operator's in python

In [ ]:
```
# 1- ARITHMETIC OPERATOR ( + , -, *, /, %, %%, **, ^
# 2- ASSIGNMEN OPERATOR (=)
# 3- RELATIONAL OPERATOR
# 4- LOGICAL OPERATOR
# 5- UNARY OPERATOR
```

Arithmetic operator

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

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

Out[2]: 15

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

Out[3]: 5

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

Out[4]: 50

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

Out[5]: 2.0

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

Out[6]: 2

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

Out[7]: 0

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

Out[8]: 100000

In [9]:
```
2 ** 3
```

Out[9]: 8

Assignment operator

In [10]:
```
x = 2
```

In [11]:
```
x = x + 2
```

In [12]:
```
x
```

Out[12]: 4

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

In [14]:
```
x
```

Out[14]: 6

In [15]:
```
x *= 2
```

In [16]:
```
x
```

Out[16]: 12

In [17]:
```
x -= 2
```

In [18]:
```
x
```

Out[18]: 10

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

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

Out[20]: 5.0

In [21]:
```
a, b = 5,6
```

In [22]:
```
a
```

Out[22]: 5

In [23]:
```
b
```

Out[23]: 6

unary operator

In [24]:
```
n = 7 #negattion
```

In [25]:
```
m = -(n)
```

In [26]:
```
m
```

Out[26]: -7

In [27]:
```
n
```

Out[27]: 7

In [28]:
```
-n
```

Out[28]: -7

Relational operator

In [29]:
```
a = 5
b = 7
```

In [30]:
```
a == b
```

Out[30]: False

In [31]:
```
a<b
```

Out[31]: True

In [32]:
```
a>b
```

Out[32]: False

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

Out[33]: False

In [34]:
```
a = 10
```

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

Out[35]: True

In [36]:
```
# hear if i change b = 6
b = 10
```

In [37]:
```
a == b
```

Out[37]: True

In [38]:
```
a >= b
```

Out[38]: True

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

Out[39]: True

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

Out[40]: False

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

Out[41]: False

In [42]:
```
b = 7
```

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

Out[43]: True

LOGICAL OPERATOR

In [45]:
```
a = 5
b = 4
```

In [46]:
```
a < 8 and b < 5 #refer to the truth table
```

Out[46]: True

In [47]:
```
a < 8 and b < 2
```

Out[47]: False

In [48]:
```
a < 8 or b < 2
```

Out[48]: True

In [49]:
```
a>8 or b<2
```

Out[49]: False

In [50]:
```
x = False
x
```

Out[50]: False

In [51]:
```
not x  # you can reverse the operation
```

Out[51]: True

In [52]:
```
x = not x
x
```

Out[52]: True

In [53]:
```
x
```

Out[53]: True

In [54]:
```
not x
```

Out[54]: False

Number system coverstion (bit-binary digit)

In [55]:
```
# 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 [56]:
```
25
```

Out[56]: 25

```
In [57]: bin(25)
Out[57]: '0b11001'
In [58]: 0b11001
Out[58]: 25
In [60]: int(0b11001)
Out[60]: 25
In [61]: bin(35)
Out[61]: '0b100011'
In [62]: int(0b100011)
Out[62]: 35
In [63]: bin(20)
Out[63]: '0b10100'
In [64]: int(0b10100)
Out[64]: 20
In [65]: 0b1111
Out[65]: 15
In [66]: oct(15)
Out[66]: '0o17'
In [67]: 0o17
```

Out[67]: 15

In [68]:
```
hex(9)
```

Out[68]: '0x9'

In [69]:
```
0xf
```

Out[69]: 15

In [70]:
```
hex(10)
```

Out[70]: '0xa'

In [71]:
```
0xa
```

Out[71]: 10

In [72]:
```
hex(25)
```

Out[72]: '0x19'

In [73]:
```
0x19
```

Out[73]: 25

In [74]:
```
0x15
```

Out[74]: 21

swap variable in python

In [75]:
```
a = 5
b = 6
```

In [76]:
```
a = b
b = a
```

In [77]:
```
a,b = b,a
```

In [78]:
```
print(a)
print(b)
```

```
6
6
```

In [79]:
```
# in above scenario we lost the value 5
a1 = 7
b1 = 8
```

In [80]:
```
temp = a1
a1 = b1
b1 = temp
```

In [81]:
```
print(a1)
print(b1)
```

```
8
7
```

In [82]:
```
a2 = 5
b2 = 6
```

In [83]:
```
#swap variable formulas
a2 = a2 + b2
b2 = a2 - b2
a2 = a2 - b2
```

In [84]:
```
print(a2)
print(b2)
```

```
6
5
```

In [85]:
```
print(0b101) # 101 is 3 bit
```

```
print(0b110) # 110 also 3bit
```

```
5
6
```

In [86]:
```
#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
```

In [87]:
```
#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
```

In [88]:
```
print(a2)
print(b2)
```

```
5
6
```

In [89]:
```
a2 , b2 = b2, a2
```

In [90]:
```
print(a2)
print(b2)
```

```
6
5
```

BITWISE OPERATOR

In [ ]:
```
# WE HAVE 6 OPERATORS
# COMPLEMENT ( ~ ) || AND ( & )  || OR ( | ) || XOR ( ^ ) || LEFT SHIFT ( << ) || RIGHT SHIFT ( >> )|
```

In [91]:
```
print(bin(12))
print(bin(13))
```

```
0b1100
0b1101
```

In [92]:
```
# complement --> you will get this key below esc character
```

In [93]:
```
# COMPLEMENT (~) (TILDE  OR TILD)
~12 # why we get -13 . first we understand what is complment means (reversr of binary format)
```

Out[93]: -13

In [94]:
```
~45
```

Out[94]: -46

In [95]:
```
~6
```

Out[95]: -7

In [96]:
```
~-6
```

Out[96]: 5

In [97]:
```
~-1
```

Out[97]: 0

In [98]:
```
# bit wise and operator
```

In [99]:
```
# 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 0
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

In [112…
```
# 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
```

Out[112… 40

In [113…
```
20<<4 #can we do this
```

Out[113… 320

In [114…
```
# BITWISE RIGHTSHIFT OPERATOR
```

In [115…
```
10>>2
```

Out[115… 2

In [116…
```
bin(20)
```

Out[116… '0b10100'

In [117…
```
20>>4
```

Out[117… 1

import math module

In [118…
```
x = sqrt(25) #sqrt is inbuild function
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[118], line 1
----> 1 x = sqrt(25)

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

In [119…
```
import math # math is module
```

In [120…
```
x = math.sqrt(25)
x
```

Out[120… 5.0

In [121…
```
x1 = math.sqrt(15)
x1
```

Out[121… 3.872983346207417

In [122…
```
print(math.floor(2.9)) #floor - minimum or least value
```

2

In [123…
```
print(math.ceil(2.9)) #ceil - maximum or highest value
```

3

In [124…
```
print(math.pow(3,2))
```

9.0

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

3.141592653589793

In [126…
```
print(math.e) #these are constant
```

2.718281828459045

In [127…
```
import math as m
m.sqrt(10)
```

Out[127… 3.1622776601683795

In [128…
```
from math import sqrt,pow # math has many function if you want to call specific function then you use from
pow(2,3)
```

Out[128… 8.0

In [129…

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

Out[129…

```
8
```