Variable Length Arguments

- The number of arguments a function takes is pre defined
- When default arguments are given we have options to pass parameter
- In python we can take as many parameters we want in a function by using variable length arguments , this is taken in the form of a tuple but the parameter name should start with *

syntax : def fun (* args) :
Print (args)

```
def fun1(*args):
    print(type(args), args)

fun1()
fun1(10, 20)
fun1(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
fun1(10, 'hello', 27.5, True)
```

Output :

```
:class 'tuple'> ()
:class 'tuple'> (10, 20)
:class 'tuple'> (10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
:class 'tuple'> (10, 'hello', 27.5, True)
```

- If there are more parameter given before *args then it is compulsory to give its value during function call for *args its optional.

```
def fun1(a, b, *args):
    print(a, b, args)

fun1()
fun1(11, 22)
fun1(11, 22, 33, 44, 55, 66)
```

Output :

```
TypeError: fun1() missing 2 required positional arguments: 'a' and 'b'
```

- If you have a sequence like list, tuple, string then you can unpack that in a function call using *

```
def fun2(a, b, c):
    print(a, b, c)

s1 = 'sky'
fun2(*s1)
```

Output:

```
s k y
```

- If a function is returning multiple values it is possible in python then all values can be unpacked in different variable or take them in a tuple

```
def fun3(a, b, c):
    return a+1, b+1, c+1

x, y, z = fun3(10, 20, 30)
print(x, y, z)
```

Output :

```
11 21 31
```