Litereals in Python :

>>a = 15

'a' is variable and 15 is integer literal

likewise we have following literals

1. Numeric literals
2. Boolean literals
3. String literals

salary = 199.234

left hand side is variable because value is assigned to it.
salary name is identifier
199.234 is floating literal

Determine datatype :
---
type(variable name)

OPERATORS IN PYTHON :
==========

* The operator is symbol that perform some operation.
* a+b => a,b are operands and + is operator

1. Arithmetic operators
2. Assignment operators
3. Unary minus operators
4. Relational operators
5. Logical operators
6. Boolean operators
7. Bitwise operators
8. Membership operators
9. Identity operators

Arithmetic operators : +,-,*,/,//,%
---
x = 15
y = 4

```
print('x + y =',x+y)
# Output: x + y = 19
```

```
print('x - y =',x-y)
# Output: x - y = 11

print('x * y =',x*y)
# Output: x * y = 60


print('x / y =',x/y)
# Output: x / y = 3.75


print('x // y =',x//y)
# Output: x // y = 3
--divide the result to whole number

print('x % y =',x%y)
# Output: x % y = 3
-- remainder value

print('x ** y =',x**y)
# Output: x ** y = 50625
```

Assignment operators

Assignment operators in Python

Operator	Example	Equivalent to
=	x = 5	x = 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
**=	x **= 5	x = x ** 5
^=	x ^= 5	x = x ^ 5

Unary minus Operator :

Unary operator is denoted by (-) symbol. when this operator used before symbol the value will be negated

```
>>> n = 10
>>> print(-n)
-10
```

```
Comparison (Relational) Operators
---------------------------------

x = 10
y = 12


print('x > y  is',x>y)
# Output: x > y is False


print('x < y  is',x<y)
# Output: x < y is True

print('x == y is',x==y)
# Output: x == y is False

print('x != y is',x!=y)
# Output: x != y is True

print('x >= y is',x>=y)
# Output: x >= y is False

print('x <= y is',x<=y)
# Output: x <= y is True


Logical operators
-----------------------
>>> a = 10
>>> b = 20
>>>
>>> c = a > b
>>> d = a < b
>>> c
False
>>> d
True
>>>
>>> c and d
False
>>>
>>> c or d
True
>>>
>>> not c
True
>>>
```

```
>>> not a
False
```

Boolean Operators :

We know there are two bool type literals. They are TRUE and FALSE. Boolean operators act upon 'bool' type literals and they provide 'bool' type output.

It mean the result provided by boolean operators will be again either True or False.

There are 3 boolean operators as mentioned below

Let's take x= True
y= False

```
>>> x = True
>>> y = False
>>> x and y
False
>>> x or y
True
>>> not x
False
>>> not y

True
```

TRUTH TABLE:

X	Y	X AND Y	X OR Y	x xor y
T	T	T	T	F
T	F	F	T	T
F	T	F	T	T
F	F	F	F	F

Membership operators

Operator Meaning Example

in True if value/variable is found in the sequence
5 in x

not in True if value/variable is not found in the sequence
5 not in x

```
x = 'Hello world'
y = {1:'a',2:'b'}


print('H' in x)
# Output: True

print('hello' not in x)
# Output: True

print(1 in y)
# Output: True

print('a' in y)
# Output: False
```

```
>>> a = [101,"srinivas","Hyd"]
>>> b = 101
>>> if(b in a):
        print("true")
else :
        print("false")
```

true

```
>>> a = {101:"srinivas",102:"Ravi",103:"Vijay"}
>>> rno = int(input("enter roll number\n"))
enter roll number
101
>>> print(rno)
101
>>> if( rno in a):
        print(a[rno])

'srinivas'
```

Identity operators :

These operators compare the memory location of two objects. Hence it is possible to know whether
the two objects are equal or not.

Object memory location can be identified by using id() function.

a =25
b =25

They are two identity operators

1. is
2. is not

*'is' operator :
it will compare the identity number of two objects. if both are same it will return true otherwise false.

* 'is not' operator :

```
>>> a = 10
>>> b = 20
>>> a is b
False
>>> a is not b
True
>>> type(a) is int
True
>>> type(a) is not int
False
```

Difference between is and ==

```
>>> a = [1,2,3,4]
>>> b = [1,2,3,4]
>>> a == b                    // value is same
True
>>> a is b                    // value same but identity location is different.
False

>>> id(a)
52538696
```

```
>>> id(b)
52538504


Operator Precedence :
--------------------

()                                --> paranthesis
**                                --> Exponential
-,~                                       --> unary mins, Bitwise
complement
*,/,//,%                  --> Multiplication,division,floor
division,Modules
+,-                                       --> Addition,subtraction
<<,>>                             --> Bitwise left shift, Bitwise right
shift
&                                         --> Bitwise AND
^                                         --> Bitwise XOR
|                                         --> Bitwise OR
>,>=,<,<=,==,!=      --> Assignment operators
is, is not                        --> Identity operators
in, not in                        --> Membership operators
not                                       --> logical not
or                                        --> logical or
and                                       --> logical and


precedence represents the priority level of the operator.
The operator which are high precedence will execute first than of lower
precedence.

>>> a=1
>>> b=2
>>> c=3
>>> d=4
>>> e=5
>>> (a*b)+c*d
14
>>> a*(b+c)*d
20
>>> a*b+c*d
14
>>>


Bitwise operators
---------------------
```

Binary Number : starts with 0b

```
>>> a = 0b1101
>>> print(bin(a))
0b1101
>>> print(a)
13


>>> b = 18
>>> print(bin(18))
0b10010
>>> b = 0b10110
>>> print(b)
22
```

Octal Number : start with 0O

```
>>> a = 0O101
>>> print(a)
65

>>> b = 96
>>> print(oct(b))
0o140
```

Hexa Decimal : starts with 0X

```
>>> b = 26
>>> print(hex(b))
0x1a
>>> b = 27
>>> print(hex(b))
0x1b
>>> b = 28
>>> print(hex(b))
0x1c
>>> b = 29
>>> print(hex(b))
0x1d
>>> b = 30
>>> print(hex(b))
0x1e
>>> b = 31
>>> print(hex(b))
0x1f
>>> b = 32
```

```
        >>> print(hex(b))
        0x20
```

i)Bitwise complement operatory(~)

```
>>> a = 4
>>> ~a
-5
>>> bin(a)
'0b100'
```

```
a=4 ---> 0100
~a --->  1011  ---> 11
```

But here compliler display -5 (in 2's complement)

~N = -(N+1)

~4 =

```
N = 4 =  0100
+1    =     1
        --------
(N+1)   0101     = 5
```

~N = -(N+1)
~4 = -5

ii) Bitwise AND operator(&)

```
>>> a = 4
>>> b = 5
>>> a & b
4
```

```
a =   0100
b =   0101
a&b = 0100 = 4
      (both 1 then only 1 otherwise 0)
```

iii) Bitwise OR operator(|)

```
>>> a = 4
>>> b = 5
>>> a | b
5
```

```
a =  0100
b =  0101
a|b = 0101 = 5

(both 0 then 0 remaining 1)


iv) Bitwise XOR Operator(^)

>>> a = 4
>>> b = 5
>>> a ^ b
1

a   = 0100
b   = 0101
a^b = 0001

(any one should be 1 then 1 otherwise 0)



v) Bitwise Left shift operator(<<)

>>> a = 4
>>> a << 2
16
>>> a << 3
32

a = 4 = 0000 0100
a << 2= 0001 0000  = 16
a << 3= 0010 0000  = 32

vi) Bitwise Right shift operator(>>)


>>> a = 24
>>> a >> 2
6

a     = 0001 1000
a >>2 = 0000 0110  = 6
```