Example:

```
class Employee:  # developing data type
    pass


def main():
    c1=complex() # creating object of "complex class/data type"
    print(c1.real,c1.imag)
    emp1=Employee() # creating object of "employee class/data type"
    # the object is created without properties and functions/methods
    list1=list() # creating object of "list class/data type"
    list1.append(10)

main()
```

Output:

```
0.0 0.0
>>>
```

Methods

Functions defined inside class are called methods
Methods defines the behavior or functionality of object
The methods defined inside class are 3 types.

1. Object level method/instance method
2. Class level method
3. Static method

Object level method or instance method

A method defined inside class with first argument “self” is called object level method.
This method is bind with object name
This method is invoked or executed with object name
This method defines behavior or functionality of object
“self” argument name, which hold address of current object to which method is bind.
“self” is not a keyword or argument name/variable name
It is an implicit variable/argument, this argument receive address of object on which the method is invoked.

Syntax:

```
def <method-name>(self,arg1,arg2,arg3,…):
        statement-1
        statement-2
```

this method cannot be invoked without creating object.

Example:

```
class Fan:
    def rotate(self): #==> object level method/instance method
        print("fan rotate")
        print(self)


def main():
   list1=list() # created object of list
   list1.append(10) # append is object level method/instance method
   fan1=Fan() # creating object of Fan class
   fan1.rotate()
   fan2=Fan() # creating object of Fan class
   fan2.rotate()


main()
```

Output:

```
fan rotate
<__main__.Fan object at 0x000000FD518F2F40>
fan rotate
<__main__.Fan object at 0x000000FD5387DC70>
>>>
```

Example:

```
class Robo:
    def talk(self,msg): # object level method/instance method
        print(msg)


def main():
    robo1=Robo() # creating object of Robo class
    robo1.talk("My Name is Robo1")
    robo2=Robo() # creating object of Robo class
    robo2.talk("My Name is Robo2")
    list1=list()
    list2=list()
    list1.append(10)
    list2.append(20)
    print(list1,list2)
main()
```

Output:

```
My Name is Robo1
My Name is Robo2
[10] [20]
>>>
```

Example:

```
class Calculator:
    def add(self,a,b):
        return a+b
    def sub(self,a,b):
        return a-b
    def multiply(self,a,b):
        return a*b
    def div(self,a,b):
        return a/b

def main():
    calc1=Calculator()
    res1=calc1.add(10,20)
    res2=calc1.sub(50,20)
    res3=calc1.multiply(5,3)
    res4=calc1.div(10,2)
    print(res1,res2,res3,res4)
main()
```

Output:

```
30 30 15 5.0
>>>
```

Object level method cannot invoke with class name.

Constructor method or constructor
What is constructor?
Constructor is a special method or magic method
Constructor is object level method or instance method
Constructor is executed automatically whenever object of class is created
Constructor is used to initialize object
Constructor is used to define initial properties of object
Constructor is used to allocate resource to object
The block of code which has to be executed on creation of object is defined inside constructor

Note: any method which is prefix and suffix with __ is called magic method

Syntax:
```
def __init__(self,arg1,arg2,arg3,…):
        statement-1
        statement-2
```

We can define the constructor,
1. With arguments
2. Without arguments

Example:
```
class A:
   def __init__(self):
      print("inside constructor")


def main():
   obj1=A()
   obj2=A()
   obj3=A()
main()
```

Output:
```
inside constructor
inside constructor
inside constructor
>>>
```

Example:

```
class A:
    def __init__(self,x,y):
        print("constructor with arguments")
        print(x,y)

def main():
    obj1=A(100,200)
    obj2=A(500,600)
    str1=str("Java")
    print(str1)
    str2=str("Python")
    print(str2)

main()
```

Output:

```
constructor with arguments
100 200
constructor with arguments
500 600
Java
Python
>>>
```