The bitwise exclusive OR operator (^) :
It compares each bit of its first operand to the corresponding bit of its second operand. If one bit is 0 and the other bit is 1, the corresponding result bit is set to 1. Otherwise, the corresponding result bit is set to 0. (Identical is 0)

Syntax:
Bitwise Exclusive OR Operation = a ^ b

Truth Table for Bitwise exclusive OR(^)

A	B	^(Result)
0	1	1
1	0	1
1	1	0
0	0	0

NOTE: Identical is 0

ANALYSIS
We declared 2 integers a and b, The binary form of
10 = 00001010 ==> for 1 Byte ==> 8 bits
20 = 00010100 ==> for 1 Byte ==> 8 bits
0000 1010 ^ 0001 0100 ==> 0001 1110=> Result is 30

Example:
```
a=10;b=20
print(a^b)
```

EXAMPLE:
```
a=9;b=65
print(bin(a))#0b0000 1001
print(bin(b))#0b0100 0001
#0000 1001 ^ 0100 0001
#0100 1000
print(0b01001000)#72
print(a^b)#72
```

EXAMPLE:
```
print(True & False)
print(True | False)
print(True ^ False)
```

Bitwise Complement:(~)
It is popularly known as bitwise not operator. It is unary operator. It is flipping bits.

Input	Output
0	1
1	0

```
>>> a=20
>>> ~a
-21
```

Formula is:
~x = -x-1

~20=-20-1=-21
~30=-30-1=-31

Binary Ones Complement (Mathematically)
It is unary and has the effect of 'flipping' bits. Given number should be convert into binary number. That binary number converting '0' to '1' and '1' to '0' is called Binary Ones Complement.

In Mathematical Approach:
x=10 ==> 0000 1010 (is Binary Number)
Binary Ones Complement is ==> 1111 0101

Binary Twos Complement:
Add '1' to Binary Ones Complement is called Binary Twos Complement.
1111 0101
+1

1111 0110

NOTE: Binary Addition computations are the following:
1+1=10, 1+0=1, 0+1=1,0+0=0,1+1+1=11

Binary Ones Complement in PYTHON Solution:
It is unary and has the effect of 'flipping' bits. Given number should be convert into binary number. That binary number converting '0' to '1' and '1' to '0' is called sign of a number (Plus+ or Minus-).

STEP1:
x=10 ==> 0000 1010 (is Binary Number)
0000 1010

If converted number starts with 1 sign is Minus-
If converted number starts with 0 sign is Plus+

1111 0101 ===> Sign is -(Minus)

NOTE:
Mathematically binary ones complement is signature (+ or -) in PYTHON step1..!

STEP2:
Binary Ones Complement:
In PYTHON adding '1' to given binary number is called Binary Ones Complement.

x=10 ==> 0000 1010 (is Binary Number)
0000 1010
+1

0000 1011 ==> -11 (Sign(-) from Step 1)

NOTE: Bitwise NOT (~) operator
Example:
a=10;print(~a)#-11

STEP3:

Binary TWOs Complement:
In PYTHON adding '1' to Binary Ones Complement is called Binary TWOs Complement.

```
0000 1011
            +1
-----------------
0000 1100 ==> -12
----------------
```

<< Binary Left Shift
The left operands value is moved left by the number of bits specified by the right operand.

```
a = 10          # 10=1010
a << 2;        # 40=101000==>Add Two digits right side
```

>> Binary Right Shift
The left operands value is moved right by the number of bits specified by the right operand.

```
a = 10        #10=1010
a >> 2;      #2=10==>Remove two digits right side
```

EXAMPLE:

```
a=10
print(a<<2)#40
print(a>>2)#2
```

EXAMPLE:Output of the following Script:

```
print(~True)
print(True<<2)
print(True>>2)
```

Ternary Operator in Python
Ternary operators also known as conditional expressions. It was added to Python in version 2.5. It allows to test a condition in a single line.

Syntax:

```
[on_true] if [expression] else [on_false]
```

Example:

```
a,b =10,20
Min = a if a < b else b
print(Min)
Max = a if a > b else b
print(Max)
```

Example:

```
a=int(input("Enter Any Number:"))
b=int(input("Enter Any Number:"))
Max=a if a>b else b
print("Max Value is:",Max)
```

NOTE:
Nesting of ternary operator is possible.

Example:

```
x=int(input("Enter First Number:"))
y=int(input("Enter Second Number:"))
z=int(input("Enter Third Number:"))
Min=x if x<y and x<z else y if y<z else z
print("Minimum Value:",Min)
```

Python Operators Precedence (PEMDAS)

1 Parentheses are always respected
2 Exponentiation (raise to a power)
3 Multiplication, Division, and Remainder
4 Addition & Subtraction
5 Left to right

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive 'OR' and regular 'OR'
<= < > >=	Comparison operators
== !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is,is not	Identity operators
in,not in	Membership operators
not or and	Logical operators

Example:

```
a = 20;b = 10;c = 15;d = 5;e = 0
e = (a + b) * c / d       #( 30 * 15 ) / 5
print("Value of (a + b) * c / d is ",  e)

e = ((a + b) * c) / d     # (30 * 15 ) / 5
print( "Value of ((a + b) * c) / d is ",  e)

e = (a + b) * (c / d);    # (30) * (15/5)
print( "Value of (a + b) * (c / d) is ",  e)

e = a + (b * c) / d;      #  20 + (150/5)
print ("Value of a + (b * c) / d is ",  e)
```