10-09-25

September 10

Bit-wise Operators:

Formula: 2**n

n is the number of bits

Binary digits if it is one bit: only two combinations are possible (1,0)

if it is 2 bits: four combinations are possible - 00 01 10 11(0-3)

if it is 3 bits: eight combinations- 000 001 010 001 100 110 101 111(0-7)

if it is 4 bits: 16 combinations (0-15)

if it is 5 bits: 0-31

Binary representation of decimal numbers:

0-000

1-001

2-010

3-011

4-100

5-101

6-110

7-111

9-1001

12-1100

15-1111

Bit wise operators in python: & | ~ >> <<

Bitwise and,or

1 1 1 1 => 15

0 1 1 0 => 6

and

0 1 1 0 => 6

or

1 1 1 1 => 15

Examples:

15 **&** 6

6

15 | 6

15

1 1 0 0 => 12

0 0 1 1 => 3

 # and

0 0 0 0 => 0

#or

1 1 1 1 => 15

12 **&** 3

0

12 | 3

15

3 **&** 12

0

3 | 12

15

left shift and right shift:

right shift 13 by 2 times

13 >> 2

1 1 0 1 => 13

after one left shift:

0 1 1 0

after two shifts:

0 0 1 1

13 >> 2 = 3

left shift 13 by 2 places

13 << 2

1 1 0 1 => 13

after one left shift:

1 1 0 1 0 => 26

after two shifts:

1 1 0 1 0 0 => 52

13 << 2 = 52

13 >> 2

3

13 << 2

52

Nested if Statements:

- one if inside another if is considered as nested if statements.
- we should write two if statements one after the other.
- Both statements are dependent and have some relation.
- The computational time is comparatively less than the if...elif...else statements.

Syntax:

```
if(condition1):#outer if
   if(condition2):#inner if
      statements of condition2
    else:
      statements of inner else
 else:
   statements of outer else
```

Example:

Write a program to print if the given number is positive,negative or zero

```
n = int(input("enter a number : "))
```

```
if(n >= 0):
    if(n > 0 ):
        print("The number is +ve")
else:
    print("The number is -ve")
```

```
enter a number : -2
The number is -ve
```