# Passing variable length argument to a Function

***args and **kwargs in python**

- It is not necessary to write *args or **kwargs. Only the * (aesteric) is necessary. We could have also written *var and **vars.
- *args and **kwargs allow you to pass a variable number of arguments to a function.
- *args is used to send a **non-keyworded** variable length argument list to the function

```
def test_var_args1(f_arg, *argv):
   print ("first normal arg:", f_arg)
   for arg in argv:
      print ("another arg through *argv :", arg)

test_var_args1('Training','Python','Third','Day')
```

**kwargs allows you to pass **keyworded** variable length of arguments to a function. You should use **kwargs if you want to handle **named arguments** in a function.

```
def test_var_args2(**kwargs):
   if kwargs is not None:
      for (key,value) in kwargs.items():
         print ("%s == %s" %(key,value))
test_var_args2(name="Test")
test_var_args2(name="Praveen",profession="Trainer")
```

**Using *args and **kwargs to call a function**

```
def test_args_kwargs(arg1, arg2, arg3):
   print ("arg1:", arg1)
   print ("arg2:", arg2)
   print ("arg3:", arg3)
args = ("two", 3,5)
test_args_kwargs(*args)
kwargs = {"arg3": 3, "arg2": "two","arg1":5}
test_args_kwargs(**kwargs)
```

**Order of using *args **kwargs and formal args**

if you want to use all three of these in functions then the order is

```
def test_var_args(f_arg, *argv,**kwargs):
   print ("first normal arg:", f_arg)
   if argv is not None:
      for arg in argv:
         print ("another arg through *argv :", arg)
   if kwargs is not None:
      for (key,value) in kwargs.items():
         print ("%s == %s" %(key,value))
test_var_args('Training')
```

```
test_var_args('Learning','Python','Third','Day')
test_var_args('Programing', 'Filehandling', 'Python',name="Praveen",profession="Trainer")
```