2.0
0.0

e.g

```
x=7 + 8j
print x.real, " ",x.imag

7.0   8.0
```

Python allows 3 types of integer literals:

(a) **Decimal Integer Literal**: An integer literal consisting of sequence of digits taken to be a decimal integer literal unless it begins with 0 (zero).

(b) **Octal Integer Literal**: A sequence of digits starting with 0 is taken as octal integer literal.
e.g. Decimal 8 is 010.

(c) **Hexadecimal Integer Literal**: A sequence of digits preceded by OX or ox is taken to be hexadecimal integer literal.

(d) **Floating Point Literal**: A number with a fractional part or decimal point consists of sign (+,-), sequence of decimal digits with a dot such as
0.0609, 15.0609

These numbers can also be used to represent a number in engineering and scientific notation

$6.09 * 10^{-2}$, $1.50609 * 10^{1}$

Note: Exponent form – A real literal in exponent form consists of 2 parts: mantissa and exponent. E.g. 6.09 E0(-2) is exponent form of 0.0609.

Boolean literals

Used to represent one of the two boolean values, that is True and False.

For example:

```
>>>bool_1=(6>10)
>>>print (bool_1)
False
```

Special literals

- **None**

It is used to indicate something that has not been created yet or to signify the absence of values in a situation. Python doesn't display anything when we give a command to display the value of a variable containing value as None.

For example:

```
>>>value1=20
>>>value2=None
>>>value1
```

20
>>>value2
>>>

Literals collections

Python supports literals collection such as tuples, list etc.

Type Function:

Type function determine the type of an object. Object can be a variable or a literal etc.

A=10
Type(A) # this type returned as <class int>
A=10.5
Type(A) # this type returned as <class float>

Expression:

An expression is any legal combination of symbols that represents a value.

For example: 15, a+10, a+3, b>5

Statement:

A statement is a programming statement that does something i.e., some action takes place.

print ("Hello") # this statement calls print function.

Note: While an expression is evaluated, a statement is executed i.e., some action takes place.

Comment:

Comments are the additional readable information, which is read by the programmer but ignored by Python interpreter. In Python comments begin with #(Pond or hash character) and end with end of physical line.

Example:

print ("Hello") # this statement calls print function.

Multi line comment:

We can enter multi line comment in two ways:

(i) Add a # symbol in beginning of every line part of the multi line comment

Example:

(ii) Type comment as a triple quoted multi line string

Example:

```
'''

'''
```

This type of multi line comment is also known as docstring. We can either use triple-apostrophe(''') or triple quotes(""") to write docstring. The docstring are very useful in documentation.

Block and Indentation:

A group of statement which are part of another statement or a function are called block or code-block or suite in Python

```
if (b < a ):
    temp=a
    a=b
    b=temp
```

Indentation creates blocks of code. Statement at same indentation level are part of same block or suite. Statement requiring suite/code block have a colon (:) at their end. We cannot unnecessary indent a statement; Python will raise error for that.

Variable and Assignment:

Variables represent name storage locations whose value can be manipulated during program run. Variables are also called symbolic variables because these are named labels. For instance, following statements create a variable of numeric type:

```
marks=90
```

Python creates a variable of the type similar to the value assigned.

In Python every element is termed as an object data. A variable/object has three main components:

(i) Identity of the variable/object
(ii) Type of the variable/object
(iii) Value of the variable/object

Identity of the variable/object:

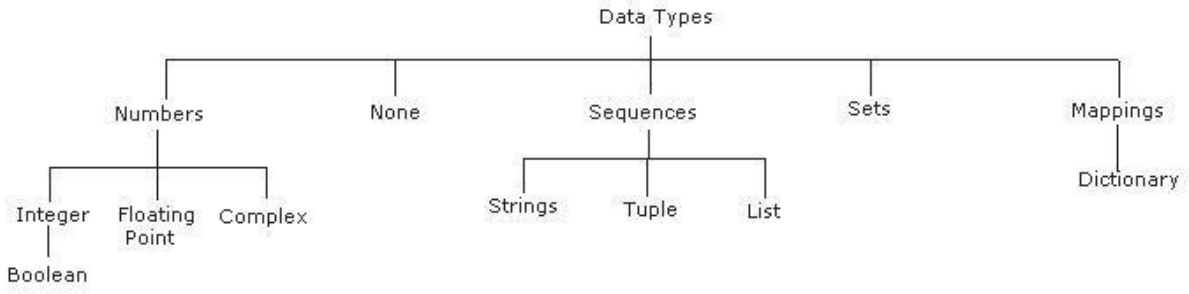
The id of an object is generally the memory location of the object. Although id is implementation dependent. The address of an object can be checked using the this method i.e., id (object).

Syntax : id(variable/object)

Example : id(x)

Type of the variable/object

By type we mean data type of a variable. Data type of a value refers to the type of value it holds and the allowable operation on those values.



Value of the variable/object

Values are assigned to the variable using assignment operator (=).

Maths=87

In the above example, Python associates the name (variable) Maths with the value 87 i.e., the name (variable) Maths is assigned the value 87 or, in other words, the name (variable) Maths refers to value 87.

Mutable and Immutable Types:

Python data objects can be broadly categorized into two –mutable and immutable types, in simple word changeable or modifiable and non-modifiable types.

Immutable types

The immutable types are those that can never change their value in place. In python the following are immutable: integers, floating point numbers, Boolean, string tuples.

Variable names are stored as references to a value-object. Each time we change the value, the variable's reference memory address changes.

Note: Mutability means that in the same memory address, new value can be stored. The type which do not support this property are immutable.

Mutable Type:

Mutable are those type whose values can be changed in place. Only three types are mutable in Python. These are: lists, dictionaries and sets.

For example to chane in a list

Chk=[1,2,45]

Chk[1]=40

It will make the list namely chk as [1,40,45]

Lvalues and Rvalues:

There should be one and only one variable on the left-hand side of the assignment operator. This variable is called the Left value or the L-value. There can be any valid expression on the right-hand side of the assignment operator. This expression is called Right value or R-value.

The statement :

L-value=R-value

is called assignment statement. When the interpreter encounters an assignment statement, it evaluates the right-hand side expression (R-value) and assigns the value to the left-hand side variable (L-value).

Lvalues are the objects to which we can assign a value or expression. Lvalues can come on lhs or rhs of an assignment statement.

Rvalues are the literals and expression that are assigned to lvalues. Rvalues can come on the rhs of an assignment statement.

For example we can write

A=20

B=10

But we cannot write

2=A

10=B

A*2=B

Multiple Assignment

1. We can assign same values to multiple variable in a single line.
   a=b=c=90

2. Assigning multiple values to multiple variables.
   X,y,z=10,20,30

Dynamic Typing:

A variable pointing to a value of a certain type, can be made to point to a value/object of different type. This is called dynamic typing.

For example:

X=10

print (x)

X="Hello World"

Output:

10

Hello World

In the above code variable X is first pointing to/referring to an integer value 10 and then to a string value "Hello World".

Note: Variable X does not have type but the value it points to does have a type. So we can make a variable point to a value of different type by reassigning the value of that type; Python will not raise any error. This is called Dynamic Typing features of Python.

Type Casting

Python internally changes the data type of some operands. These types of conversion is automatic, i.e. implicit and hence known as implicit type conversion.

However, Python, also supports Explicit type conversion.

⇒ **Explicit Type Conversion**: User defined that forces an expression to be of specific type. Also known as type casting.
e.g. >>>x = 7.5
>>>x=int(x)
>>>print x
>>>7

Input function:

Input refers to the data entered by the user of the program. In Python, we have input() function available for input.

syntax:

variable=input("statement")

e.g. x=input('What is your name')

The input () function always returns a value of string type.

Reading Numbers:

After inputting value through input we can change the value to integer or float by using int () and float ().

For example

x=input('Enter the no')

x=int(x)

or we can combine bothe the previous statement as

x=int(input('Enter the no'))