Operators

1.Arithmetic Operators

```
In [1]: x, y=10,5
```

Addition(+)

```
In [3]: x+y

Out[3]: 15
```

Substraction(-)

```
In [4]: x-y

Out[4]: 5
```

Multiplication(*)

```
In [5]: x*y

Out[5]: 50
```

Division(/)

```
In [7]: x/y

Out[7]: 2.0
```

Floor Division(//)

```
In [8]: x//y

Out[8]: 2
```

Modulus(%)

```
In [9]: x%y

Out[9]: 0
```

Exponentiation(**)

In [10]:
```
x ** y
```

Out[10]:
```
100000
```

2. Assignment Operators

=

In [18]:
```
x=2
```

In [19]:
```
x
```

Out[19]:
```
2
```

+=

In [20]:
```
x+=3 # x=x+3
x
```

Out[20]:
```
5
```

In [21]:
```
x+=2
x
```

Out[21]:
```
7
```

-=

In [22]:
```
x -=3
```

In [23]:
```
x
```

Out[23]:
```
4
```

*=

In [24]:
```
x *=4
x
```

Out[24]:
```
16
```

/=

In [25]:
```
x /=2
x
```

Out[25]:
```
8.0
```

//=

In [28]:
```
y=10
y //=2
y
```

Out[28]:
```
5
```

%=

In [29]:
```
y %=2
y
```

Out[29]:
```
1
```

**=

In [30]:
```
x=5
x **=3
x
```

Out[30]:
```
125
```

3.Unary Operator

In [31]:
```
n=7
n
```

Out[31]:
```
7
```

In [32]:
```
m=-(n)  # minus(-) is unary operator
m
```

Out[32]:
```
-7
```

In [33]:
```
n
```

```
Out[33]: 7
```

4. Relational Operator

```
In [34]: a=5
         b=6
```

==

```
In [35]: a==b

Out[35]: False
```

!=

```
In [36]: a != b

Out[36]: True
```

>

```
In [37]: a > b

Out[37]: False
```

<

```
In [38]: a < b

Out[38]: True
```

>=

```
In [39]: a >= b

Out[39]: False
```

<=

```
In [40]: a<=b
```

Out[40]:
```
True
```

5. Logical operators

(and, or, not)

In [1]:
```
a=5
b=4
```

and

In [4]:
```
a < 8 and b<5 # True and True =True
```

Out[4]:
```
True
```

In [5]:
```
a<8 and b<2  # True and False =False
```

Out[5]:
```
False
```

or

In [7]:
```
a<8 or b<2  # True or False =True
```

Out[7]:
```
True
```

not

In [9]:
```
x=False
x
```

Out[9]:
```
False
```

In [10]:
```
not x
```

Out[10]:
```
True
```

In [11]:
```
not not x
```

Out[11]:
```
False
```

Number System

In [12]:
```
25
```

```
Out[12]:  25

In [13]:  bin(25)

Out[13]:  '0b11001'

In [14]:  int(0b11001)

Out[14]:  25

In [15]:  bin(30)

Out[15]:  '0b11110'

In [16]:  int(0b11110)

Out[16]:  30

In [17]:  oct(25)

Out[17]:  '0o31'

In [18]:  int(0o31)

Out[18]:  25

In [19]:  bin(7)

Out[19]:  '0b111'

In [20]:  oct(25)

Out[20]:  '0o31'

In [21]:  int(0o31)

Out[21]:  25

In [22]:  hex(25)

Out[22]:  '0x19'

In [23]:  hex(256)

Out[23]:  '0x100'

In [24]:  int(0xa)

Out[24]:  10

In [25]:  hex(1)
```

```
Out[25]: '0x1'
```

```
In [26]: hex(25)
```

```
Out[26]: '0x19'
```

```
In [27]: int(0x19)
```

```
Out[27]: 25
```

Swap 2 variables in python

(a,b=5,6) After swap we should get ===> (a,b=6,5)

```
In [28]: a=5
         b=6
```

```
In [29]: a=b
         b=a
```

```
In [31]: print(a)
         print(b)
```

```
6
6
```

```
In [32]: # in above scenario we lost the vale 5
```

swap with help of third variable(temp)

```
In [33]: a1=7
         b1=8
```

```
In [34]: temp=a1
         a1=b1
         b1=temp
```

```
In [35]: print(a1)
         print(b1)
```

```
8
7
```

swap with no help of third variable

```
In [37]: a2=5
         b2=6
```

```
In [38]: a2=a2+b2
         b2=a2-b2
```

```
a2=a2-b2
```

```
In [39]: print(a2)
         print(b2)
```

```
6
5
```

other easy way to swap

```
In [41]: a3=10
         b3=20
```

```
In [42]: a3,b3=b3,a3
```

```
In [44]: print(a3)
         print(b3)
```

```
20
10
```

BITWISE OPERATOR()

1. complement(~)

```
In [45]: ~12
```

```
Out[45]: -13
```

```
In [46]: ~46
```

```
Out[46]: -47
```

```
In [47]: ~54
```

```
Out[47]: -55
```

2. and (&)

```
In [48]: 12 & 13
```

```
Out[48]: 12
```

```
In [49]: 1 & 0
```

```
Out[49]: 0
```

3. or(|)

```
In [52]: 12 | 13
```

```
Out[52]: 13
```

```
In [53]: 1 | 0
```

```
Out[53]: 1
```

4. Xor(^)

```
In [55]: 12 ^ 13
```

```
Out[55]: 1
```

```
In [56]: 25 ^ 30
```

```
Out[56]: 7
```

5. left shift(<<)

here we gain the bit

```
In [59]: 12 << 2 # we gain 2 zeros to bin(12)
```

```
Out[59]: 48
```

```
In [60]: 12 << 3  # we gain 3 zeros to bin(12)
```

```
Out[60]: 96
```

```
In [61]: 10 << 1
```

```
Out[61]: 20
```

```
In [62]: 10 << 2
```

```
Out[62]: 40
```

6. Right shift(>>)

here we loose the bit

```
In [63]: 10>>1
```

```
Out[63]: 5

In [64]: 10>>2

Out[64]: 2

In [65]: 10>>3

Out[65]: 1

In [66]: 20 >> 4

Out[66]: 1

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```