Unary Operators

```
n=7

m=(-n)

m

-7

n

7

-n

-7
```

Logical Operator

```
#AND OR NOT

a=5
b=4

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

True

a < 8 and b < 2

False

a < 8 or b < 2

True

x = False
x

False

not x

True

x = not x
x

True
```

```
not x
```

```
False
```

Number system coversion

```
binary : base (0-1) --> please divide 15/2 & count in reverse order
octal : base (0-7)
hexadecimal : base (0-9 & then a-f)
when you check ipaddress you will these format --> cmd - ipconfig
```

```
25
```

```
25
```

```
bin(25)
```

```
'0b11001'
```

```
0b11001
```

```
25
```

```
int(0b11001)
```

```
25
```

```
bin(20)
```

```
'0b10100'
```

```
int(0b10100)
```

```
20
```

```
oct(15)
```

```
'0o17'
```

```
0o17
```

```
15
```

```
hex(9)
```

```
'0x9'
```

```
0xf
```

```
15
```

```
hex(10)
```

```
'0xa'
```

```
hex(25)
```

```
'0x19'
```

```
hex(25)
```

```
'0x19'
```

```
0x15
```

```
21
```

swap variable in python

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

```
a = 5
b = 6

a = b
b = a

a,b = b,a

print(a)
print(b)
```

```
6
6
```

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

temp = a1
a1 = b1
b1 = temp

print(a1)
print(b1)
```

```
8
7
```

```
a2 = 5
b2 = 6

#swap variable formulas
a2 = a2 + b2
b2 = a2 - b2
a2 = a2 - b2
```

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

```
6
5
```

```
print(0b101) # 101 is 3 bit
print(0b110) # 110 also 3bit
```

```
5
6
```

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

```
0b1011
11
```

```
#there is other way to work using swap variable also which is XOR
because it will not waste extra bit
a2 = a2 ^ b2
b2 = a2 ^ b2
a2 = a2 ^ b2

print(a2)
print(b2)
```

```
5
6
```

```
a2 , b2 = b2, a2
```

BITWISE OPERATOR

- WE HAVE 6 OPERATORS COMPLEMENT (~) || AND (&) || OR (|) || XOR (^) || LEFT SHIFT (<<) || RIGHT SHIFT (>>)

```
print(bin(12))
print(bin(13))
```

```
0b1100
0b1101
```

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 image.png

```
~12

-13

~-1

0
```

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

image.png

```
12 & 13

12

1 & 1

1

1 | 0

1

12 | 13

13

1 & 0

0
```

```
35 & 40 #please do the homework conververt 35,40 to binary format
```

```
32
```

```
print(bin(35))
```

```
0b100011
```

```
print(bin(40))
```

```
0b101000
```

```
35 | 40
```

```
43
```

```
12 ^ 13
```

```
1
```

```
int(0b000111)
```

```
7
```

```
10>>2
```

```
2
```

```
bin(20)
```

```
'0b10100'
```

import math module

```
x = sqrt(25) #sqrt is inbuild function
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[140], line 1
----> 1 x = sqrt(25)

NameError: name 'sqrt' is not defined
```

```
import math
```

```
x = math.sqrt(25)
x
```

```
5.0
```

```
x1 = math.sqrt(15)
x1
```

```
3.872983346207417
```

```
print(math.floor(2.9)) #floor - minimum or least value
```

```
2
```

```
print(math.ceil(2.9)) #ceil - maximum or highest value
```

```
3
```

```
print(math.pow(3,2))
```

```
9.0
```

```
print(math.pi) #these are constant
```

```
3.141592653589793
```

```
import math as m
m.sqrt(10)
```

```
3.1622776601683795
```

```
from math import sqrt,pow # math has many function if you want to call
specific function then you use from
pow(2,3)
```

```
8.0
```

```
from math import * # math has many function if you want to call
specific function then you use from

print(pow(2,3))
print(floor(2.3))
```

```
8.0
2
```

```
round(pow(2,3))
```

```
8
```

```
# pycharm run debug
# how to install python idle
# how to install pycharm & starts working on pycharm

### user input function in python || command line input

x1 = input('Enter the 1st number') #whenevery you works in input
function it always give you string
y1 = input('Enter the 2nd number') # it wont understand as arithmetic
operator
z1 = x1 + y1
print(z1)
```

```
Enter the 1st number 1
Enter the 2nd number 1

11
```

```
x1 = input('Enter the 1st number') #whenevery you works in input
function it always give you string
a1 = int(x1)
y1 = input('Enter the 2nd number') # it wont understand as arithmetic
operator
b1 = int(y1)
z1 = a1 + b1
print(z1)
```

```
Enter the 1st number 1
Enter the 2nd number 1

2
```

```
ch = input('enter a char')
print(ch)
```

```
enter a char dfnc

dfnc
```

```
print(ch[0])
```

```
d
```

```
print(ch[-1])
```

```
c
```

```
ch = input('enter a char')[0]
print(ch)
```

```
enter a char efrgbfvcdsxza

e
```

```
ch = input('enter a char')[1:3]
print(ch)
```

```
enter a char wdefs

de
```

```
result = eval(input('enter an expr'))
print(result)
```

```
enter an expr 3+4

7
```