Functions

A function is a reusable code which is used to organize the structure of code . we can call functions multiple time in same program.

keyword **def** is used to start function with name and parentheses (( )).

The code block within every function starts with a colon **(:)** and is indented.

Syntax

#defining function

```
def fun1(  ):
  print (“”)
```

Now you can call printme function

```
fun1("Hii...i am your function!")
fun1("i am your second function")
```

```
def add(a, b):
    print ("ADDING" ,a+b)

add(10,20)
add(2,3)
```

Output:-

```
Adding  30
Adding   5
```

Function Arguments

You can call a function by using the following types of arguments:

- Required arguments
- Keyword arguments
- Default arguments
- Variable-length arguments

Required Argument

```
def add(a1, a2):
    print ("ADDING" ,a1+a2)


add(10,20)
add(2)
```

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
=============== RESTART: D:\python programs\4functions\fun.py ===============
ADDING 30
Traceback (most recent call last):
  File "D:\python programs\4functions\fun.py", line 8, in <module>
    dd=f1(2)
TypeError: f1() missing 1 required positional argument: 'a2'
>>>
```

Keyword arguments

There are two ways to pass arguments to method: positional arguments and Keyword arguments.

```
def f1(name,age,color):
   print("my name ,age and color  is ",name,age,color)


 f1(name='ABC',color='white',age='20')
#keyword argument(name with value)
f1('XYZ',color='black',age='30')
#positional and keyword argument
```

Default arguments

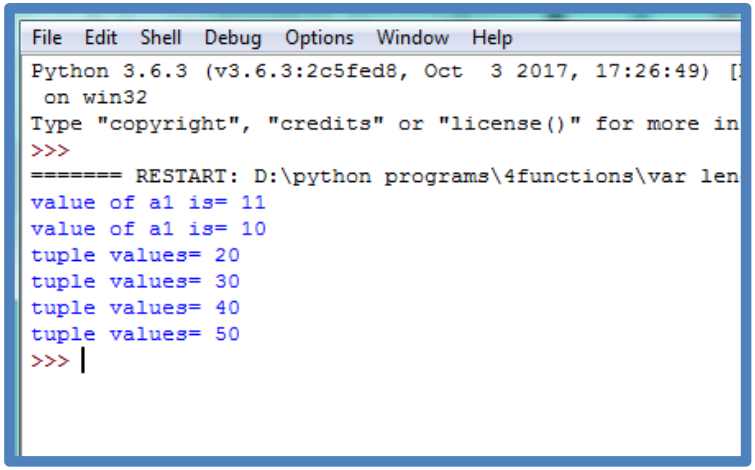
A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument.

```
# Function definition is here
def data(a1, a2=20):
 print("value of a1 is",a1 )
 print("value of a2  is", a2)


data("ASD",44)
data("XYZ")
```

```
Type "copyright", "credits" or
>>>
=============== RESTART: D:\py
value of a1 is ASD
value of a2 is 44
value of a1 is XYZ
value of a2 is 20
>>>
```

Variable length arguments

```
def printinfo( a1,*a2):

  print ("value of a1 is=",a1)
  for var in a2:
    print ("tuple values=",var)
```

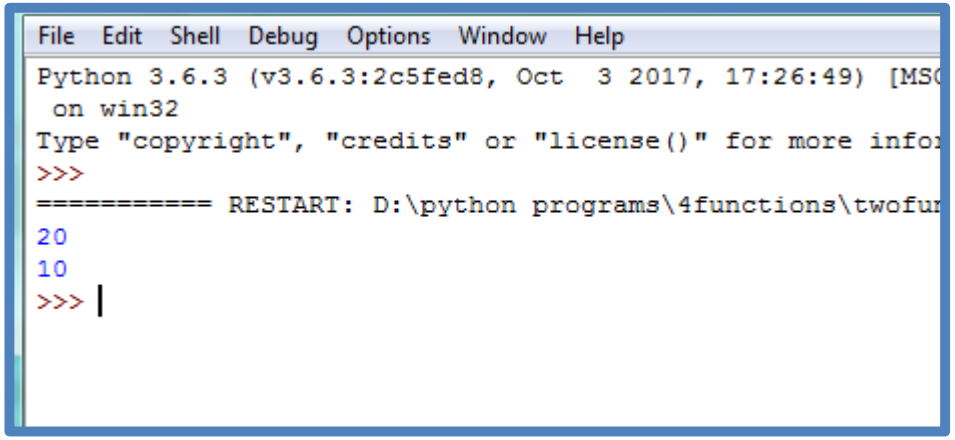



```
printinfo(11)
printinfo(10,20,30,40,50)
```

#This tuple remains empty if no additional arguments are specified during the function call.

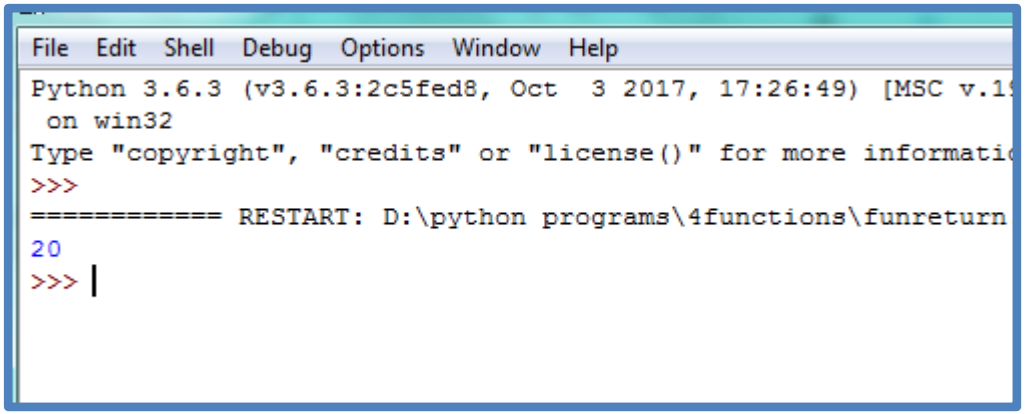
Call one function into another function

```
def f1():
  var1=10
  print(var1)

def f2():
  var2=20
  print(var2)
  f1()

f2()
```



return value

```
def f1(a1,a2):
    return a2


d=f1(10,20)
print(d)
```



Scope of Variables

All variables in a program may not be accessible at all locations in that program. This depends on where you have declared a variable.

The scope of a variable determines the portion of the program where you can access a particular identifier.

There are two basic scopes of variables in Python –

- Global variables
- Local variables

Global vs. Local variables

Variables that are defined inside a function body have a local scope, an those defined outside have a global scope. This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions. When you call a function, the variables declared inside it are brought into scope.Following is a simple example –

```
x1 = 100
def sum():
    x2 = 200
    y1=x1+x2
    return y1

r1=sum()
print(r1)
print(x1)
print(y1)
```

```
================ RESTART: C:\Users\Piford\Desktop\welcome.py ================
300
100
Traceback (most recent call last):
  File "C:\Users\Piford\Desktop\welcome.py", line 25, in <module>
    print(y1)
NameError: name 'y1' is not defined
>>>
```