and. |
| ^ \| | Binary xor, and or |
| <= < > >= | Comparison operators (less than, less than equal to, greater than, greater then equal to). |
| <> == != | Equality operators. |
| = %= /= //= -= += *= **= | Assignment operators |
| is is not | Identity operators |
| in not in | Membership operators |
| not or and | Logical operators |