Operators:

- Operators are special symbols in Python that carry out arithmetic or logical computation.

Operator Types

1. Arithmetic operators
2. Assignment operators
3. Comparison (Relational) operators
4. Logical (Boolean) operators
5. Membership Operators
6. Identity Operators

Arithmetic Operators

- Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication etc.
-
  - , -, , */, %, //, ** are arithmetic operators

```
In [1]: # Addition
        3+5
```

```
Out[1]: 8
```

```
In [2]: # Subtraction
        54 - 46
```

```
Out[2]: 8
```

```
In [3]: # Multiplication
        2 * 3
```

```
Out[3]: 6
```

```
In [4]: # Division
        7/2
```

```
Out[4]: 3.5
```

- Division -> always gives output as float

```
In [5]: # floor division  -->quotient
        16//5
```

```
Out[5]: 3
```

```
In [6]: # Modulo --> remainder
        16%5
```

```
Out[6]: 1
```

```
In [7]: # power/exponent
        10**2
```

```
Out[7]: 100
```

```
In [8]: # parantasis
        (2 + 3) * (5 + 5)
```

```
Out[8]: 50
```

Arthimetic operators Precedence

- Paracentheis
- exponents
- floor division
- Multiplication
- Division
- Modulus
- Addition
- Subtraction

```
In [9]: 8//3*3/2+10%2**2
```

```
Out[9]: 5.0
```

When we use arthimetic operators, the boolean values will be automatically converted to int

```
In [10]: int(True)
```

```
Out[10]: 1
```

```
In [11]: int(False)
```

```
Out[11]: 0
```

```
In [12]: True + True
```

```
Out[12]: 2
```

```
In [13]: b = 3.9
         c = False
         b+c
Out[13]: 3.9
```

```
In [14]: b = 3.9
         c = False
         b/c
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-14-0468be761b07> in <module>
      1 b = 3.9
      2 c = False
----> 3 b/c

ZeroDivisionError: float division by zero
```

Assignment operators

- Assignment operators are used in Python to assign values to variables.
- (=, +=, -=, =, /=, %=, //=, *=) are Assignment operators
- First right side part will be executed and then assign to the left side variable

```
In [15]: # = is a simple assignment operator that assigns the value on the right to the va
         a=10
         a
Out[15]: 10
```

```
In [16]: id(a)
Out[16]: 140706160851008
```

```
In [17]: a=a+1
         print(a)

11
```

```
In [18]: id(a)
Out[18]: 140706160851040
```

```
In [19]: a-=4  # a=a-4
         a
Out[19]: 7
```

In [20]:
```
#Multiply AND (*=)

#Divide AND (/=)

#Modulus AND (%=)

#Floor Division (//=)

#Exponent AND (**=)
```

Comparison/Relational Operators

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

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

In [21]:
```
# is greater than
45>34
```
Out[21]:
```
True
```

In [22]:
```
# is less than
56<23
```
Out[22]:
```
False
```

In [23]:
```
3*3 < 4*2
```
Out[23]:
```
False
```

In [24]:
```
# is equal to
45 == 45
```
Out[24]:
```
True
```

In [25]:
```
# not equal
3!=5
```
Out[25]:
```
True
```

In [26]:
```
#greaterthan or equalto
1 >= 1
```
Out[26]:
```
True
```

```
In [27]: #lessthan or equalto
         5 <= 4

Out[27]: False
```

```
In [28]: 45==45.0

Out[28]: True
```

```
In [29]: 'hi' == 'HI'

Out[29]: False
```

Logical Operators

- It returns bool type only
- Logical operators are **and, or, not** operators.

```
In [30]: (1 > 2) or (2 < 3)

Out[30]: True
```

```
In [31]: (1 > 2) and (2 < 3)

Out[31]: False
```

```
In [32]: not True

Out[32]: False
```

```
In [33]: not False

Out[33]: True
```

```
In [34]: my_str='Siva'

         my_str.isalpha() or my_str.isalnum()  #isalphabets or is alphanumeric

Out[34]: True
```

```
In [35]: (2 == 2) or (3 == 3) and (3 == 4)   #--> "logical and" followed by "logical or"

Out[35]: True
```

Identity opertors

is and is not are the identity operators in Python.

They are used to check if two values (or variables) are indicating to same object or not

- is operator (# is - True if the opernds are identical)
- is not operators (# is not - True if the operands are not identical)

In [36]:
```
a = 5
b = 5
print(a is b)    #5 is object created once both a and b points to same object
```
```
True
```

In [37]:
```
s1 = "satish"
s2 = "Satish"
print(s1 is s2)
```
```
False
```

In [38]:
```
a=6
b=8
a is not b
```
Out[38]:
```
True
```

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 [39]:
```
a='venkatesh'
'esh' in a
```
Out[39]:
```
True
```

In [40]:
```
lst = [1, 2, 3, 4]
1 in lst              #check 1 is present in a given list or not
```
Out[40]:
```
True
```

In [41]:
```
a=[1,2,3,4,5,6]
9 not in a
```
Out[41]:
```
True
```

only arthimetic operators, return with value

remaining all operators, return boolean value

Operators precedence:

#Arthimetic Operators

#Comparision operators ((<,<=,>,>=,==,!=))

#membership

#identity

#Logical AND

#Logical OR