Operators

Operators are special symbols in Python that carry out arithmetic or logical computation. The value that the operator operates on is called the operand.

Operator Types

1. Arithmetic operators
2. Comparison (Relational) operators
3. Logical (Boolean) operators
4. Bitwise operators
5. Assignment operators
6. Special operators

Arithmetic Operators

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication etc.

```
+ , -, *, /, %, //, **  are arithmetic operators
```

Example:

In [2]:

```
x, y = 10, 20

#addition
print(x + y)

#subtraction(-)

#multiplication(*)

#division(/)

#modulo division (%)

#Floor Division (//)

#Exponent (**)
```

```
30
```

Comparision Operators

Comparison operators are used to compare values. It either returns True or False according to the condition.

```
>, <, ==, !=, >=, <= are comparision operators
```

In [3]:

```
a, b = 10, 20

print(a < b)  #check a is less than b

#check a is greater than b

#check a is equal to b

#check a is not equal to b (!=)

#check a greater than or equal to b

#check a less than or equal to b
```

```
True
```

Logical Operators

Logical operators are **and, or, not** operators.

In [4]:

```
a, b = True, False

#print a and b
print(a and b)

#print a or b

#print not b
```

```
False
```

Bitwise operators

Bitwise operators act on operands as if they were string of binary digits. It operates bit by bit

```
&,  |,  ~,  ^,  >>,  << are Bitwise operators
```

In [5]:

```
a, b = 10, 4

#Bitwise AND
print(a & b)

#Bitwise OR


#Bitwise NOT


#Bitwise XOR


#Bitwise rightshift


#Bitwise Leftshift

```

```
0
```

Assignment operators

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

a = 5 is a simple assignment operator that assigns the value 5 on the right to the variable a on the left.

```
=,  +=,  -=,  *=,  /=,  %=,  //=,  **=, &=,  |=,  ^=,  >>=,  <<= are A
ssignment operators
```

In [6]:

```
a = 10

a += 10          #add AND
print(a)

#subtract AND (-=)

#Multiply AND (*=)

#Divide AND (/=)

#Modulus AND (%=)

#Floor Division (//=)

#Exponent AND (**=)
```

```
20
```

Special Operators

Identity Operators

is and is not are the identity operators in Python.

They are used to check if two values (or variables) are located on the same part of the memory.

In [7]:

```
a = 5
b = 5
print(a is b)    #5 is object created once both a and b points to same object

#check is not
```

```
True
```

In [8]:

```
l1 = [1, 2, 3]
l2 = [1, 2, 3]
print(l1 is l2)
```

```
False
```

In [9]:

```
s1 = "Satish"
s2 = "Satish"
print(s1 is not s2)
```

```
False
```

MemberShip Operators

in and not in are the membership operators in Python.

They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).

In [10]:

```
lst = [1, 2, 3, 4]
print(1 in lst)       #check 1 is present in a given list or not

#check 5 is present in a given list
```

```
True
```

In [11]:

```
d = {1: "a", 2: "b"}
print(1 in d)
```

```
True
```

Python Input and Output

Python Output

We use the print() function to output data to the standard output device

In [12]:

```
print("Hello World")
```

```
Hello World
```

In [15]:

```
a = 10
print("The value of a is", a) #python 3
print ("The value of a is " + str(a))
```

```
The value of a is 10
The value of a is 10
```

Output Formatting

In [16]:

```
a = 10; b = 20 #multiple statements in single line.

print("The value of a is {} and b is {}".format(a, b))    #default
```

```
The value of a is 10 and b is 20
```

In [17]:

```
a = 10; b = 20  #multiple statements in single line

print("The value of b is {1} and a is {0}".format(a, b)) #specify position of a
```

```
The value of b is 20 and a is 10
```

In [18]:

```
#we can use keyword arguments to format the string
print("Hello {name}, {greeting}".format(name="satish", greeting="Good Morning"))
```

```
Hello satish, Good Morning
```

In [19]:

```
#we can combine positional arguments with keyword arguments
print('The story of {0}, {1}, and {other}'.format('Bill', 'Manfred',
                                                       other='Georg'))
```

```
The story of Bill, Manfred, and Georg
```

Python Input

want to take the input from the user. In Python, we have the input() function to allow this.

In [21]:

```
num = input("Enter a number: ")
print (num)
```

```
Enter a number: 9
9
```