28 Sept 2022

Lec 17

Bitwise Operators

- bitwise operators work on bits
- used to compare binary numbers (operate on)

eg:- a = 5
b = 4
print (a & b) ⇒ 4

[∵ in binary 5 = 0101
4 = 0100

5 & 4 | 0100 | ⇒ 4]

&	bitwise AND
|	OR
^	XOR
~	NOT / Compliment
<<	left shift
>>	right shift.

Bitwise & (and) sets the bit 1 if both the bits are 1

print (5 | 4) • print (a | b) ⇒ 5

[∵ 0101
0100
5 | 4 | 0101 | ⇒ 5]

Bitwise OR (|) sets the bit 1 if one of the two bits is 1

XOR (^) :-

print (a ^ b) ⇒ 1

[∵ 5 = 0101
4 = 0100
| 0001 | ⇒ 1]

Bitwise XOR (^) sets 1 if both bits are different

Compliment (~) :-

print (~a) ⇒ -6

~ will do reverse of the bit.

eg:- ~0 = 1
~1 = 0

So here ~a a = 5

0101

So ~a = 1010

Now -6 to store -ve numbers in memory, we have to convert them, bcz only +ve numbers can be stored in memory.

So using 2's compliment we can store -ve numbers.

2's = 1's + 1

what is 1's (ones compliment) ?

we can get 1's compliment of any binary number by just reversing its bits.

eg:- 6 = 0110

1's of 6 = 1001

1's + 1 : 1001 + 1 = 1010 ⇒ it is 2's compliment of -6 which is same as ~5

So we got ~a = -6

~5 = -6

left shift (<<) :- a = 5

print(a << 2) ⇒ 20

5 = 00000101

00010100 ⇒ 20

[x<<y = x * 2^y]

- It shifts the bits of first operand with respect to second operand
- shifts left by pushing zeros in from the right & let the leftmost bits fall off. In left shift we gain bits.

right shift (>>) :-

```
a = 5
print (a >> 2)
```

5 = 0101 → 0001 ⇒ 1

here we are losing bits.

$[x >> y = x / 2^y]$

exercise:

```
print (26 & 23)
print (17 | 24)
print (17 ^ 24)
print (~45)
print (68 << 2)
print (56 >> 3)
```