In [5]:

```
def add(a,b):  #formal arguments
    print("a =",a)
    print("b =",b)
add(1,2,3,4,5)   #actual arguments
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-5-6624fa2884c8> in <module>
      2     print("a =",a)
      3     print("b =",b)
----> 4 add(1,2,3,4,5)   #actual arguments

TypeError: add() takes 2 positional arguments but 5 were given
```

In [6]:

```
def add(a,*b):  #formal arguments
    print("a =",a)
    print("b =",b)
add(1,2,3,4,5)   #actual arguments
```

```
a = 1
b = (2, 3, 4, 5)
```

In [8]:

```
def add(a,*b):
    summation = a  #summation = 1
    for i in b:  # 2
        summation += i
    print(summation)

add(1,2,3,4,5)
```

```
15
```

What is Object Oriented Programming(OOPS)?

- OOps allows decomposition of a problem into a no of units called objects.
- Python is an object Oriented Programming Language.

Why to use OOPs?

- Provides a clear program structure.
- It makes the development and maintance easier.
- Code reusability

Classes

- Class is a collection of variables and methods.

  Syntax: class className:

```
list of variables
list of methods
```

Objects

- An object is also called an instance of a class.
- An object is a collection of data and methods.

Syntax: objectname = ClassName

In [9]:

```
# Example for Class creation
class Hi:
    a,b = 10,20
    def display():
        print("Hi,Iam from display function")

obj = Hi
print(obj.a)
print(obj.b)
print(obj.display())
```

```
10
20
Hi,Iam from display function
None
```

In [10]:

```
# Example for Class creation
class Hi:
    a,b = 10,20
    def display():
        print("Hi,Iam from display function")

obj = Hi
print(obj.a)
print(obj.b)
obj.display()
```

```
10
20
Hi,Iam from display function
```

In [15]:

```
class Math:
    def add(n1,n2):
        return n1+n2
    def mul(n1,n2):
        return n1*n2

obj = Math
print(obj.add(12,13))
print(obj.mul(2,3))
```

```
25
6
```

Constructor

- It's task is to intialize to the data members of a class when an object of a classis created.

```
Syntax:
        class classNamme:
             def __init__(self): it is a constructor
             def __init__(self,a,b):
             def __init__(a,b,self):
```

- The self parameter is a reference to the current instance of the class,and is used to access variables thatt belongs to the class.

In [17]:

```
class Math:
    def __init__(self,n1,n2):
        self.n1 = n1
        self.n2 = n2
    def show(self):
        print(self.n1)
        print(self.n2)

obj = Math(2,5)
obj.show()
```

```
2
5
```

In [18]:

```
class MyClass:
    x = 5

print(MyClass)
```

```
<class '__main__.MyClass'>
```

In [19]:

```
class Math:
    def __init__(abc,n1,n2):
        abc.n1 = n1
        abc.n2 = n2
    def show(abc):
        print(abc.n1)
        print(abc.n2)

obj = Math(2,5)
obj.show()
```

```
2
5
```

Single inheritance

In [30]:

```
class A:
    a,b = 10,20
    def display():
        print("I am form class A")
class B(A):
    c,d = 13,15
    def show():
        print('I am from class B')

obj = B
print(obj.b)
print(obj.c)
print(obj.display())
```

```
20
13
I am form class A
None
```

In [ ]:

```

```

```
### Multilevel inheritance

* One or more parent classes and one or more child classes.

```

In [23]:

```
class A:
    def classA():
        print("I am from classA")
class B(A):
     def classB():
        print("I am from classB")
class C(B):
    def classB():
        print("I am from classB")


obj = C
print(obj.classA())
print(obj.classB())

```

```
I am from classA
None
I am from classB
None
```

Multiple Inheritance

- More than one parent class and one child class

In [44]:

```
class A:
    b = 10
    def classA():
        print("I am from classA")
class B:
    a = 30
    def classB():
        print("I am from classB")
class C(A,B):
    def classB():
        print("I am from classB")
        # sum = add(a+b)


obj = C
print(obj.classA())
# print(obj.sum)
```

```
I am from classA
None
```

In [45]:

```
print(obj.classB())
print(obj.a)
```

```
I am from classB
None
30
```

In [ ]:

```

```