* Python Bitwise Operators

They are used to Compare binary numbers.

(i) AND (&): Performs a binary AND operation.

(ii) OR (|): Performs a binary OR operation.

(iii) XOR (^): Performs a binary XOR (exclusive OR) operation.

(iv) NOT (~): Performs a binary NOT operation.

(v) left Shift (<<): Shifts bits to the ~~right~~ left and fills 0 on voids or Copies the sign bit.

(vi) Right Shift (>>): Shifts bits to the right.

Example

2^0 2^1 2^2 2^3 2^4

1 2 4 8 16

→ AND 5 & 3 = 1

```
  1 0 1 0
  1 1 0 0
---------
  1 0 0 0  = 1
```

Satisfy both Condition

→ OR 5 | 3 = 7.

```
  (1) (2) (4) (8)
  1 0 1 0
  1 1 0 0
---------
  1 1 1 0  - 7.
```

Satisfy atleast one Condition.

→ XOR (^)

5 ^ 3 = 6

Result is 1 if bits are diff.

```
  1 0 1 0
  1 1 0 0
---------
  0 1 1 0  - 6
```

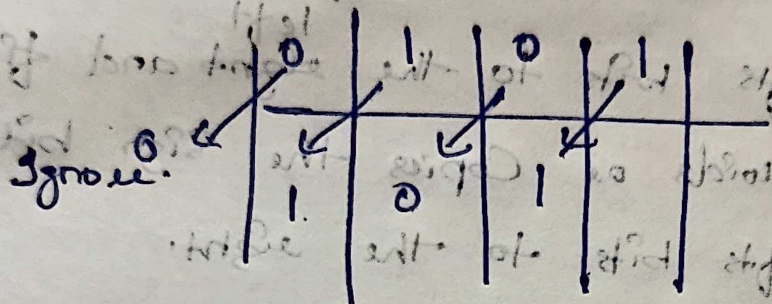
→ Not (~)

(v) left shift (<<)

eg: 5 << 1 = 10

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1

5<<1



→ 10

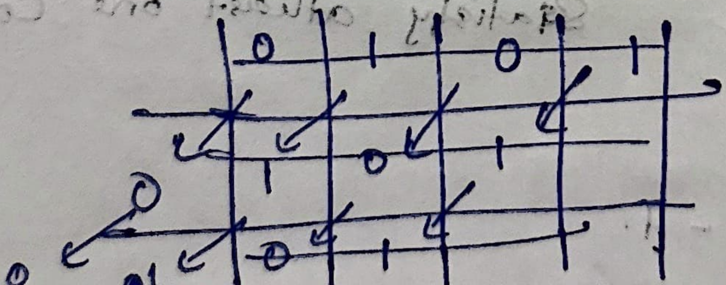
(vi) Right shift (>>)

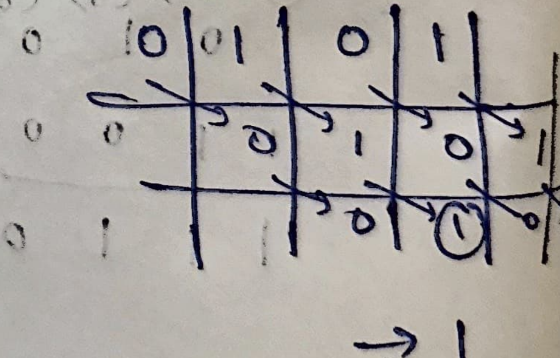
Shift bits to the right.

for positive numbers, fill left side with 0

for negative number, fill left side with 1.

eg: 5 >> 2 = 1





→ 1

* Nested if Statement

using one if an inside another.

(an if statement inside another if)

Syntax:

```
if (condition 1): outer if
    if (condition 2): inner if:
        (Statement of Condition 2)
    else:
        (Statement of ~~Condition~~ inner if)
else:
    Statement of Condition 1 (outer if).
```

Example

write a program to check the given input is +ve, -ve, 0

```
n = int (input ("enter the number"))
if (n>=0):
    if (n>0):
        print ("positive")
    else:
        print ("~~negative~~ zero")
else:
    print ("negative")
```