Python Operators

Operators are tokens that triggers some computation when applied to variables and other objects in an expression. Variable and objects to which the computation is applied, are called operands. So an operator requires some operands to work upon. Python provides following types of operators

1 . Arithmetic Operator

These are simple operators performing arithmetic operations, they are also seven types

+ addition
- subtraction
* multiplication

+, - and * have their basic meaning

/ division

Divide left operand with right and result is in float

```
>>>6/3
>>>2.0
```

% remainder

Remainder of division of left operand by the right

```
>>> 5 %2
1
```

** exponent(raise to power)

Left operand raised to the power of right

```
>>> 2**3
>>>8
```

// floor division

When we divide the left operand with right one then if there is any decimal part it adjusted into whole numbers to the left in the number line.

for e.g.

```
>>> 22 // 7
3
>>> 3.8 // 2
1.0
>>> 7 // 3
2
```

2. Assignment Operator

=	Assignment
/=	Assign quotient
+=	Assign sum
-=	Assign difference
*=	Assign product
%=	Assign remainder
**=	Assign exponent
//=	Assign floor division

```
>>> x=5
>>> x /=5      # x = x /5
>>> x
1.0
----------------------------------------
>>> x =5
>>> x %=5      # x = x % 5
>>> x
0
----------------------------------------
>>> x = 5
>>> x //=2
>>> x
2
----------------------------------------
>>> x **=5
```

```
>>> x
3125
```

---

3. Comparison Operator

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal to

they compare two values and after the comparison it will return Boolean values i.e. True/ False

```
>>> num1 = 5
>>> num2 = 7
>>> num1 > num2
False
>>> num1 >= num2
False
>>> num1 == num2
False
```

4. Logical Operator

We have basically three logical operators

- and
- or
- not

they compare two comparison expression and then return True or False
and will return True if both the expressions are True
or will return True if any one of them will be True

```
>>> num1 > 2 and num2 > 5
True
>>> num1 > 2 or num1 > 5
True
>>> num1 > 2 and num2 < 5
False
```

apart from comparing two relational expression it also work in some different manner like if we write

```
>>> x and y
```

it return x if x is false, otherwise it return y. x is false when it zero or negative number. for e.g.

```
>>>2 and 3
```

it return 3 because it 2 is not zero or negative.

in case of or operator

```
>>> x or y
```

it return y if x is False, x otherwise

```
>>> 2 or 3
2
```

5. Bitwise Operator

There are three bitwise operators

bitwise or	|
bitwise and	&
bitwise exclusive or	^

Bitwise or

if we apply bitwise 'or' on two numbers then first it convert number into bits(binary form) and then on each bit it perform or operation. In case of or the output will be 1 if any of the bit will be 1. for e.g.

```
>>> 7 | 5
111 ----->  7
101 ----->  5
---------------
111 ----> 7
```

Bitwise and

if we apply bitwise 'and' on two numbers then first it convert number into bits(binary form) and then on each bit it perform and operation. In case of and the output will be 1 if both the bits will be 1. for e.g.

```
>>> 7 & 5
111 ----->  7
101 ----->  5
---------------
101 ----> 5
```

Bitwise xor

if we apply bitwise 'xor' on two numbers then first it convert number into bits(binary form) and then on each bit it perform and operation. In case of and the output will be 1 if odd number ones in input, and 0 if both inputs are1. for e.g.

```
>>> 7 & 5
```

```
111 ----->  7
101 ----->  5
---------------
010 ----> 5
```

6. Identity Operator

There are two identity operator ‘is’ and ‘is not’

is -- True if operands are identical (refer to same object)

```
>>>x=5
>>>x is 5
>>> True
```

is not **--** True if the operands are not identical

```
>>>x=5
>>>x is not 5
>>> False
```

7. Membership Operator

There two membership operators ‘in’ and ‘not in’, that considered to deals with list. List basically collection of multiple objects.

```
x = [1, 2,  3,  4,  5]
```

in operator :

check an element finds in specified sequence then it gives True otherwise False.

```
>>> 3 in x
>>>True
```

not in operator:

True if it does not find elements in the specified sequence

```
>>> 3 not in x
>>> False
```