Arithmetic Operator

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

In [2]: x1

Out[2]: 10

In [3]: y1

Out[3]: 5

In [4]: #x1^y1

In [5]: x1 + y1

Out[5]: 15

In [6]: x1 - y1

Out[6]: 5

In [7]: x1 * y1

Out[7]: 50

In [8]: x1 / y1 #float division

Out[8]: 2.0

In [9]: x1 // y1 #int Division

Out[9]: 2

In [10]: x1 % y1

Out[10]: 0

In [11]: x1 ** y1

Out[11]: 100000

In [12]: 3 ** 2

Out[12]: 9

In [14]: 3 * 2
```

```
Out[14]:  6
```

```
In [15]:  x2 = 3
          y2 = 3

          x2 ** y2
```

```
Out[15]:  27
```

Assignment Operator

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

```
In [17]:  x = x + 2 # if you want to increment by 2
```

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

```
Out[18]:  4
```

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

```
Out[19]:  6
```

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

```
Out[20]:  12
```

```
In [21]:  x -= 2
          x
```

```
Out[21]:  10
```

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

```
Out[22]:  5.0
```

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

```
Out[23]:  2.0
```

```
In [24]:  a, b = 5,6 # you can assigned varible in one line as well
          print(a)
          print(b)
```

```
5
6
```

```
In [25]: a
Out[25]: 5
```

```
In [26]: b
Out[26]: 6
```

```
In [27]: a = 5
         b = 6
         print(a)
         print(b)
5
6
```

```
In [28]: a
Out[28]: 5
```

```
In [29]: b
Out[29]: 6
```

Unary Operator

```
In [30]: n = 7
         n
Out[30]: 7
```

```
In [31]: m = -(n)
         m
Out[31]: -7
```

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

```
In [33]: -n
Out[33]: -7
```

Relational Operator

we are useing this operator for comparing

```
In [34]: a = 5
         b = 6

In [35]: a<b

Out[35]: True

In [36]: a>b

Out[36]: False

In [38]: # a = b we cannot use = operator that means it is assiging

In [39]: a == b

Out[39]: False

In [40]: a != b

Out[40]: True

In [42]: # hear if i change b = 6
         b = 5

In [43]: a == b

Out[43]: True

In [44]: a

Out[44]: 5

In [45]: b

Out[45]: 5

In [46]: a > b

Out[46]: False

In [48]: a >= b

Out[48]: True

In [49]: a <= b

Out[49]: True

In [50]: a< b

Out[50]: False
```

```
In [51]: a > b

Out[51]: False

In [55]: b = 7

In [53]: b

Out[53]: 7

In [54]: a != b

Out[54]: True
```

Logical Operator

```
In [56]: a = 5
         b = 4

In [57]: a < 8 and b < 5

Out[57]: True

In [58]: a < 8 and b < 2

Out[58]: False

In [59]: a < 8 or b<2

Out[59]: True

In [60]: a>8 or b<2

Out[60]: False

In [61]: x = False
         x

Out[61]: False

In [62]: not x

Out[62]: True

In [63]: x = not x
         x

Out[63]: True
```

```
In [64]: x
Out[64]: True

In [65]: not x
Out[65]: False
```

Number System Converstion (bit - binary digit)

```
In [1]: 25
Out[1]: 25

In [2]: bin(25)
Out[2]: '0b11001'

In [3]: int(0b11001)
Out[3]: 25

In [4]: bin(30)
Out[4]: '0b11110'

In [5]: int(0b11001)
Out[5]: 25

In [6]: oct(25)
Out[6]: '0o31'

In [10]: int(0o31)
Out[10]: 25

In [9]: int(0b11110)
Out[9]: 30

In [11]: 0o31
Out[11]: 25

In [12]: 0b11001
```

```
Out[12]: 25
```

```
In [13]: int(0b11001)
```

```
Out[13]: 25
```

```
In [14]: bin(7)
```

```
Out[14]: '0b111'
```

```
In [15]: oct(25)
```

```
Out[15]: '0o31'
```

```
In [16]: int(0o31)
```

```
Out[16]: 25
```

```
In [17]: hex(25)
```

```
Out[17]: '0x19'
```

```
In [18]: hex(16)
```

```
Out[18]: '0x10'
```

```
In [20]: 0xa
```

```
Out[20]: 10
```

```
In [21]: 0xb
```

```
Out[21]: 11
```

```
In [22]: hex(1)
```

```
Out[22]: '0x1'
```

```
In [23]: 0x19
```

```
Out[23]: 25
```

```
In [24]: 0x15
```

```
Out[24]: 21
```

Swap 2-variable in python

```
In [40]: a = 4
         a
```

```
Out[40]:  4
```

```
In [41]: b = 8
         b
```

```
Out[41]:  8
```

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

```
In [43]: print(a)
```

```
8
```

```
In [44]: print(b)
```

```
8
```

```
In [45]: print(a)
         print(b)
```

```
8
8
```

```
In [46]: a1 = 7
         b1 = 8
```

```
In [47]: temp = a1
         a1 = b1
         b1 = temp
```

```
In [48]: print(a1)
         print(b1)
```

```
8
7
```

```
In [49]: a2 = 5
         b2 = 6
```

```
In [50]: # swap variable formal without using 3rd formul
         a2 = a2 + b2 # 5+6 = 11
         b2 = a2 - b2 # 11-6 = 5
         a2 = a2 - b2 # 11-5 = 6
```

```
In [51]: print(a2)
         print(b2)
```

```
6
5
```

```
In [52]: 0b110
```

```
Out[52]:  6
```

```
In [53]: 0b101
Out[53]: 5
```

```
In [54]: print(0b110)
         print(0b101)

6
5
```

```
In [55]: print(0b101)
         print(0b110)

5
6
```

```
In [57]: print(0b1011)

11
```

XOR

```
In [59]: print(a2)
         print(b2)

6
5
```

```
In [60]: # there is other way to work using swap variable also which is XOR becuse it will
         a2 = a2 ^ b2
         b2 = a2 ^ b2
         a2 = a2 ^ b2
```

```
In [61]: print(a2)
         print(b2)

5
6
```

```
In [62]: a2, b2
Out[62]: (5, 6)
```

```
In [64]: a2 , b2 = b2, a2 # how it work is b2 6 a2 is 5 first it goes into stack & then it
```

```
In [65]: print(a2)
         print(b2)

5
6
```

Bitwise Operator

1. Complement (~)

2. And (&)

3. Or (|)

4. XOR (^)

5. Lift Shift (<<)

6. Right Shift (>>)

In [66]:
```
print(bin(12))
print(bin(13))
```
```
0b1100
0b1101
```

In [67]:
```
0b1100
```
Out[67]:
```
12
```

In [68]:
```
0b1101
```
Out[68]:
```
13
```

In [69]:
```
# Complement (~) (TILDE) OR (TILD)
~12
```
Out[69]:
```
-13
```

In [70]:
```
~46
```
Out[70]:
```
-47
```

In [71]:
```
~54
```
Out[71]:
```
-55
```

In [72]:
```
~10
```
Out[72]:
```
-11
```

```
In [73]: # AND & Operator
         12 & 13

Out[73]: 12

In [74]: 12 | 13

Out[74]: 13

In [75]: 1 & 0

Out[75]: 0

In [76]: 1 | 0

Out[76]: 1

In [77]: bin(13)

Out[77]: '0b1101'

In [78]: print(bin(35))
         print(bin(40))

        0b100011
        0b101000

In [79]: 35 & 40

Out[79]: 32

In [80]: 35 | 40

Out[80]: 43

In [81]: 12 ^ 13

Out[81]: 1

In [82]: print(bin(25))
         print(bin(30))

        0b11001
        0b11110

In [83]: 25 ^ 30

Out[83]: 7

In [84]: bin(7)

Out[84]: '0b111'
```

```
In [85]: bin(25)

Out[85]: '0b11001'

In [86]: bin(30)

Out[86]: '0b11110'

In [87]: 0b00111

Out[87]: 7

In [88]: bin(10)

Out[88]: '0b1010'

In [89]: 10<<1

Out[89]: 20

In [90]: 10<<2

Out[90]: 40

In [91]: bin(10)

Out[91]: '0b1010'

In [92]: 10<<1

Out[92]: 20

In [93]: 10<<2

Out[93]: 40

In [94]: 10<<3

Out[94]: 80

In [95]: bin(20)

Out[95]: '0b10100'

In [96]: 20<<4

Out[96]: 320

In [97]: 10>>1

Out[97]: 5
```

```
In [98]: 10>>2
```

```
Out[98]: 2
```

```
In [ ]: 10>>3
```