Example:

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



res1=add(10,20,30,40,50)
res2=add(a=10,b=20,c=30,d=40,e=50)
res3=add(10,20,30,a=40,b=50)
print(res1,res2,res3)
```

Output

```
150 150 150
```

Example:

```
def display_dictionary(**kwargs):
    for key,value in kwargs.items():
        print(f'{key}---->{value}')



stud_dict={'naresh':'python',
           'suresh':'java','ramesh':'oracle','kishore':'c++'}
display_dictionary(**stud_dict)

sales_dict={'jan':50000,'feb':65000,'mar':75000}
display_dictionary(**sales_dict)
```

Output

```
naresh---->python
suresh---->java
```

ramesh---->oracle
kishore---->c++
jan---->50000
feb---->65000
mar---->75000

Nested Function
Defining function inside function is called nested function or inner function.

Use of nested functions
1. Hiding functionality of one function inside function
2. For developing special functions in python
    a. Decorator
    b. Closures

Syntax:
```
def <outer-function-name>([parameters]):
    statement-1
    statement-2
    def <inner-function-name>([parameters]):
        statement-1
        statement-2
```

Points to remember
1. Inner function is accessible within outer function but cannot accessible outside outer function

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

```
fun1()
fun2()
```

Output

```
inside outer function
Traceback (most recent call last):
  File "E:/python5pmjun/test236.py", line 9, in <module>
    fun2()
NameError: name 'fun2' is not defined. Did you mean: 'fun1'?
```

2. Inner function can access local variables of outer function but outer function cannot access local variables of inner function

Example:

```
def fun1():
    x=100 # Local Variable of fun1
    def fun2():
        print(f'Local variable of fun1 x={x}')
    fun2()

def fun3():
    def fun4():
        x=30 # Local variale of fun4

    print(x)


fun1()
fun3()
```

Output

```
Local variable of fun1 x=100
Traceback (most recent call last):
  File "E:/python5pmjun/test237.py", line 16, in <module>
    fun3()
  File "E:/python5pmjun/test237.py", line 11, in fun3
    print(x)
```

NameError: name 'x' is not defined

3. Inner function can access local variables of outer function directly but cannot modify it.

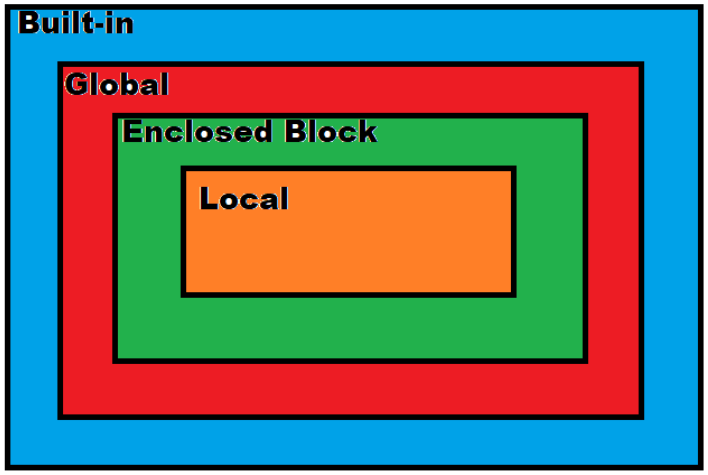
```
def fun1():
   x=100 # L. V. fun1
   def fun2():
      x=300 # L.V. fun2
      print(x)
   fun2()
   print(x)


fun1()
```

Output

300
100

What is LEGB in python?

The LEGB stands for Local, Enclosing, Global and Built-in.
Python resolves names using the LEGB rules.
The LEGB rule is a kind of name lookup procedure, which determines the order in which Python looks up names.



Example:

```
x=100 # Global Variable
def fun1():
    y=200 # Local Variable fun1
    def fun2():
        z=300 # Local variable of fun2
        print(x)
        print(y)
        print(z)
        print(__name__)
       # print(p) NameError
    fun2()


fun1()
```

Output

```
100
200
300
__main__
```

nonlocal keyword

Inner function can modify or update the local variable of outer function using nonlocal keyword.

Syntax:

Nonlocal variable-name,variable-name

After this declaration, variable list is referred as nonlocal variables.

Example:

```
def fun1():
    x=100 # L.V. of fun1
    def fun2():
```

```
        nonlocal x
        x=300

    print(x)
    fun2()
    print(x)

fun1()
```

Output

100
300

Example:

```
def calculator(n1,n2,opr):
        res=0 # L.V
        def add():
            nonlocal res
            res=n1+n2
        def sub():
            nonlocal res
            res=n1-n2
        def multiply():
            nonlocal res
            res=n1*n2
        def div():
            nonlocal res
            res=n1/n2

        if opr=='+':
            add()
        elif opr=='-':
            sub()
        elif opr=='*':
            multiply()
        elif opr=='/':
            div()
```

```
    else:
        res="ERR"

    return res



# main
num1=int(input("Enter First Number "))
num2=int(input("Enter Second Number "))
opr=input("Enter Operator ")
result=calculator(num1,num2,opr)
print(f'{num1}{opr}{num2}={result}')
```

Output

```
Enter First Number 6
Enter Second Number 3
Enter Operator *
6*3=18

Enter First Number 5
Enter Second Number 2
Enter Operator -
5-2=3

Enter First Number 9
Enter Second Number 2
Enter Operator /
9/2=4.5
```