Date : 22/12/25
—---------------------------

To-Do:
- Function in python
—----------------------------

<u>FUNCTIONS IN PYTHON:</u>

- ➔ Functions are a block of statement that do a specific task.
- ➔ We usually write code that is reused a lot so that we don't have to write the same code again and again
- ➔ Characteristics :
  - ◆ Code reusability
  - ◆ Modularity
  - ◆ Readability
  - ◆ Maintainability

Defining a Function :

- ➔ We use 'def' keyword to define a function
- ➔ Syntax:
  - ◆
    ```
    def function_name(parameters):
            #statement
            return expression
    ```
- ➔ Example:
  - ◆
    ```
    def greeting():
            print("welcome coder!!!")
    ```

Function Arguments:

- ➔ Arguments are the values passed inside the function parenthesis
- ➔ A function can have any number of arguments separated by commas
- ➔ """Docstring""" is a special string written inside the functions that has the description of what the function does
- ➔ Syntax:
  - ◆
    ```
    def function_name(parameters):
            """Docstring"""
            #body of the function
            return expression
    ```

- Example:
    - def evenodd(n):
      """"This function shows if number is even or odd""""
      if(n%2==0):
      return "even"

      else:
      return "odd"

```
def evenodd(n):
        """This function shows if number is even or odd"""
        if(n%2==0):
                return "even"

        else:
                return "odd"

print (evenodd(16))
print (evenodd(5))
```

- **TYPES OF FUNCTION ARGUMENT:**
    - Default arguments

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

      Example:

```
        def myfun(x,y=44):
                print("x:",x)
                print("y:",y)
        myfun(10) #so in the function call i will put the default value and
this will go to 'x' since y value is specified.
```

- Keyword arguments

  Here values are passed by explicitly specifying the parameter names, so order doesn't matter

  Example:

```
        Def myfun(fname,lname):
                print("First name:",fname)
                print("Last name:",lname)
        myfun(fname="Mahek", lname="junnedi")
        myfun(lname="junnedi", fname="Mahek")
```

- Positional arguments

  In this values are assigned to parameters based on their order in the function call

  Example:

```
        def myfun(name,age):
                print("my name is:",name)
                print("my age is:",age)
```

```
myfun(“Mahek”,18) # name= mahek, age= 18
myfun(18,”Mahek”) #name=18,  age= mahek
```

- <u>Arbitrary arguments:</u>
  - *args: non keyword argument. args is a tuple inside function
    It collects values without names.
  - **kwargs: keyword argument. Kwargs is a dictionary inside function
    It collects values with names.

Example:

```
def myfun(*args, **kwargs):
        print(“non keyword *args:”)
        for arg in args:
                print(arg)

        print(“keyword **kwargs:”)
        For key,value in kwargs.items():
                print(f”{key} == {value”})
myfun(“hello” , “Mahek” , “course” : “Internship” , “Duration” : “6 Months”)
```

Function Within Functions:

A function defined inside another function is called an inner function or nested function. It can access variables from the enclosing functions scope and is often used to keep logic protected and organized.

Example:

```
Def f1():
        s = “i like reading”

        Def f2():           #defining 2nd function in 1st function
                print(s)

        f2()                #calling the nested function inside the first functio
f1()                        #calling 1st function from outside
```

Anonymous function:

A function without a name is called an anonymous function . lambda keyword is used to define/create these anonymous functions.

Use case of lambda:

- Used for one time use function
- Short one line functions
- Useful simple calculations
- Used for temporary functions
- Used when working with data frames

Example:

```
def cube(x) : return x*x*x   # without lambda
Cube_l : lambda x : x*x*x  # with lambda
print(cube(7))
print(cube_l(7))
```

Methods used with lambda:

➔ map() : when we want to apply logic on all the elements of list

- ◆ Example:

```
Num = [1,2,3,4,5]
Result = list(map(lambda x: x*4 , Num))
print(Result)
```

➔ filter() : used when we want filtered items from the list

- ◆ Example:

```
Even = list(filter(lambda x: x%2==0, Num))
```

Return Statement:

Return statement ends a function and sends back a value to the caller. It can return any data type, multiple values in a tuple.

Example:

```
def sq(x):
        return(x**2)
print(sq(6))
```

Pass by Value	**Pass by Reference**
Copy of value passed	Reference passed
Original variable not affected	Original variable affected
Immutable objects	Mutable objects
int, float, string	list, dict, set

Example:

```
# Function modifies the first element of list
def myFun(x):
    x[0] = 20

lst = [10, 11, 12, 13]
myFun(lst)
print(lst)   # list is modified

# Function tries to modify an integer
def myFun2(x):
    x = 20

a = 10
myFun2(a)
print(a)     # integer is not modified
```

Recursive function:

A recursive function is a function that calls itself to solve a problem.

Example:

```
def factorial(n):
        if n == 0:
                return 1
        else:
                return n * factorial(n - 1)

print(factorial(4))
```