Operator in Python

```
#1.arithmetic(completed)
#2. assingment
#3. comparison
#4. logical
#5. bitwise
```

[ ] Python

```
#Assignment Operator
x = 2
print(x)
x += 2
print(x)
x -=2
print(x)
x *=2
print(x)
```

[50] ✓ 0.0s Python

```
2
4
2
4
```

```
#Unary Operator
n =  7
n
```

[53] ✓ 0.0s Python

```
7
```

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

[55] ✓ 0.0s Python

```
-7
7
-7
```

```
#Relational Operator
a = 5
b = 6
```

[56] ✓ 0.0s Python

```
print(a > b)
print(a<b)
print(a == b)
print(a != b)
```

[57] ✓ 0.0s Python

```
False
True
False
True
```

```
b = 5
```

[17] Python

```
a == b
```

[18] Python

```
True
```

```
print(a > b)
print(a>= b)
print(a<= b)
print(a< b)
```

[22] Python

```
False
False
True
True
```

```
b =7
```

[23] Python

```
a != b
```

[21] Python

```
True
```

```
#Logical Operator
#truth table
#And
#1 - 1 =1
#0 - 1 = 0
#1 - 0 = 0
#0 - 0 = 0

#Or
#1 - 1 =1
#0 - 1 = 1
#1 - 0 = 1
#0 - 0 = 0
```

[ ] Python

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

[58] ✓ 0.0s Python

```
print(a < 8 and b < 5)
print(a < 8 and b < 2)
print(a < 8 or b < 2)
print(a > 8 or b < 2)
```

[60] ✓ 0.0s Python

```
True
False
True
False
```

```
x = False
x
```

[29] Python

```
False
```

```
not x
```

[30] Python

```
True
```

```
#Number System Conversion
#binary-write as 0b(base 2)
#octal - write as 0o(base 8)
#decimal - Write as 0x(base 10)
#hexdecimal - write as 0x(base 16)
```

[ ] Python

```
25
```

[31] Python

```
25
```

```
print(bin(25))
print(int(0b11001))
```

[61] ✓ 0.0s Python

```
0b11001
25
```

```
print(bin(30))
print(int(0b11110))
```

[38] Python

```
0b11110
30
```

```
print(oct(25))
print(int(0o31))
print(hex(25))
print(int(0x19))
```

[41] Python

```
0o31
25
0x19
25
```

```python
print(int(0o31))
print(bin(7))
print(int(0x19))
print(hex(16))
print(int(0xa))
print(int(0x19))
print(int(0x15))
```

[62] ✓ 0.0s Python

```
25
0b111
25
0x10
10
25
21
```

```python
#Swapping Values
#Method 1(ROT Two)
a = 5
b = 6
print("a : ",a)
print("b : ", b)
a , b = b, a
print("Swapped Values")
print("a : ",a)
print("b : ", b)
```

[ ] Python

```
a :  5
b :  6
Swapped Values
a :  6
b :  5
```

```python
#method 2
a1 = 4
b1 = 5
temp = a1
a1 = b1
b1 = temp
print(a1)
print(b1)
```

[ ] Python

```
5
4
```

```python
a3 = 10
b3 = 20
a3 = a3 + b3
b3 = a3 - b3
a3 = a3 - b3
print(a3)
print(b3)
```

[58] Python

```
20
10
```

```python
#Bitwise Operator
#AND - &
#OR - |
#XOR - ^(exclusive or )
#NOT - ~(Complement)
#left shift - <<
#right shift - >>
```

[ ] Python

```python
#Complement
print(~12)
print(~46)
print(~54)
print(~10)
```

[63] ✓ 0.0s Python

```
-13
-47
-55
-11
```

```python
#and - & and or -|
print(12 & 13)
print(12 | 13)
print(1 & 0)
print(1 | 0)
print(35 & 40)
print(35 & 40)
print(35 | 40)
```

[65] ✓ 0.0s Python

```
12
13
0
1
32
32
43
```

```python
#XOR
# 0 and 0 = 0
# 0 and 1 = 1
# 1 and 0 = 1
# 1 and 1 = 0
35 ^ 40
```

[2] ✓ 0.0s Python

```
11
```

```python
print(12 ^ 13)
```

[3] ✓ 0.0s Python

```
1
```

```python
print(bin(25))
print(bin(30))
```

[4] ✓ 0.0s Python

```
0b11001
0b11110
```

```python
print(25 ^ 30)
print(bin(7))
```

[66] ✓ 0.0s Python

```
7
0b111
```

```python
#Left shift - <<(gain the bit)
print(15<< 2)
print(10<< 1)
print(10<< 2)
```

[14] ✓ 0.0s Python

```
60
20
40
```

```
print(bin(15))
print(bin(10))
```
[11] ✓ 0.0s Python

```
0b1111
0b1010
```

```
print(int(0b111100))
print(int(0b10100))
print(int(0b101000))
```
[15] ✓ 0.0s Python

```
60
20
40
```

```
#right shift - >>(lose the bit)
bin(10)
```
[16] ✓ 0.0s Python

```
'0b1010'
```

Generate + Code + Markdown

```
print(10 >> 1)
print( 10 >> 2)
print(10>> 3)
```
[19] ✓ 0.0s Python

```
5
2
1
```

```
print(bin(20))
print(20 >> 3)
```
[20] ✓ 0.0s Python

```
0b10100
2
```

```
#Packages/Library/Framwork = collection of module
#Module = collection of functions
#Function = collection of statements(inbuild and user define func
```
[ ] Python

```
#math
import math
x = math.sqrt(25)
print(x)
x1 = math.sqrt(5)
print(x1)
```
[26] ✓ 0.0s Python

```
5.0
2.23606797749979
```

```
print(math.floor(3.87))#minimun or least value
print(math.ceil(3.87))#maximum or highest value
```
[25] ✓ 0.0s Python

```
3
4
```

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

[28] ✓ 0.0s Python

```
8.0
9.0
```

```
math.pi
```

[29] ✓ 0.0s Python

```
3.141592653589793
```

```
math.e
```

[30] ✓ 0.0s Python

```
2.718281828459045
```

```
from math import sqrt, floor, ceil, pow, pi, e
print(pow(2,3))
print(sqrt(30))
print(floor(45.85))
print(ceil(45.85))
```

[31] ✓ 0.0s Python

```
8.0
5.477225575051661
45
46
```

✓ 0s [4]
```
print(type(x))
print(type(y))
```

```
<class 'int'>
<class 'int'>
```

✓ 6s [5]
```
str = input("Enter a string:")
print(str)
```

```
Enter a string:hello
hello
```

✓ 0s [6]
```
print(str[0])
print(str[1])
```

```
h
e
```

✓ 6s [7]
```
st = input("Enter a string:")[1]
st
```

```
Enter a string:hello
'e'
```

```
#input
x = input("Enter first value: ")
y = input("Enter second value: ")
z = x + y
print(z) # input function by default takes input as string
```

```
Enter first value: 5
Enter second value: 6
56
```

```
[2] print(type(x))
    print(type(y))
```

```
<class 'str'>
<class 'str'>
```

```
[3] #input
    x = int(input("Enter first value: "))
    y = int(input("Enter second value: "))
    z = x + y
    print(z)
```

```
Enter first value: 5
Enter second value: 8
13
```

```
[8] st1 = input("Enter a string:")[5:8]
    st1
```

```
Enter a string:nareshit
'hit'
```

```
[9] result = eval(input("Enter a expression: "))
    print(result)
```

```
Enter a expression: 10+20-4*7
2
```