Operator

1- ARITHMETIC OPERATOR ( + , -, , /, %, %%, *, ^)

2- ASSIGNMEN OPERATOR (+=, -=, *=, /=, //=)

3- RELATIONAL OPERATOR (<=, >=, ==, !=, <, >)

4- LOGICAL OPERATOR (AND, OR )

5- UNARY OPERATOR (-)

In [ ]:

Arithmetic Operator

In [3]:
```
x,y = 22,7
```

In [4]:
```
x+y
```

Out[4]: 29

In [5]:
```
x-y
```

Out[5]: 15

In [6]:
```
x/y
```

Out[6]: 3.142857142857143

In [7]:
```
x*y
```

Out[7]: 154

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

Out[8]: 3

In [9]:
```
x%y
```

Out[9]: 1

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

Out[10]: 2494357888

In [11]:
```
x1 = 7
y1 = 7
```

In [12]:
```
x1 ** y1
```

Out[12]: 823543

## Assignment Operator

In [14]:
```
x=7
x
```

Out[14]: 7

In [15]:
```
x = x+7
x
```

Out[15]: 14

In [16]:
```
x+=7
```

In [17]:
```
x
```

Out[17]: 21

In [18]:
```
x+=7
x
```

Out[18]: 28

In [19]:
```
x*=2
```

In [20]:
```
x
```

Out[20]: 56

In [22]:
```
x-=10
x
```

Out[22]: 46

In [23]:
```
x/=2
```

In [24]:
```
x
```

Out[24]: 23.0

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

Out[25]: 11.0

In [27]:
```
a, b = 7,8 # you can assigned variable in one line as well
print(a)
print(b)
```

```
7
8
```

In [28]:
```
a = 7
b = 8
print(a)
print(b)
```

```
7
8
```

In [29]:
```
a
```

Out[29]: 7

In [30]:
```
b
```

Out[30]: 8

In [ ]:
```

```

Unary Operator

unary means 1 || binary means 2

Here we are applying unary minus operator(-) on the operand n; the value of m becomes -7, which indicates it as a negative value.

In [32]:
```
n = 7
n
```

Out[32]: 7

In [33]:
```
m = -(n)
m
```

Out[33]: -7

In [36]:
```
n
```

Out[36]: 7

In [37]:
```
-n
```

Out[37]: -7

In [38]:
```
m
```

Out[38]: -7

In [ ]:
```

```

Relational Operator

We are using this operator for comparing

In [39]:
```
a = 6
b = 7
```

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

Out[40]: True

In [41]:
```
a>b
```

Out[41]: False

In [42]:
```
# a=b # we cannot use = operatro that means it is assigning
a==b
```

Out[42]: False

In [43]:
```
a!=b
```

Out[43]: True

In [44]:
```
a = 7
a
```

Out[44]: 7

In [46]:
```
b=7
b
```

Out[46]: 7

In [47]:
```
a>b
```

Out[47]: False

In [48]:
```
a<b
```

Out[48]: False

In [49]:
```
a>=b
```

Out[49]: True

In [50]:
```
a<=b
```

Out[50]: True

In [51]:
```
a == b
```

Out[51]: True

In [52]:
```
b = 9
```

In [53]:
```
a!= b
```

Out[53]: True

In [ ]:
```

```

Logical Operator

logical operator you need to understand about true & false table

3 importand part of logical operator is --> AND, OR, NOT

1 AND 1 is 1 and rest of them are 0

0 OR 0 is 0 and rest of them are 1

In [54]:
```
a = 6
b = 4
```

In [55]:
```
a < 8 and b < 5 #refers to the truth table
```

Out[55]: True

In [56]:
```
a < 8 and b < 2
```

Out[56]: False

In [57]:
```
a < 8 or b < 2
```

Out[57]: True

In [58]:
```
x = False
x
```

Out[58]: False

In [59]:
```
not x
```

Out[59]: True

In [60]:
```
x = not x
x
```

Out[60]: True

In [61]:
```
not x
```

Out[61]: False

In [ ]:

Number System Conversion (Bit - Binary digit)

In the programing we are using binary system, octal system, decimal system & hexadecimal system

but where do we use this in cmd - you can check your ip address & lets understand how to convert from one system to other system

when you check ipaddress you will these format --> cmd - ipconfig

binary : base(2) (0-1) --> please divide 15/2 & count in reverse order

ocatl : base(8) (0-7)

hexadecimal : base(16)(0-9 & then a-f)

decimal : base(10)

In [62]:
```
25
```

Out[62]:
```
25
```

In [63]:
```
bin(25)
```

Out[63]:
```
'0b11001'
```

In [64]:
```
int(0b11001)
```

Out[64]:
```
25
```

In [65]:
```
bin(30)
```

Out[65]:
```
'0b11110'
```

In [66]:
```
int(0b11110)
```

Out[66]:
```
30
```

In [69]:
```
bin(100000)
```

Out[69]:
```
'0b11000011010100000'
```

In [70]:
```
oct(25)
```

Out[70]:
```
'0o31'
```

In [72]:
```
int(0o31)
```

Out[72]: 25

In [78]:
```
oct(66)
```

Out[78]: '0o102'

In [79]:
```
0b101010
```

Out[79]: 42

In [80]:
```
int(0b101010)
```

Out[80]: 42

In [82]:
```
0o01234567
```

Out[82]: 342391

In [83]:
```
hex(16)
```

Out[83]: '0x10'

In [88]:
```
hex(20)
```

Out[88]: '0x14'

In [89]:
```
0x46
```

Out[89]: 70

In [90]:
```
hex(1)
```

Out[90]: '0x1'

In [ ]:
```

```

Swap 2-variable in python

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

In [91]:
```
a = 5
b = 6
```

In [92]:
```
a = b
b = a
```

In [93]:
```
print(a)
print(b)
```

```
6
6
```

In [94]:
```
# in above scenario we lost the value 5
a1 = 7
b1 = 8
```

In [95]:
```
temp = a1
a1 = b1
b1 = temp
```

In [96]:
```
print(a1)
print(b1)
```

```
8
7
```

In [97]:
```
temp
```

Out[97]: 7

In [ ]:

In [100…
```
a2 = 5
b2 = 6
```

In [101…
```
#swap variable formulas without using 3rd formul
a2 = a2 + b2 # 5+6 = 11
b2 = a2 - b2 # 11-6 = 5
a2 = a2 - b2 # 11-5 = 6
```

In [103…
```
print(a2)
print(b2)
```

```
6
5
```

In [104…
```
0b110
```

Out[104… 6

In [105…
```
0b101
```

Out[105… 5

In [106…
```
print(0b110)
print(0b101)
```

```
6
5
```

In [107…
```
print(0b101)
print(0b110)
```

```
5
6
```

In [108…
```
# but when we use a2 + b2 then we get 11 that means we will get 4 bit which is 1
print(bin(11))
print(0b1011)
```

```
0b1011
11
```

In [ ]:
```

```

XOR

-There is other way to work using swap variable also which is XOR because it will not waste extra bit

In [109…
```
print(a2)
print(b2)
```

```
6
5
```

In [111…
```
#There is other way to work using swap variable also which is XOR because it wil
a2 = a2 ^ b2
b2 = a2 ^ b2
a2 = a2 ^ b2
```

In [112…
```
print(a2)
print(b2)
```

```
6
5
```

In [113…
```
a2,b2
```

Out[113… (6, 5)

In [114…
```
a2,b2=b2,a2
```

In [115…
```
print(a2)
print(b2)
```

```
5
6
```

Bitwise Oprator

We Have 6 Operators

Complement(~)|| AND (&) || OR(|) || XOR(^)||Left Shift(<<)||Right Shift(>>)

In [116…
```
print(bin(12))
print(bin(13))
```

```
0b1100
0b1101
```

In [117…
```
0b1100
```

Out[117…
```
12
```

In [118…
```
0b1101
```

Out[118…
```
13
```

# complement --> you will get this key below esc character

12 ==> 1100 ||

first thing we need to understand what is mean by complement.

complement means it will do reverse of the binary format i.e. - ~0 it will give you 1 ~1 it will give 0

12 binary format is 00001100 ( complement of ~00001100 reverse the number - 11110011 which is (-13)

but the question is why we got -13

to understand this concept ( we have concept of 2's complement

2's complement mean (1's complement + 1)

in the system we can store +Ve number but how to store -ve number

lets understand binary form of 13 - 00001101 + 1

In [119…
```
~12
```

Out[119…
```
-13
```

In [120…
```
~45
```

Out[120…
```
-46
```

In [121…
```
~9958
```

Out[121…
```
-9959
```

Bit wise and operator

AND - LOGICAL OPERATOR ||| & - BITWISE AND OPERATOR

(we know that 1 & 1 is 1) 12 - 00001100 13 - 00001101 when we are add both then then outut we will get as 12

In [122…
```
12 & 13
```
Out[122… 12

In [123…
```
12|13
```
Out[123… 13

In [124…
```
1 & 0
```
Out[124… 0

In [125…
```
1|0
```
Out[125… 1

In [126…
```
bin(13)
```
Out[126… '0b1101'

In [127…
```
print(bin(35))
print(bin(40))
```
```
0b100011
0b101000
```

In [128…
```
5&40
```
Out[128… 0

In [129…
```
35|40
```
Out[129… 43

In [136…
```
5&10
```
Out[136… 0

In [137…
```
12^13
```
Out[137… 1

In [138…
```
10^18
```
Out[138… 24

# BIT WISE LEFT OPERATOR

# bit wise left operator bydefault you will take 2 zeros ( )

# 10 binary operator is 1010 | also i can say 1010

10<<2

In [140…
```
20<<4
```

Out[140…
```
320
```

# BITWISE RIGHTSHIFT OPERATOR

In [141…
```
10>>2
```

Out[141…
```
2
```

In [142…
```
bin(20)
```

Out[142…
```
'0b10100'
```

In [143…
```
20>>4
```

Out[143…
```
1
```

In [ ]: