operator's in python

1- ARITHMETIC OPERATOR (+ , -, *, /, %, %%, **, ^ 2- ASSIGNMEN OPERATOR (=) 3- RELATIONAL OPERATOR 4- LOGICAL OPERATOR 5- UNARY OPERATOR

Arithmetic operator

```
In [3]: x1, y1 = 10, 5
```

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

```
Out[4]: 15
```

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

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

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

```
Out[6]: 50
```

```
In [7]: x1 / y1
```

```
Out[7]: 2.0
```

```
In [8]: x1 // y1
```

```
Out[8]: 2
```

```
In [9]: x1 % y1
```

```
Out[9]: 0
```

```
In [10]: x1 ** y1
```

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

```
In [11]: 2 ** 3
```

```
Out[11]: 8
```

Assignment operator

```
In [13]: x = 2
```

```
In [14]: x = x + 2
```

```
In [15]: x
```

```
Out[15]: 4
```

```
In [16]: x += 2
```

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

```
Out[17]: 6
```

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

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

```
Out[19]: 8
```

```
In [20]: x *= 2
```

```
In [21]: x
```

```
Out[21]: 16
```

```
In [22]: x /= 2
```

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

```
Out[23]: 8.0
```

```
In [24]: a, b = 5,6
```

```
In [25]: a
```

```
Out[25]: 5
```

```
In [26]: b
```

```
Out[26]: 6
```

unary operator

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 [28]: n = 7 #negattion
```

```
In [29]: m = -(n)
```

```
In [30]: m
```

```
Out[30]: -7
```

```
In [31]: n
```

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

```
In [32]: -n
```

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

Relational operator

we are using this operator for comparing

```
In [33]:  a = 5
          b = 7
```

```
In [34]:  a == b
Out[34]:  False
```

```
In [35]:  a<b
Out[35]:  True
```

```
In [36]:  a>b
Out[36]:  False
```

```
In [37]:  # a = b # we cannot use = operatro that means it is assigning
```

```
In [38]:  a == b
Out[38]:  False
```

```
In [39]:  a = 10
```

```
In [40]:  a != b
Out[40]:  True
```

```
In [41]:  # hear if i change b = 6
          b = 10
```

```
In [42]:  a == b
Out[42]:  True
```

```
In [43]:  a >= b
Out[43]:  True
```

```
In [44]:  a <= b
Out[44]:  True
```

```
In [45]:  a < b
Out[45]:  False
```

```
In [46]:  a>b
```

```
Out[46]:  False
```

```
In [47]:  b = 7
```

```
In [48]:  a != b
```

```
Out[48]:  True
```

LOGICAL OPERATOR

AND, OR, NOT

```
In [49]:  a = 5
          b = 4
```

```
In [50]:  a < 8 and b < 5 #refer to the truth table
```

```
Out[50]:  True
```

```
In [51]:  a < 8 and b < 2
```

```
Out[51]:  False
```

```
In [52]:  a < 8 or b < 2
```

```
Out[52]:  True
```

```
In [53]:  a>8 or b<2
```

```
Out[53]:  False
```

```
In [54]:  x = False
          x
```

```
Out[54]:  False
```

```
In [55]:  not x  # you can reverse the operation
```

```
Out[55]:  True
```

```
In [56]:  x = not x
          x
```

```
Out[56]:  True
```

```
In [57]:  x
```

```
Out[57]:  True
```

```
In [58]:  not x
```

```
Out[58]:  False
```

Number system coverstion (bit-binary digit)

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

```
In [59]: 25
Out[59]: 25

In [60]: bin(25)
Out[60]: '0b11001'

In [61]: 0b11001
Out[61]: 25

In [62]: int(0b11001)
Out[62]: 25

In [63]: bin(35)
Out[63]: '0b100011'

In [64]: int(0b100011)
Out[64]: 35

In [65]: bin(20)
Out[65]: '0b10100'

In [66]: int(0b10100)
Out[66]: 20

In [67]: 0b1111
Out[67]: 15

In [68]: oct(15)
Out[68]: '0o17'

In [69]: 0o17
Out[69]: 15

In [70]: hex(9)
Out[70]: '0x9'
```

```
In [71]: 0xf

Out[71]: 15

In [72]: hex(10)

Out[72]: '0xa'

In [73]: 0xa

Out[73]: 10

In [74]: hex(25)

Out[74]: '0x19'

In [75]: 0x19

Out[75]: 25

In [76]: 0x15

Out[76]: 21
```

swap variable in python

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

```
In [77]: a = 5
         b = 6

In [78]: a = b
         b = a

In [79]: a,b = b,a

In [80]: print(a)
         print(b)

6
6

In [81]: # in above scenario we lost the value 5
         a1 = 7
         b1 = 8

In [82]: temp = a1
         a1 = b1
         b1 = temp

In [83]: print(a1)
         print(b1)

8
7
```

```
In [84]: a2 = 5
         b2 = 6
```

```
In [85]: #swap variable formulas
         a2 = a2 + b2
         b2 = a2 - b2
         a2 = a2 - b2
```

```
In [86]: print(a2)
         print(b2)
```

```
6
5
```

```
In [87]: print(0b101) # 101 is 3 bit
         print(0b110) # 110 also 3bit
```

```
5
6
```

```
In [88]: #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 [89]: #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 [90]: print(a2)
         print(b2)
```

```
5
6
```

```
In [91]: print(a2)
         print(b2)
```

```
5
6
```

```
In [92]: a2 , b2 = b2, a2
```

```
In [93]: print(a2)
         print(b2)
```

```
6
5
```

BITWISE OPERATOR

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

```
In [94]: 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 con cept (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

COMPLEMENT (~) (TILDE OR TILD)

~12 # why we get -13 . first we understand what is complment means (reversr of binary format)

```
In [96]: ~45
Out[96]: -46

In [97]: ~6
Out[97]: -7

In [98]: ~-6
Out[98]: 5

In [99]: ~-1
Out[99]: 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

```
In [100…: 12 & 13
```

```
Out[100… 12

In [101… 1 & 1

Out[101… 1

In [102… 1 | 0

Out[102… 1

In [103… 1 & 0

Out[103… 0

In [104… 12 | 13

Out[104… 13

In [105… 35 & 40 #please do the homework conververt 35,40 to binary format

Out[105… 32

In [106… 35 | 40

Out[106… 43

In [107… # in XOR if the both number are different then we will get 1 or else we will get

         12 ^ 13

Out[107… 1

In [108… 25 ^ 30

Out[108… 7

In [109… bin(25)

Out[109… '0b11001'

In [110… bin(30)

Out[110… '0b11110'

In [111… int(0b000111)

Out[111… 7
```

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 [112…  20<<4 #can we do this

Out[112…  320
```

BITWISE RIGHTSHIFT OPERATOR

```
In [113…  10>>2

Out[113…  2

In [114…  bin(20)

Out[114…  '0b10100'

In [115…  20>>4

Out[115…  1
```

import math module

```
In [116…  x = sqrt(25) #sqrt is inbuild function

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[116], line 1
----> 1 x = sqrt(25)

NameError: name 'sqrt' is not defined

In [117…  import math # math is module

In [118…  x = math.sqrt(25)
          x

Out[118…  5.0

In [119…  x1 = math.sqrt(15)
          x1

Out[119…  3.872983346207417

In [121…  print(math.floor(2.9)) #floor - minimum or least value

         2

In [122…  print(math.ceil(2.9)) #ceil - maximum or highest value
```

```
3
```

In [123…
```
print(math.pow(3,2))
```

```
9.0
```

In [124…
```
print(math.pi) #these are constant
```

```
3.141592653589793
```

In [125…
```
print(math.e) #these are constant
```

```
2.718281828459045
```

In [126…
```
import math as m
m.sqrt(10)
```

Out[126…
```
3.1622776601683795
```

In [127…
```
from math import sqrt,pow # math has many function if you want to call specific
pow(2,3)
```

Out[127…
```
8.0
```

In [128…
```
round(pow(2,3))
```

Out[128…
```
8
```

In [129…
```
#help(math)
```

In [130…
```
# pycharm run debug
# how to install python idle
# how to install pycharm & starts working on pycharm
```

user input function in python || command line input

In [ ]:
```
x = input()
y = input()
z = x + y
print(z) # console is waiting for user to enter input
# also if you work in idle
```

In [ ]:
```
x1 = input('Enter the 1st number') #whenevery you works in input function it alw
y1 = input('Enter the 2nd number') # it wont understand as arithmetic operator
z1 = x1 + y1
print(z1)
```

In [ ]:
```
type(x1)
type(y1)
```

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

- for the above code notice we are using many lines because fo that wasting some memory spaces as well

In [ ]:
```
x2 = int(input('Enter the 1st number'))
y2 = int(input('Enter the 2nd number'))
z2 = x2 + y2
z2
```

In [ ]:
```
```