In [2]:
```
#Arguments in Function
#Default arguments
#Default arguments means if the arguments ae defined in function def but when you call it you give them new val

# Create a function with name average

def average(a, b):  #a and b are the arguments
    print("The average is:", (a+b)/2)   #you need to specify what operation you want to perform

#Call function
average(2,8)   #give values
```

```
The average is: 5.0
```

In [3]:
```
# Create a function with name average
def average(a=5, b=1):  #a and b are the arguments
    #even the arguments in function def are 5 and 1 but it will call for 2 and 8, "This is Default argument"
    print("The average is:", (a+b)/2)   #you need to specify what operation you want to perform

#Call function
average(2,8)   #give values
```

```
The average is: 5.0
```

In [4]:
```
# Create a function with name average
def average(a=5, b=1):  #a and b are the arguments
    #even the arguments in function def are 5 and 1 but it will call for 2 and 8, "This is Default argument"
    print("The average is:", (a+b)/2)   #you need to specify what operation you want to perform

#Call function
average(2)   #here it will assign 2 to a and b will be taken by default as 1
```

```
The average is: 1.5
```

In [5]:
```
# Create a function with name average
def average(a=5, b=1):  #a and b are the arguments
    #even the arguments in function def are 5 and 1 but it will call for 2 and 8, "This is Default argument"
    print("The average is:", (a+b)/2)   #you need to specify what operation you want to perform

#Call function
average(b=2)   #here it will assign 2 to b and a will be taken by default as 5 as already defined
```

```
The average is: 3.5
```

In [19]:
```
def name (first_name, middle_name, last_name):
    print(first_name, middle_name, last_name)

name("Syeda", "Faiza", "Iqbal")
```

```
Syeda Faiza Iqbal
```

In [20]:
```
def name (first_name, middle_name= "Faiza", last_name="Iqbal"):
    print(first_name, middle_name, last_name)

name("Syeda") #Here it is taking Firstname as called but middle and lastname as already defined
```

```
Syeda Faiza Iqbal
```

In [21]:
```
#Keyword arguments
#You can change the order of arguments here

def average(a=5, b=1):  #a and b are the arguments
    print("The average is:", (a+b)/2)

#Call function
average(b=2, a=8)  #It does not care about the order and take b as  2 and a as 8 and perform average function
```

```
The average is: 5.0
```

In [24]:
```
#Required arguments
def average(a, b=1):  #a and b are the arguments
    print("The average is:", (a+b)/2)

#Call function
average(a=8) #You have and have to give the value of a it is required at any cost but b can be taken as already

#average(a=8, b=2)  #If you give b while calling it will take the new value
```

```
The average is: 4.5
```

In [37]:
```
#Variable length arguments
def mean (*numbers):   #This staric means it can take as many arguments of variable length
    print(type(numbers)) #it will take vaues of num as a tuple
    sum = 0
    for i in (numbers):
        sum = sum + i

    print ("The mean is:", sum / len(numbers))
```

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

```
<class 'tuple'>
The mean is: 3.0
```

In [36]:
```
#Variable length arguments
def sum_all (*num):  #it will take vaues of num as a tuple
    print(type(num))
    sum = 0
    for i in (num):
        sum = sum + i

    print("Sum is:", sum)

sum_all(1,2,3)
```

```
<class 'tuple'>
Sum is: 6
```

In [41]:
```
#Variable length arguments
def mean (*numbers):   #This staric means it can take as many arguments of variable length
    print(type(numbers)) #it will take vaues of num as a tuple
    sum = 0
    for i in (numbers):
        sum = sum + i

    return sum / len(numbers)

c = mean(1,2,3,4,5)
print(c)
```

```
<class 'tuple'>
3.0
```

In [43]:
```
#Return statement is used to return the value of the expression back to the calling function

def summ (a, b):
    return a + b  #it is returning the sum
x = summ(2,3)     #Storing in c
print(x)          #and printing it as a value of c
```

```
5
```

In [40]:
```
#Keyword arbitrary arguments
#The function accesses the arguments by passing them as a form of dictionary
def name(**name):
    print(name["fname"], name["mname"], name["lname"])
name(fname="Syeda", mname="Faiza", lname="Iqbal")
```

```
Syeda Faiza Iqbal
```

In [ ]:

In [ ]: