Python Identity Operators
They are used to check if two values (or variables) are located on the same part of the memory. Identity operators compare the memory locations of two objects.

Operator	Meaning
is	True if the operands are identical
is not	True if the operands are not idenical

Syntax:
```
operand1 is operand2
operand1 is not operand2
```

Example:
```
a = b = [1,2,3]
c = [1,2,3]
print( a is b)
print( a is c)
```

Difference between "is" vs "=="
The is operator may seem like the same as the equality operator but they are not same. The is checks if both the variables point to the same object whereas the == sign checks if the values for the two variables are the same.

Example:
```
a=[1,2,3];b=a;print(b)
print(a is b);print(a == b)
c=list(a);print(c)
print(a is c);print(a == c)
```

id() function:
It is used to return the identity of an object

Syntax:
```
id(object)
```

Example:
```
a=[1,2,3]
print(id(a))#72563690248
b=a
print(id(b))#72563690248
print(b)#[1,2,3]
print(a is b)#True
print(a==b)#True
```

NOTE:
Avoid using 'is' operator for immutable types such as strings and numbers, the result is unpredictable

Python Bitwise Operators
These are used to perform bit operations. All the decimal values will be converted into binary values and bitwise operators will work on these bits.

OPERATOR	MEANING
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
~	Bitwise complement
<<	Shift left
>>	Shift right

bitwise AND (&)
It compares each bit of the first operand to the corresponding bit of the second operand. If both bits are 1, the corresponding result bit is set to 1.

Truth Table for Bitwise AND (&)

A	B	&(Result)
0	1	0
1	0	0
1	1	1
0	0	0

NOTE: Both are True the result will be True..!!

Example:
```
a=10;b=20
```

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 ==> 0000 0000=> Result is 0

Example:
```
a=10;b=20
print(a&b)#0
```

Example:
```
a=10;b=20
print(bin(a))#0b1010
print(bin(b))#0b10100
#0000 1010 & 0001 0100
print(0b00000000)#0
print(a&b)#0
```

Example:
```
a=9;b=65
print(bin(a))#0b1001
print(bin(b))#0b1000001
#0000 1001 & 0100 0001
print(0b00000001)#1
print(a&b)#1
```

bitwise OR
It takes two bit patterns of equal length. The result in each position is 0 if both bits are 0, while otherwise the result is 1 (Any One 1 the Result is 1)

Syntax:

Bitwise OR Operation = a | b

Truth Table for Bitwise OR

A	B	|(Result)
0	1	1
1	0	1
1	1	1
0	0	0

NOTE:Any '1' the result is 1.

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)#30
```

Example:
```
a=9;b=65
print(bin(a))#0b1001
print(bin(b))#0b1000001
#0000 1001 | 0100 0001
#01001001
print(a|b)#73
print(0b1001001)#73
```

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)
```