1-10-2024

Training Day- 11

FUNCTIONS: A function is a reusable block of code that performs a specific task. Functions allow you to break your program into smaller, modular pieces, making the code easier to read, maintain, and reuse.

Defining a Function
In Python, functions are defined using the def keyword.
Syntax:
python
Copy code

```
def function_name(parameters):
    """Optional docstring"""
    # Function body
    return value  # Optional
```

* **FUNCTION_NAME:** The name of the function, following Python naming conventions.
* **PARAMETERS**: Input values passed to the function (optional).
* **RETURN:** Returns a value from the function (optional).

PL (Presentation Layer): Responsible for user interaction
BLL (Business Logic Layer): Responsible for writing the business logic

Camel Nomenclature: These are guidelines designed by Developers community to take logical identifier names.

1. Function Name should start with small letter and rest all trailing words should start with capital letters
Example:
add_Customer() or addNewCustomer()

2. Class Name should be in title case: First letter of all words should start with Capital Letter
MyCustomer   or M_Customer

Above 2 guidelines are as per camel nomenclature
rest other guidelines for identifier names:

GUIDELINES:

1. Try to use underscore in between words
2. Variable names should not have capital letter
3. Identifiers names should be logical names ie they should represent the purpose of identifier.

Ex: Variable names
cus_id, cus_age, cus_name, cus_mob, cus_email, cus_add, cus_city, cus_dob, cus_state, cus_gender......

FUNCTIONS:

The formal parameters and actual parameters names can be same or can be different as per the interest or requirement.

Whenever we call a function in any programming langauge, the actual parameters are assigned to formal parameters.

Now if we want a program for calculator then try to make it using 2 layers.
"""

> BLL(Business logic layer)

```
# def add(a,b):      #a,b formal parameters
#     r=a+b
#     return r
# def sub(a,b):
#     return a-b
```

> PL(presentation layer)

```
# a,b,c,d=1,2,3,4     #This statement can be a part of PL or BLL
# r1=add(a,b)
# r2=sub(c,d)
# print(r1,r2)

"""
```

In Python return statement is optional.
The functions which return nothing in the program then they return None value.

```
"""


# #New Program
# def func1(a,b):
#     r=a+b
#
# r=func1(5,7)
# print(r)

# #New Program
# s=input("Enter Your Name")

# #New Program
# s=print("CETPA")
# print(s)

# #New Program
# def add(a,b):
#     r=a+b
#     return r
# u,v=5,7
# s=add(u,v)      #a=u, b=v
# print(s)
"""u and v are actual variables in above program.
a and b are formal variable in above program
"""
```

# #New Program: Not a better approach to create a function

```
# def add():
#     r=a+b
#     return r
# a,b=5,7
# s=add()
# print(s)
```

# #New Program: Better approach

```
# def add(a,b):      #a,b,r are called local variables
#     r=a+b
#     return r
# a,b=5,7
# s=add(a,b)      #a,b,s are called global variables
# print(s)
```

"""

We can access global variables inside functions directly but we can't access local variables outside functions.

…New Program….

```
# def func1():
#     a=5        #Local Variable
#     print(a)
# func1()
# print(a)       #NameError: name 'a' is not defined
```

…New Program….

```
# def func1():
#  print(a)       #a global variable
# a=5
# func1()
# print(a)           #a global variable
```

"""

If there are two variables with same name, one inside function and one outside function then inside function local variable will be accessed and outside function global variable will be accessed. In following program, why global variable is not modified, because in python, how variables are created? By assigning the value. If we have a variable outside function (Global variable) and we try to access it inside function then it is accessible, but the moment we try to modify the global variable inside function, what happens, a new local variable is created. Still if we want to modify global variable inside function.

…..New Progra,……

```
# def func1():
#     a=5           #
#     print(a)
# a=7
# func1()
# print(a)
```

….New Program….

```
# def func1():
#     global a
#     a=5           #
#     print(a)
# a=7
# print(a)
# func1()
# print(a)
```

…..New Program…..

```
# def func1():
#     a=5           #local variable
#     print(a)
#
# func1()
# print(a)       #NameError: name 'a' is not defined
```

àà..New Programàà.

```
# def func1():
#     global a
#     a=5           #global variable
#     print(a)
# func1()
# print(a)
```