Day13_Functions

- **A function is a set of code which is already defined and compiled in the python environment and is used for performing a particular task**

Ex: print()
type()
id()

-As a programmer we need to call the function to be able to process the task
Ex: calling or invoking
print("Hello") --> Hello
type(45) --> int
id(45) --> 165897233

All the above are said to be predefined functions i.e., the functions that are already developed and we need to just call them.

In addition to the predefined function a programmer can have his own functions which are said to be Userdefined functions

The user defined functions are said to be defined-compiled-called

Definition of the Function
Syntax:

```
def functionname(formalparameters or arguments):
    '''docstring'''
     statements
     return data
```

Calling the function:
functionname(actualparameters or arguments)

There are 4 combinations that are used for defining the functions

1. Functions without parameters and without return
2. Functions with parameter and without return
3. Functions without parameter and with return
4. Functions with parameter and with return

1. Functions without parameters and without return

```
def add():
    '''
    It add two numbers
    Parameter
    ---------
    None
    Returns
    -------
    None.
    '''
    var1 = 10
    var2 = 5
    print(var1+var2)

help(add)

add()
#15
```

2. Functions with parameter and without return

```
def cube(var):#formal parameter or argument
    print(var**3)


cube(5)#5 --> actual parameter or argument
#125

cube(5,3)
```

#TypeError: cube() takes 1 positional argument but 2 were given

3. Functions without parameter and with return

```
def rectarea():
    length =float(input("Enter Length:"))
    breadth=float(input("Enter Breadth:"))

    area = length * breadth
    return area

rectarea()
#281500.0

#Find the total cost of flooring @5$

print("Total cost=",rectarea()*5,"$")
```

4. Functions with parameter and with return

```
def power(b,e):
    return b**e

power(3,2)
#9
```

A function can also return multiple values after performing the task. These values are returned as tuple and we can even assign them to individual variables depending on the number of return values

```
def arithop(a,b):
    return a+b,a-b,a*b,a//b,a/b,a%b,a**b

arithop(3,2)
#(5, 1, 6, 1, 1.5, 1, 9)

add,sub,mul,idiv,fdiv,md,po = arithop(3,2)
```

add	int	1	5
fdiv	float	1	1.5
idiv	int	1	1
md	int	1	1
mul	int	1	6
po	int	1	9