Types of parameters or arguments

* positional arguments

- By default all formal parameters acts as positional parameters.
- If any function defined with positional parameter then order of parameter must be maintained else we will get wrong result.
- count of parameter must be maintained otherwise we will get an Error.

Example:

```
def myFun(name,age,city):
    print("Name is ",name)
    print("Age is : ",age)
    print("city is : ",city)
    print("-----------------------")

#calling
myFun("Ramesh",23,"hyd")
'''
myFun("Ramya",34)
TypeError: myFun() missing 1 required
positional argument: 'city'   '''

myFun("kadapa",45,"Roja")
output:
Name is  kadapa
Age is :  45
city is :  Roja
-----------------------
```

- keyword arguments
  - All the formal parameter works like even keyword arguments

- keyword arguments is the process of passing the values to formal parameter by using their names, in this case order not matter, but count of the parameter should be maintained.

Example of keyword args:

```
def myFun(name,age,city):
    print("Name is : ",name)
    print("Age is : ",age)
    print("city is : ",city)
    print("==================")

#calling
myFun("ramesh",34,"Hyd") #positional arguments
myFun(age=32,name="sai",city="kmm") #keyword args

myFun("raj",45,city="kadapa")
```

"'based on the application request we can pass the values to the formal parameter either by using positional parameter or by using keywords if required both, but keyword arguments must be followed by positional Parameters '"

#myFun(name="raj",age=45,"kadapa") Error

3.positional only arguments

> if any parameters which are declared left side of the / parameter are act as positional only parameter, i.e we must pass the values to that parameter by using their positions only.

Example:

```
def myFun(name,age,city,/):
    print("Name is : ",name)
    print("age is : ",age)
    print("city is : ",city)
    print("================")

#calling
myFun("ramesh",34,"hyd")

myFun(name="Sai",city="kmm",age=22)
'''TypeError: myFun() got some positional-only
arguments passed as keyword arguments:
'name, age, city'
'''
```

* keyword only arguments

1.If any formal parameters which are declared right side of * parameter acts as keyword only arguments. i.e we must pass the values to the formal parameter by using their names only.

Example:

```
def myFun(*,name,age,city):
    print("Name is : ",name)
    print("age is : ",age)
    print("city is : ",city)
    print("================")

#calling
myFun(name="Raj",age=22,city="kadapa")

myFun("james",22,'hyd')
'''
TypeError: myFun() takes 0 positional
arguments but 3 were given
'''
```

IQ:

```
def myFun(*,a,b,/):  #invalid
    pass
```

IQ:

```
def myFun(a,b,/,c,d,*,e):
    print(a,b,c,d,e,sep=' <<>> ')

#Here a,b are positional only arguments
#c,d are either positional or keyword arguments
#e is the keyword only argument

#calling
myFun(10,20,30,40,e=50)
myFun(10,20,c=30,d=40,e=50)
#myFun(a=10,b=20,c=30,d=30,50) Error
```

* default arguments

1. it is the process of defining the formal parameter with some values.
2. the default parameter are acts as optional parameter
3. the default arguments will takes place whenever we fail to pass the values the formal parameter.
4. default parameters are replaced by the actual arguments

Example:

```
def myFun(name,age=99,city="Hyd"):
    print("Name is : ",name)
    print("Age is : ",age)
    print("City is :",city)
    print("==================")

#calling
myFun("Ravi",23,"kadapa")
myFun(name="Raj",age=22,city="kmm")
myFun("Ram",23)
myFun("Ramya")
myFun()
TypeError: myFun() missing 1 required
positional argument: 'name'
```

Varargs Parameter

- It is the process of declare the formal parameter with * symbol
- Varargs internally works as tuple collection
- Varargs can take 0 arguments to N.no.of arguments
- Varargs can also take with other types of parameter but in this case varargs should be the last parameter

Example:

```
#varargs parameter

import time

def myAdd(*x):
    time.sleep(1)
    print("Type of x : ",type(x))
    print("Data of x : ",x)
    print("-"*30)

#calling
myAdd(10,20)
myAdd(10,20,30,40,50)
myAdd(10)
myAdd()
```

Example 2:

```
#varargs parameter
import time

def myAdd(*x):
    s=0
    for i in x:
        s=s+i
    print("Sum is : ",s)
    print("-"*30)

#calling
myAdd(10,20,30,40,50)
myAdd(10,20,30)
myAdd(10,20)
myAdd(10)
myAdd()
```

Example 3:

```
import time

def result(name,*marks):
    print("Name is : ",name)
    print("Marks : ",marks)
    print("total is : ",sum(marks))  #sum(iterable) -> int
    print("- "*30)

#calling
result("Sudha",50,40,50)
result("manasa",40,50)
result("bunny",30)
result("nani")
```

- Kwargs Parameters
  - Kwargs parameters are the process of declaring the formal parameters with ** symbols
  - Internally kwargs works as “dict” collections
  - It will take 0-items to N-no of.items

- Kwargs can also takes with other type of parameter but in this case also kwargs must be last arguments

Examples:

```
import time

def myStudent(**x):
    for item in x.items():
        time.sleep(1)
        k,d=item  #tuple unpacking
        print(k,d,sep='<<>>')

    print("-" *30)

#calling
myStudent(sno=101)
myStudent(sno=102,sname="Ramesh")
myStudent(sno=103,sname="Madhu",scity="Khammam")
myStudent( )
```