Python Operators

```
In [ ]:  1.Arithmetic operator
         2.Assignment Operator
         3.Relational Operator
         4.Logical Operator
         5.Unary Operator
         6.Bitwise operator
```

Arithmetic Operator

```
In [1]:  x1,y1=10,5
```

```
In [2]:  x1+y1
```

```
Out[2]:  15
```

```
In [3]:  x1*y1
```

```
Out[3]:  50
```

```
In [4]:  x1/y1
```

```
Out[4]:  2.0
```

```
In [5]:  x1//y1
```

```
Out[5]:  2
```

```
In [6]:  x1%y1
```

```
Out[6]:  0
```

```
In [8]:  x1**y1# x1 power y1
```

```
Out[8]:  100000
```

Assignment Operator

```
In [29]:  x=2
```

```
In [30]:  x=x+2
          x
```

```
Out[30]:  4
```

In [31]:

```
x+=2
x
```

Out[31]:

```
6
```

In [32]:

```
x*=2
x
```

Out[32]:

```
12
```

In [33]:

```
x-=2
x
```

Out[33]:

```
10
```

In [34]:

```
x//=2
x
```

Out[34]:

```
5
```

In [35]:

```
x%=2
x
```

Out[35]:

```
1
```

Unary Operator

1.unary means 1 || binary means 2 2.Here we are applying unary minus operator(-) on the operend n;the value of m beccomes-7,which indicates it as negative value.

In [58]:

```
n=7# Negation
n
```

Out[58]:

```
7
```

In [38]:

```
m=(-n)
m
```

Out[38]:

```
-7
```

Relational Operator

we use this operator for comparing

In [67]:

```
a=5
b=6
```

In [68]:

```
a<b
```

```
Out[68]: True

In [69]: b<a

Out[69]: False

In [70]: a>b

Out[70]: False

In [71]: a==b

Out[71]: False

In [78]: a=b # we cannot use = operator that means it is assigning

In [79]: a!=b

Out[79]: False

In [74]: # if i change b=6
         b=5
         b

Out[74]: 5

In [80]: a == b

Out[80]: True

In [82]: a >= b

Out[82]: True

In [83]: a <= b

Out[83]: True

In [84]: a<b

Out[84]: False

In [85]: a>b

Out[85]: False

In [86]: a!=b

Out[86]: False
```

Logical Operator

In [ ]:

```
1. logical operator true or false table
2. 3 important point of logical operator is( AND,OR,NOT )
3. In truth table TRUE=1, FALSE=0
```

In [ ]:

```
AND Truth Table
    x  y  c
    0  0  0
    0  1  0
    1  0  0
    1  1  1
```

In [ ]:

```
OR Truth Table
    x  y  c
    0  0  0
    0  1  1
    1  0  1
    1  1  1
```

In [88]:

```
a=5
b=4
```

In [89]:

```
a<8 and b<5 #refer to truth table
```

Out[89]:

```
True
```

In [90]:

```
a<8 and b<2
```

Out[90]:

```
False
```

In [91]:

```
a>8 or b<2
```

Out[91]:

```
False
```

In [94]:

```
x=False
x
```

Out[94]:

```
False
```

In [95]:

```
not x # not means reverse the operations
```

Out[95]:

```
True
```

In [96]:

```
x=not x
x
```

Out[96]:

```
True
```

In [97]:

```
not x
```

Out[97]:

```
False
```

Number system

Binary (Base-2) 0-1

Octal (Base-8) 0-7

Decimal(Base-10) 0-9

Hexa Decimal(Base-16) 0-9 abcdef (10 11 12 13 14 15)

```
In [ ]: Number system used in command prompt(ip config)
```

```
In [98]: 25
```

```
Out[98]: 25
```

```
In [99]: bin(25)
```

```
Out[99]: '0b11001'
```

```
In [ ]: bin(25)
               2|25
                |12 (2 into 12 is 24 remainder is 1)
                |6  (2 into 6 is 12 remainder is 0)
                |3  (2 into 3 is 6  remainder is 0)
                |1  (2 into 1 is 2 remainder is 1)
              ('set the value from last remiander to first')=11001
```

```
In [100…: int(0b11001)
```

```
Out[100…: 25
```

```
In [ ]: 0b is binary 11001-(1*2^4)+(1*2^3)+(0*2^2)+(0*2^1)+(1*2^0)
                          =16+8+0+0+1
                          =25
```

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

```
Out[1]: '0o31'
```

```
In [3]: int(0o31)
```

```
Out[3]: 25
```

```
In [4]: hex(25)
```

```
Out[4]: '0x19'
```

In [5]:

```
hex(16)
```

Out[5]:

```
'0x10'
```

In [10]:

```
hex(256)
```

Out[10]:

```
'0x100'
```

In [ ]:

```
0x19 0x means hexadecimal 19 we can write
                          19=(1*16^1)+(9*16^0)
                            =16+9=25
```

In [ ]:

```
0o31 0o means octal 31=(3*8^1)+(1*8^0)
                      =24+1=25
```

In [7]:

```
0xa
```

Out[7]:

```
10
```

In [6]:

```
0xb
```

Out[6]:

```
11
```

In [9]:

```
hex(1)
```

Out[9]:

```
'0x1'
```

In [8]:

```
hex(2)
```

Out[8]:

```
'0x2'
```

Swap two numbers

In [ ]:

```
a=5,b=6 after swaping we get a=6,b=5
---we can swap  using a,b=b,a
```

In [11]:

```
a=5
b=6
```

In [12]:

```
a=b
b=a
print(a)
print(b)
```

```
6
6
```

In [13]:

```
# in the above scenario we last the value 5
a1=7
b1=8
```

In [14]:

```
temp=a1
a1=b1
b1=temp
```

In [15]:

```
print(a1)
print(b1)
```

```
8
7
```

In [16]:

```
#variable formula without using 3 formula
a2=5
b2=6
```

In [21]:

```
a2=a2+b2# 5+6=11
b2=a2-b2# 11-6=5
a2=a1-b2# 11-5=6
```

In [28]:

```
print(a2)
print(b2)
```

```
17
-9
```

In [29]:

```
a1,b1
```

Out[29]:

```
(8, 7)
```

In [32]:

```
a1,b1=b1,a1
```

In [33]:

```
print(a1)
print(b1)
```

```
7
8
```

Bitwise Operators

In [34]:

```
The following are the bitwise operators

    NAME                    SYMBOL
1.Complement                  ~
2.AND                         &
3.OR                          |
4.XOR                         ^
5.LEFT SHIFT                  <<
6.RIGHT SHIFT                 >>
```

```
  Cell In[34], line 4
    1.Complement                  ~
     ^
SyntaxError: invalid decimal literal
```

Complement (~)

In [ ]:
```
1. Complement it will do reverse of binary format.
2. ~0 it will give you 1,~1 it will give you 0
3. In virtual memory it cannot store (-) numbers the only way to store nagative num
```

In [1]:
```
~12
```

Out[1]:
```
-13
```

In [2]:
```
~13
```

Out[2]:
```
-14
```

In [3]:
```
print(bin(12))
```

```
0b1100
```

In [4]:
```
print(bin(13))
```

```
0b1101
```

In [ ]:
```
The above output is calculated below
1.bin means binary, In binary base value is 2
```

In [6]:
```
0b1100
```

Out[6]:
```
12
```

In [7]:
```
0b1101
```

Out[7]:
```
13
```

In [ ]:
```
The above result showing as 13 below is the calculation
```

In [ ]:
```
0b1101:1 is replaced by 2^0
       1 is replaced by 2^1
       0 is replaced by 2^2
       1 is replaced by 2^3

    =(1*2^3)+(1*2^2)+(0*2^1)+(1*2^0)
    =8+4+0+1
    =13
```

AND

X	Y	X*Y
0	0	0
1	0	0
0	1	0
1	1	1

OR

X	Y	X+Y
0	0	0
1	0	1
0	1	1
1	1	1

In [35]:
```
12&13
```

Out[35]:
```
12
```

In [ ]:
```
The above output shows 12 as as a result, see the calculation below
binary of 12 is 1100
binary of 13 is 1101
truth table (REF:AND) 1100
SO 1100 is refered to binary number number of 12 that is why the result is 12
```

In [36]:
```
12|13
```

Out[36]:
```
13
```

In [ ]:
```
The above output shows 12 as as a result, see the calculation below
binary of 12 is 1100
binary of 13 is 1101
truth table (REF:OR) 1101
SO 1101 is refered to binary number number of 13 that is why the result is 13
```

XOR(^)

In [ ]:
```
    x   y   z
    0   0   0
    0   1   1
    1   0   1
    1   1   0
```

In [37]:
```
12^13
```

Out[37]:
```
1
```

In [ ]:
```
The above output shows 1 as as a result, see the calculation below
binary of 12 is 1100
binary of 13 is 1101
truth table (REF:XOR) 0001
0001 : (0*2^3)+(0*2^2)+(0*2^1)+(1*2^0)
     : 0 + 0 + 0 + 1 = 1
" Hence the result is 1"
```

In [38]:
```
print(bin(35))
print(bin(40))
```

```
0b100011
0b101000
```

In [39]:
```
35&40
```

Out[39]:
```
32
```

In [ ]:
```
The output shows as 32 see the calculation below
binary of 35 is 100011
binary of 40 is 101000
truth table (REF:AND) 100000

100000 : (1*2^5)+(0*2^4)+(0*2^3)+(0*2^2)+(0*2^1)+(0*2^0)
       : 32 + 0 + 0 + 0 + 0 + 0 = 32
" Hence the result is 32"
```

In [40]:

```
35|40
```

Out[40]:

```
43
```

In [ ]:

```
The output shows as 43 see the calculation below
binary of 35 is 100011
binary of 40 is 101000
truth table (REF:OR) 101011

101111 : (1*2^5)+(0*2^4)+(1*2^3)+(0*2^2)+(1*2^1)+(1*2^0)
       : 32 + 0 + 8 + 0 + 2 + 1 = 43
"Hence the result is 43"
```

<< Left Shift

left shift means gain the bit

In [41]:

```
bin(10)
```

Out[41]:

```
'0b1010'
```

In [43]:

```
10<<1 # this code refers to 1 bit(after symbol)
```

Out[43]:

```
20
```

In [ ]:

```
The above codes says 10<<1 bit it means
we need to bring one zero in then it is 10100
10100 : (1*2^4)+(0*2^3)+(1*2^2)+(0*2^1)+(0*2^0)
      : 16 + 0 + 4 + 0 + 0 = 20
"Hence the result is 20"
```

In [44]:

```
10<<2 # this code refer to 2 bit(after symbol)
```

Out[44]:

```
40
```

In [ ]:

```
101000 : (1*2^5)+(0*2^4)+(1*2^3)+(1*2^2)+(0*2^1)+(0*2^0)
       : 32 + 0 + 8 + 0 + 0 + 0 = 40
     "Hence the result is 40"
```

In [51]:

```
bin(20)
```

Out[51]:

```
'0b10100'
```

In [50]:

```
20<<4# 101000000
```

Out[50]:

```
320
```

In [ ]:

```

```

Right Shift>>

Right shift means we lose the bit

In [45]:

```
bin(10)
```

Out[45]:

```
'0b1010'
```

In [47]:

```
10>>1
```

Out[47]:

```
5
```

In [ ]:

```
101 : (1*2^2+0+1*2^0)
    :(4+1)=5
 result is 5
```

In [52]:

```
10>>2
```

Out[52]:

```
2
```

In [ ]:

```
we need to remove two numbers then it is 10
10 : 1*2^1)+(0*2^0)
: 2 + 0 = 2
"Hence the result is 2"
```

In [ ]:

```

```