Python Operator

- Arithmetic Operators
- Assignment Operators
- Relational Operators
- Logical Operators
- Unary Operator

Arithmetic Operator

In [6]:
```
x1, y1 = 100, 50

print(x1 + y1)
```
```
150
```

In [7]:
```
x1 - y1
```
Out[7]:
```
50
```

In [8]:
```
x1 * y1
```
Out[8]:
```
5000
```

In [9]:
```
x1 / y1 # float division
```
Out[9]:
```
2.0
```

In [10]:
```
x1 // y1 # int division
```
Out[10]:
```
2
```

In [11]:
```
x1 % y1
```

```
Out[11]: 0
```

```
In [12]: x1 ** y1
```

```
Out[12]: 10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

Assignment Operator

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

```
Out[14]: 4
```

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

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

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

```
Out[17]: 14
```

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

```
Out[18]: 7.0
```

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

```
Out[19]: 3.0
```

```
In [20]: a, b = 5,6
         print(a, b)
```

```
5 6
```

Unary Operator

- Unary means 1 where as binary means 2
- Here we are applying minus operator (-) on the operand n; the value of m becomes -7, which indicates it as a negative vlue.

```
In [21]: n =7
         n
```

```
Out[21]: 7
```

```
In [22]: m = -n
         m
```

```
Out[22]: -7
```

```
In [23]: print(n, -n)
```

```
7 -7
```

Relational Operator

We aer using this operator for comapring

```
In [27]: a, b = 100, 50
```

```
In [28]: a < b
```

```
Out[28]: False
```

```
In [29]: a> b
```

```
Out[29]: True
```

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

Out[30]:
```
False
```

In [31]:
```
a != b
```

Out[31]:
```
True
```

In [33]:
```
b = 100 # changing b = 100
```

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

Out[34]:
```
True
```

In [35]:
```
a > b
```

Out[35]:
```
False
```

In [36]:
```
a >= b
```

Out[36]:
```
True
```

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

Out[37]:
```
True
```

Logical Operator

- Logical operator you need understand about true & false table

1. And
2. Or
3. Not

In [38]:
```
a= 100
b = 50
```

`Logical And Operator`

```
In [39]: a < 150 and b < 100
```

```
Out[39]: True
```

```
In [40]: a < 150 and b < 49
```

```
Out[40]: False
```

`Logical Or Operator`

```
In [41]: a < 150 or b < 40
```

```
Out[41]: True
```

```
In [42]: a > 150 or b < 40
```

```
Out[42]: False
```

`Logical Not Operator`

```
In [43]: x = False
         x
```

```
Out[43]: False
```

```
In [44]: not x
```

```
Out[44]: True
```

Number System conversion (bit-binary digit)

- Binary Number Systems (0b) -> base 2 (0, 1)
- Octal Number Systems (0o) -> base 8 (0, 1, 2, 3, 4, 5, 6, 7)
- Decimal Number System (0x) -> base 10 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

- Hexadecimal Number System(0xa, b, c, d, e, f) -> base 16 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a(10), b(11), c(12), d(13), e(14), f(15))

`Binary Number System`

```
In [45]: bin(25)

Out[45]: '0b11001'

In [46]: int(0b11001)

Out[46]: 25

In [47]: bin(30)

Out[47]: '0b11110'

In [48]: int(0b11110)

Out[48]: 30

In [50]: int(0b11101)

Out[50]: 29
```

`Octal Number System`

```
In [52]: oct(25)

Out[52]: '0o31'

In [53]: int(0o31)

Out[53]: 25

In [54]: oct(77)

Out[54]: '0o115'
```

```
In [55]: int(0o115)
```

```
Out[55]: 77
```

`Hexa Decimal Number System`

```
In [56]: hex(25)
```

```
Out[56]: '0x19'
```

```
In [57]: 0x19
```

```
Out[57]: 25
```

```
In [58]: hex(10)
```

```
Out[58]: '0xa'
```

```
In [59]: int(0xa)
```

```
Out[59]: 10
```

```
In [60]: hex(1)
```

```
Out[60]: '0x1'
```

BITWISE OPERATOR

- Complement Operator (~)
- AND Operator (&)
- OR Operator (|)
- XOR Operator (^)
- Left Shift Operator (<<)
- Right Shift Operator (>>)

Complement Operator (~)

```
In [61]: ~ 12 # Complement means it stores -ve values
```

```
Out[61]: -13
```

```
In [62]: ~1007
```

```
Out[62]: -1008
```

AND Operator (&)

```
In [63]: 100 & 101
```

```
Out[63]: 100
```

```
In [64]: print(bin(100), bin(101))
```

```
0b1100100 0b1100101
```

- 0b1100100 & 0b1100101 -> 0b1100100 (These are all acording to the truth table)

```
In [66]: 1000 & 1077
```

```
Out[66]: 32
```

OR Operator (|)

```
In [65]: 100 | 101
```

```
Out[65]: 101
```

- 0b1100100 | 0b1100101 -> 0b1100101 (Acroding to the truth table)

```
In [67]: 1057 | 1999
```

```
Out[67]: 2031
```

XOR (^)

```
In [68]: 100 ^ 101
```

```
Out[68]: 1
```

- 0b1100100 ^ 0b1100101 -> 000001

```
In [70]: int(0b000001)
```

```
Out[70]: 1
```

Left Shift (<<)

```
In [71]: 10 << 1
```

```
Out[71]: 20
```

- 10 ->1010
- 1010 << 1 = 10100 = 20

```
In [73]: int(0b10100)
```

```
Out[73]: 20
```

```
In [74]: 10 << 2
```

```
Out[74]: 40
```

```
In [75]: 10 << 5
```

```
Out[75]: 320
```

Right Shift (>>)

```
In [78]: 10>> 1
```

```
Out[78]: 5
```

- 10 -> 1010
- 1010 >> 1 = 101 = 5

```
In [79]: int(0b101)
```

```
Out[79]: 5
```

```
In [80]: 10 >> 2
```

```
Out[80]: 2
```

```
In [81]: 10 >> 3
```

```
Out[81]: 1
```

```
In [82]: 10 >> 4
```

```
Out[82]: 0
```

```
In [ ]:
```