Chapter -04 Using Python Libraries

1. Library:

A *Library* refers to a collection of *modules* that together *cater to specific type of needs or applications*. A *library* can have ***multiple modules*** in it.

Some commonly used Python Libraries are as listed below:

i) Python Standard Library:

This *library* is distributed with Python that contains *modules* for *various types of functionalities*. Some commonly used *modules* of Python standard *library* are:

- **math module,** which provides *mathematical function* to support *different types of calculations.*
- **Cmath module,** which provides *mathematical functions* for *complex numbers.*
- **Random module,** which provides *functions for generating pseudo-random numbers.*
- **Statistics modules,** which provides *mathematical statistics functions.*
- **Urllib module,** which provides *URL handling*, functions so that we can *access websites from within our program.*

ii) NumPy library:

This *library* provides some advance *math functionalities along with tools to create and manipulate numeric arrays.*

iii) SciPy library:

This is another useful *library* that offers *algorithmic and mathematical* tools for *scientific calculations.*

iv) tkinter library:

This *library* provides traditional *Python user interface toolkit* and help us to create user friendly *GUI interface* for *different types of applications.*

v) **Malplotlib library:** This *library* offers many *functions* and *tools* to produce *quality output* in variety of formats such as *plots, charts, graphs etc.*

2. Module:

A *act of partitioning* a program into *individual components* ***(known as modules)*** is called *modularity*. A *module* is a separate unit in itself. The justification for partitioning a program is that

- It reduces its *complexity to some degree*
- It creates a number of *well-defined, documented boundaries* within the *program.*

Or

A Python *module is a normal Python file* ***(.py file)*** containing *one or more* of the following objects related to a particular task:

- **docstrings** — *triple quoted comments;* useful for *documentation* purpose. For *documentation*, the *docstrings* should be the first string stored inside a *module/function – body/class*
- **variables and constants:** — *labels* for data
- **classes:** — *templates / blueprint* to create *objects* of a *certain kind*.
- **objects:** — *instances* of *classes.* In general, *objects* are representation of some *real or abstract* entity.
- **statements** — *Instructions.*
- **functions** — *name group* of instructions.

Hence, we can say that module 'ABC' means it is a file 'ABC.py'.

Note: *Python comes loaded with some predefined modules that we can use and we can even create our own modules.*

A module, in general:

- is independent grouping of *code* and *data (variables, definitions, statements and functions)*
- Can be *re-used* in other *programs.*
- Can *depend* on *other module*

3. Creating user defined Module:

The most common way to create a module is to define a *file* with the *.py* extension that will contain the code required to group separately from the rest of the application.

Example 1:

In this example we will create a *module* with *two simple functions* to display the *area and perimeter of rectangle*. We will then *import* and use these *functions* in *another file.*

Example module, namely Rectangle.py

```
#Rectangle.py

def Area_rec(s1,s2):
    area = s1*s2
    return area

def Perimeter_rec(s1,s2):
    peri=2*(s1+s2)
    return peri
```

*Run the above code and type following on the **Python's shell prompt** >>> after importing the module with **import<module-name>** command:*

>>> import <Rectangle)

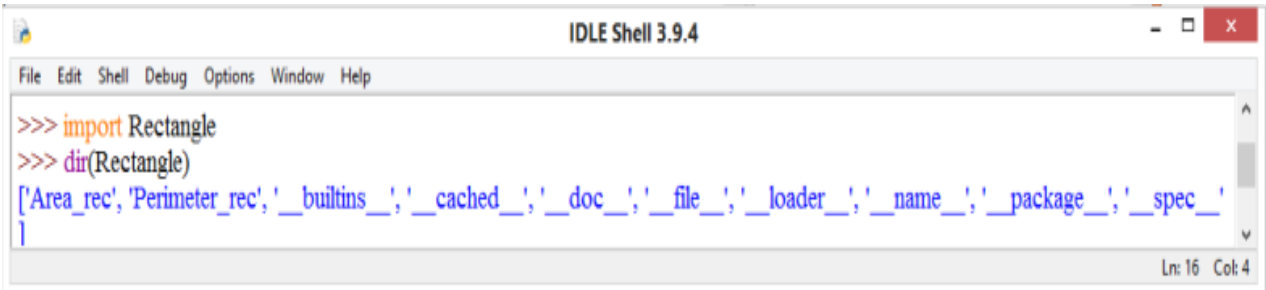
After *importing* the module, module *Rectangle*, if we write;

>>>help(tempconversion)

Python will display all *docstrings* along with *module name, filename, functions' name and constants as shown below:*

```
>>> import Rectangle
>>> help(Rectangle)
Help on module Rectangle:

NAME
    Rectangle

FUNCTIONS
    Area_rec(s1, s2)

    Perimeter_rec(s1, s2)

FILE
    c:\users\naresh choudhary\appdata\local\programs\python\python39\assignments(2021-22)pra
ctical\class xiia2 2021-22\chapter 4 using python libraries\rectangle.py
```

There is one more ***function dir()*** when applied to a ***module***, gives us *names of all* that is *defined* inside the ***module***



Example 2:
Example module, namely tempconversion.py

```
#tempconversion.py

"""Conversion functions between farenheit and centigrade"""

def to_centigrade(x):
    """Returns: x converted to centigrade """
    return 5 * (x-32)/9.0

def to_fahrenheit(x):
    """Returns: x converted to fahrenheit"""
    return 9 * x /5.0 +32

#Constants
FREEZING_C = 0.0    #Water freezing temp. (in celcius
FREEZING_F= 32.0    #Water freezing temp. (in fahrenheit)
```

*Run the above code and type following on the **Python's shell prompt** >>> :*