Example:

```
def add(*n):
    s=0
    for value in n:
        s=s+value
    return s

def main():
    res1=add(10,20)
    res2=add(1,2,3,4,5)
    res3=add(100,200,300,400,500)
    print(res1,res2,res3)

main()
```

Output:

30 15 1500

Example:

```
# custom print function
import sys
def display(*values,sep=' ',end='\n',file=sys.stdout):
    l=list(map(str,values))
    str1=sep.join(l)
    str1=str1+end
    file.write(str1)


def main():
    display(10,20)
    display(10,20,30,sep=',')
    display()
    display(10,20,30,end=':')
main()
```

Output:

10 20
10,20,30

10 20 30:

Keyword arguments
Function with keyword arguments receives key and value.
Keyword arguments of type dictionary.
Keyword arguments are prefix with **
Keyword arguments receive 0 or more values.
A function can be defined with one keyword argument

Syntax:

```
def  <function-name>(**kwargs):
        statement-1
        statement-2
        statement-3
```

function with keyword arguments is used to manipulate dictionary

Example:

```
def fun1(**a):
    print(a)
    print(type(a))


def main():
    fun1()
    fun1(a=10)
    fun1(a=10,b=20,c=30)
    fun1(x=100,y=200,z=300)

main()
```

Output:

```
{}
<class 'dict'>
{'a': 10}
<class 'dict'>
{'a': 10, 'b': 20, 'c': 30}
<class 'dict'>
{'x': 100, 'y': 200, 'z': 300}
```

<class 'dict'>

Example:

```
def add(*values,**kwargs):
    s=0
    for value in values:
        s=s+value

    for value in kwargs.values():
        s=s+value

    return s


def main():
    res1=add(10,20,30)
    res2=add(a=10,b=20,c=30)
    res3=add(10,20,30,x=40,y=50)
    print(res1,res2,res3)

main()
```

Output:

60 60 150

Example:

```
def display_books(**kwargs):
     for key,value in kwargs.items():
         print(key,value)

def main():
    books_dict={'python':'rossum',
                'c':'dennis richie',
                'java':'james'}
    display_books(**books_dict)
```

→ **** is used for unpacking dictionary items**

```
main()
```

Output:

======== RESTART: F:/python6pmaug/funtest33.py =======
python rossum
c dennis richie
java james

Nested Function
Function within function is called nested function or inner function.

What is need of inner functions?
1. To divide the functionality of one function into number of sub functions
2. Nested functions are used to develop special functions
    a. Decorator
    b. Closure

Syntax:
```
def  <function-name>(arg1,arg2,arg3,…):  → Outer function
        statement-1
        statement-2
        def <function-name>(arg1,arg2,arg3,…): → Inner function
                statement-1
                statement-2
```

Example:
```
def fun1():
   print("outer function")
   def fun2():
      print("inner function")
   fun2()



fun1()
```

Output:
outer function
inner function