Modularization and Classes

Python Modules:

A python module can be defined as a python program file which contains a python code including python functions, class, or variables. In other words, we can say that our python code file saved with the extension (.py) is treated as the module. We may have a runnable code inside the python module.

Modules in Python provide us the flexibility to organize the code in a logical way. To use the functionality of one module into another, we must have to import the specific module.

Example

In this example, we will create a module named as file.py which contains a function func that contains a code to print some message on the console.

Let's create the module named as file.py.

```
#displayMsg prints a message to the name being passed.

def displayMsg(name)

    print("Hi "+name);
```

Here, we need to include this module into our main module to call the method displayMsg() defined in the module named file.

Loading the module in our python code

We need to load the module in our python code to use its functionality. Python provides One types of statements as defined below.

1. The import statement

The import statement

The import statement is used to import all the functionality of one module into another. Here, we must notice that we can use the functionality of any python source file by importing that file as the module into another python source file.

We can import multiple modules with a single import statement, but a module is loaded once regardless of the number of times, it has been imported into our file.

The syntax to use the import statement is given below.

import module1,module2,…… module n

Hence, if we need to call the function displayMsg() defined in the file file.py, we have to import that file as a module into our module as shown in the example below.

Example:

```
import file;
name = input("Enter the name?")
file.displayMsg(name)
```

Python Class and Objects

As we have already discussed, a class is a virtual entity and can be seen as a blueprint of an object. The class came into existence when it instantiated. Let's understand it by an example.

Suppose a class is a prototype of a building. A building contains all the details about the floor, doors, windows, etc. we can make as many buildings as we want, based on these details. Hence, the building can be seen as a class, and we can create as many objects of this class.

Creating classes in python

In python, a class can be created by using the keyword class followed by the class name. The syntax to create a class is given below.

Syntax

```
class ClassName:
    #statement_suite
```

In python, we must notice that each class is associated with a documentation string which can be accessed by using **<class-name>.__doc__**. A class contains a statement suite including fields, constructor, function, etc. definition.

Consider the following example to create a class Employee which contains two fields as Employee id, and name.

The class also contains a function display() which is used to display the information of the Employee.

Example

```
class Employee:
    id = 10;
    name = "ABC"
    def display (self):
        print(self.id,self.name)
```

Here, the self is used as a reference variable which refers to the current class object. It is always the first argument in the function definition. However, using self is optional in the function call

Creating an instance of the class

A class needs to be instantiated if we want to use the class attributes in another class or method. A class can be instantiated by calling the class using the class name.

The syntax to create the instance of the class is given below.

<object-name> = <class-name>(<arguments>)

The following example creates the instance of the class Employee defined in the above example.

Example

```
class Employee:
    id = 10;
    name = "John"
    def display (self):
        print("ID: %d \n Name: %s"%(self.id,self.name))
emp = Employee()  // Here we are creating a object called as emp
emp.display()
```

Python Functions:

Functions are the most important aspect of an application. A function can be defined as the organized block of reusable code which can be called whenever required.

Python allows us to divide a large program into the basic building blocks known as function. The function contains the set of programming statements enclosed by {}. A function can be called multiple times to provide reusability and modularity to the python program.

In other words, we can say that the collection of functions creates a program. The function is also known as procedure or subroutine in other programming languages.

Advantage of Functions in Python

There are the following advantages of Python functions.

- By using functions, we can avoid rewriting same logic/code again and again in a program.
- We can call python functions any number of times in a program and from any place in a program.
- We can track a large python program easily when it is divided into multiple functions.
- Reusability is the main achievement of python functions.
- However, Function calling is always overhead in a python program.

Creating a function

In python, we can use **def** keyword to define the function. The syntax to define a function in python is given below.

```
def my_function():
    function-suite
    return <expression>
```

The function block is started with the colon (:) and all the same level block statements remain at the same indentation.

A function can accept any number of parameters that must be the same in the definition and function calling.

Function calling

In python, a function must be defined before the function calling otherwise the python interpreter gives an error. Once the function is defined, we can call it from another function or the python prompt. To call the function, use the function name followed by the parentheses.

A simple function that prints the message "Hello Word" is given below.

```
def hello_world():
    print("hello world")

hello_world() // Calling function
```

Functions and arguments (signature)

Required Arguments:

Till now, we have learned about function calling in python. However, we can provide the arguments at the time of function calling. As far as the required arguments are concerned, these are the arguments which are required to be passed at the time of function calling with the exact match of their positions in the function call and function definition. If either of the

arguments is not provided in the function call, or the position of the arguments is changed, then the python interpreter will show the error.

Consider the following example.

Example 1

```
    #the argument name is the required argument to the function func
def func(name):
   message = "Hi "+name;
   return message;
name = input("Enter the name?")
print(func(name))
```