UNIT- V

FUNCTIONS

- Group of statements that are intended to perform a specific task.

What is functions in Python?

- Performs a specific task.
- Reusability
- Modularity
- More organized and manageable
- Code maintenance will become easy.
- Code debugging will become easy.
- Reduce the length

Difference between a Function and a Method

- Functions is called using its name.
- When a function is written inside a class, it become **method**.
- Function and a method are same except their placement and the way they are called.

```
Objectname.methodname()
Classname.methodname()
```

DEFINING A FUNCTION

Syntax: **def functionname (parameters) :**

```
def functionname (parameters) :
        " " " Function Docstring " " "
        Function Statements
```

Example:

```
def sum (a,b):
        """Sum of Two Numbers"""
        c=a+b
        print("Sum = ",c)
```

- The function body contains a group of statements called ‘suite’.
- String is called a ‘doc-string’ that gives information about the function.

Significance of Indentation (Space) in Python

- Python functions don’t have any explicit begin or end like curly braces to indicate the start and stop for the function, they have to rely on indentation.
- Maintain the same indent for the rest of your code.

Example:

```
# Indentation Error: Expected an indented block
def func1():
print(“I am learning Python Function”)

# Unindent does not match any other indentation level
def func1():
        print(“I am learning Python Function”)
print(“Still in func1”)

# Get the expected output
def func1():
        print(“I am learning Python Function”)
        print(“Still in func1”)
```

CALLING A FUNCTION

- Once we have defined a function, we can call it from another function, program or even the Python Prompt.

Example:
```
sum(45,5)
sum(45.5,5.8)
```

Returning Results from a Function (The return Statement)

- The **return** statement is used to exit a function and go back to the place from where it was called.
- Return command in Python specifies what value to give back to the caller of the function.

Syntax: return [expression _list]

Example:
```
def sum (a,b):
        """Sum of Two Numbers"""
        c=a+b
        return c
    x=sum(23,56)
    print("Sum = ",x)
    y=sum(2.5,5.6)
    print("Sum = ",y)
```

Returning Multiple Values from a Function

Example:

```
def sum_sub(a,b):
    """Sum and subtraction of Two Numbers"""
    c=a+b
    d=a-b
    return c ,d
x, y=sum_sub(23,56)
print("Sum = ",x)
print("Subtraction = ",y)
```

TYPES OF FUNCTIONS

1. Built-in Functions
2. User-Defined Functions

Functions are First Class Objects

- We can use functions as perfect objects.
- It is possible to assign a function to a variable.
- It is possible to define one function inside another function.
- It is possible to pass a function as parameter to another function.
- It is possible that a function can return another function.

Example 1:

#assign a function to a variable

```
def display(str):
    return 'Hello' + str
#assign function to a variable x
x=display("Naman")
print(x)
```

Example 2:

#define a function inside another function

```
def display(str):
    def message():
        return 'Hello'
    result=message()+str
    return result
print(display("Naman"))
```

Pass By Object Reference

- Pass by value represents that a copy of the variable value passed to function and any modifications to that value will not reflect outside the function.
- Pass by reference represents sending the references or memory address of the variable to the function.
- Call by value and call by reference -: Neither of these concepts is applicable in Python.
- The values are sent to functions by means of object references.
- In python, an object can be imagined as a memory block where we can store some value like X=10.
- In this case, 10 is the object and x is the name given to that object.
- Objects are created on heap memory that depends on the RAM of our computer system.
- To know the location of the object in heap, we can use **id () function** that gives identity number of an object.

Example:

```
x=10
id(x)
print(x,id(x))
```

#passing an integer to a function

```
def modify(x):
  x=15
  print(x,id(x))

x=10
modify(x)
print(x,id(x))
```