Functions and Modules

Syntax

- In Python a function is defined using the def keyword:

```
# function definition
def function_name(parameters):
        statement1
       statement2
        …

# Driver Code
        statement1
       statement2
        …
```

Example

```
def function1():
        print("Inside function")

function1()
```

Example

```
def display(name):
        print('Hello!',name)

for i in range(0,3):
        n = input("Enter Name")
       display(n)
```

Input Parameters:
Addition without using function

```
i = int(input("Enter Num1: "))
j = int(input("Enter Num2: "))

res = i + j
print(res)
```

Input Parameters
Addition using function

```
def add(a,b):
    res=a+b
    print("Addition = ",res)


i=int(input("Enter Num1: "))
j=int(input("Enter Num2: "))
add(i,j)
```

return statement

```
def add(a,b):
    res=a+b
    return res


i=int(input("Enter Num1: "))
j=int(input("Enter Num2: "))
res = add(i,j)
print("Result = ",res)
```

return multiple values

```
def add(a,b):
        res=a+b
        a = a*10
        return res, a, b            # forms tuple

i=int(input("Enter Num1: "))
j=int(input("Enter Num2: "))
res = add(i,j)
print("Result = ",res,type(res))
```

Default Arguments

```
def add(a,b=20):
    res=a+b
    return res
```

```
i=int(input("Enter Num1: "))
j=int(input("Enter Num2: "))

res = add(i,j)
print("Result = ",res)

res = add(i)
print("Result = ",res)
```

Arbitrary Arguments

 # If we want to make it flexible for n-number of arguments as input parameters

```
def display(*name):
        for n in name:
                print('Hello!',n)


display('AAA','BBB')
display('AAA','BBB','CCC')
```

Arbitrary Arguments: Similar Data type

```
def add(* a):
    res=0
    for i in a:
            res = res + i
    print("Sum = ",res)
```

add(10,20,30)

add(1,2,3,4,5,6,7,8)

Arbitrary Arguments: varied data type

```
def add(* a):
    res=0
     for i in a:
                res = res + l
    print("Sum = ",res)
```

add(10,20,30)

add(1,2,3,4,5,6,7,8)

add(1.9,2.4,3.55,5)

Arbitrary Arguments: varied data type

```
def add(* a):
    res=0
    for i in range ( 1 , len(a)):
        res=res+a[i]
    print( a[0] , res)

add("Addition = ",10,20)
add("Sum = ",3,5,10,12,5)
```

Common Function for all collection data types

```
def disp(L):
    for i in L:
        print(i,end=" ")
    print()
```

```
L1 = [ 10, 20, 30, 40, 50 ]
disp (L1)

T1 = ( 1, 2, 3, 4 )
disp (T1)

S1 = { 4, 5, 6, 8 }
disp (S1)
```

Function for all collection data types

```
def disp(L):
    for i in L:
        print(i,end=" ")
    print()
```

```
# Display keys by default:

D1 = { 1:10, 2:20, 3:30 }

disp (D1)

# To display values:

disp (D1.values())
```

Example: Addition of list elements

```
def add(L):
    res=0
    for i in L:
            res = res + i
    return res
```

```
L1=[ ]
ch='y'
while ch != 'n':
        i = int(input("Enter Num: "))
        L1.append(i)
        ch = input("Want to continue y/n")
res = add(L1)
print(res)
```

Example: Addition of Dictionary values

```
def add(L):
    res=0
    for i in L:
            res = res + i
    return res
```

```
L1={}
ch='y'
while ch != 'n':
        i = int(input("Enter Key: "))
        j = int(input("Enter Value: "))
        L1[i]=j
       ch = input("Want to continue y/n")
res = add(L1.values())
print(res)
```

Arbitrary Arguments: varied data type

```
def disp(*L):
        for i in L:
            print(i)


L1 = [10,20,30]
i = 'abc'
j = 10.15
disp ( L1, i, j )
```

Recursive function

```
def fact(num):
        if num == 1:
            return 1
         else:
            return num*fact(num-1)


res=fact(5)
print(res)
```

Modules

- What is a Module?
  - Consider a module to be the same as a code library.
  - A file containing a set of functions you want to include in your application.

Create a Module

- save the code you want to add in module with the file extension .py
- E.g. MyModule1.py:

```
def display(name):
    print('Hello!',name)
```

- Use a Module

```
import MyModule1
MyModule1.display(“John”)
```

Built-in Modules

- There are several built-in modules in Python, which you can import whenever you like.

- dir() Function: There is a built-in function to list all the function names (or variable names) in a module.

Example: Import the datetime module and display the current date

```
import datetime

x = datetime.datetime.now()
print(x)

# The date contains year, month, day, hour, minute, second, and microsecond.
```

Example: Return the year

```
import datetime

x = datetime.datetime.now()

print(x.year)
```

Example: Create a date object

```
import datetime

x = datetime.datetime(2019, 3, 19)

print(x)
```