Python Module

A Python module, also known as a sub program or function, is a file consisting of docstrings, variables, constants, classes, objects and statements. When we directly execute a Python program, it will execute the main module of the program. The main modules give the origin for a complete program in Python. They may be allowed to import one or more modules, while each of these modules can import other modules. However, the main modules can't be imported into other modules.

Like main module, imported modules may contain a set of statements and / or definitions. These statements / definitions of imported modules are executed only once in main module. The Python standard library contains a set of predefined modules also called as built-in modules.

We have already discussed about some of these modules and their uses, such as the math, random and random in class XI. For instance,

- Module math is used for the built-in functions such as pi(), e(), sqrt(), ceil(), floor(), pow(), fabs(), sin(), cos(), tan(), etc…
- Module random is used for the functions like random(), randint() and randrange()
- Module statistics is used for the functions such as mean(), median() and mode()

By convention, name of the modules are written with lowercase letters and optional underscores. The modules are stored with the extension .py, like a Python program.

The advantages of using modular programming in Python are

- **Software design**

  It facilitates development of well-designed program.

- **Software development**:

  It permits dividing complex problems into small pieces, known as sub programs or modules, thereby facilitating modular programming.

  It permits reuse of the code, thereby preventing replication of the code,

  It improves clarity of the code.

  It allows information hiding by implementing object oriented programming.

- **Software Testing**

  It facilitates various means of testing parts of program separately.

  It permits integrating various parts of program while carrying out testing.

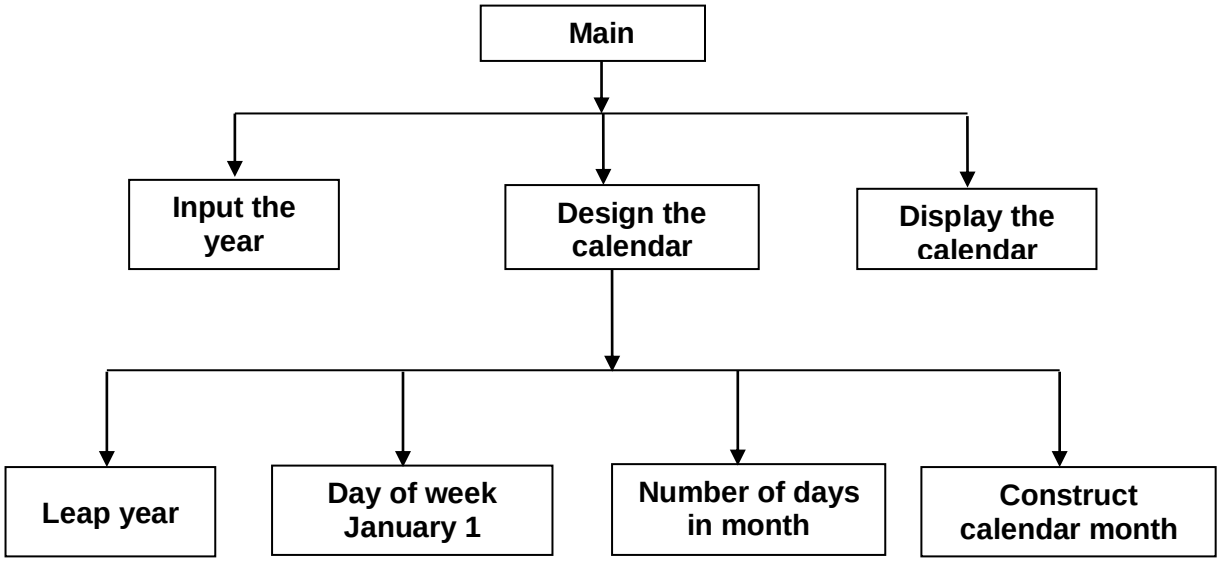
- **Software maintenance and modification**

  It permits the modification of one or more modules while maintaining the software. The software design and development take place once, whereas the software maintenance and hence testing are carried out on many occasions depending on up-gradation in technology

Top-Down Design

Top-down design is an approach involved in deriving a modular design, where the development of overall design of a system is first carried out and more detailed aspects of the specification of the design are deferred to later steps. The following diagram illustrates the modular design using a top-down approach for writing a program to design a calendar year.

The modules, **Input the year** and **Display the year** use simple logic and therefore, they need not further be broken down, whereas the module **Design the calendar** involves various actions and hence it needs further breakdown into different modules as shown in the diagram. The main goal of top-down design is that each module provides clearly defined actions/ events, which collectively provide all of the required process of the program.



Python Library

A library is a collection of modules/ functions / methods which permit us to carryout various applications or requirements. Some of the most commonly used libraries in Python are

- **Standard library** consisting of modules like math, random, statistics etc.
- **Pickle library,** which is mainly used for converting an object in Python into byte of stream and re-creating the original object through the functions load() and dump(). The function load() is used to read objects from the files/databases, whereas the function dump() is used to store objects in the files / databases.

- **NumPy library**, which consists of many modules for creating and manipulating numeric arrays**.**
- SciPy library, which helps in scientific calculations as it consists of tools for algorithmic and mathematical functions.
- **Tkinter library**, which provides toolkit for creating GUI interface for various applications in event driven programming.
- **Malplotlib library,** which is useful for producing output in various forms such as plots, charts, graphs etc…

Modules and Namespaces

A namespace is a container, which facilitates a named framework for a set of identifiers. It ensures the Python programs in avoiding potential name clashes by associating each identifier with the namespace from which it is driven. In software development, a name clash takes place when two distinct entities with the same name become part of the same scope.

Importing Modules

In Python, the **main module** is the first module executed of any program. The Python interpreter acts as the main module, containing the *global namespace* while working interactively in the Python shell. As we discussed in class XI, the built-in functions are automatically imported in Python programs, providing all the built-in constants, functions, and classes. However, for using standard library functions, modules need to be imported. Python supports various forms of importing modules as depicted below.

Importing entire module(s)

The import statement can be used for importing entire module(s). Its syntax is

import <module1> [, <module2>] [, <module3>]

Here arguments written within the angular brackets < > are mandatory, whereas the arguments written within the square brackets are optional

Example

>>> import math

>>> print (math.pow(2,4))

Then, output will be 16.0

Instead of using module name, we can use duplicate name or alias name while importing the module as given below. Its syntax is

import <module> as <aliasname>

Example

>>> import math as m

>>> print (m.sqrt(36))

Then, output will be 6.0

Importing selected objects/ events from a module

To accomplish this task, the import statement can be written with the syntax

from <module> import <objname1> [,<objname2>] […]

Example

>>>from math import e

>>> print(e)

The output will be 2.718…

Notice that module name math is not used while printing value of 'e'

>>> from random import randint, randrange

This form permits us to import more objects at a time.

>>> from math import *

It helps us to import all the objects from the module math.

>>>print (floor(2.6), ceil(5.1))

The output will be 2 6

Steps involved while using various forms of import command

- **Import <module>**
  - i) The code(s) of the imported module(s) will be first interpreted and executed.
  - ii) The functions and variables of imported module(s) are available for the program.
  - iii) A new namespace is created as that of the module imported.

  For instance, when the module Factorial consisting of function fact() is imported, then Factorial.fact() is used in the program, where Factorial is the namespace.

- **from < module > import < object >**
  - i) The code(s) of the imported module(s) will be first interpreted and executed.
  - ii) The functions and variables of only selected objects of the imported module(s) are available for the program.

iii) No new namespace is created as the objects are imported from the module by using name of the objects.

For instance, when an object **pi** of the module **math** is imported, then name **pi** can directly be used without creating the namespace.

Module Loading and Execution

In a Python program, each imported module has to be located and loaded into memory. It first searches for modules in the current directory. If the module is not found, then it searches other directories specified in the Python-path environment variable. If Python-path is not defined, the searching is done from a Python installation specific path such as C:\Python32\Lib. If the module still is not found, an error (ImportError exception) is reported.

Generally, all the modules of a program will be kept in the same directory. When modules are to be shared by other programs, then they can be saved in directory specified in the Python-path, or stored in the particular Python installation Lib directory.

```
#Program to display values generated through random functions
import random as r

A = [10, 20, 30, 40, 50, 60, 70, 80]
S = r. randint(1,4)
F = r. randint(S,5)

for K in range(S, F+1):
    print(A[K], end='#')
```

The possible output will be one of the options as listed below.

20#30#40#50#60#	50#60#	30#
30#40#50	20#30#40#50#	20# and so on.

```
#Program to display values generated through random functions
import random as r

SUB = ['ENG','PHY','CHEM','CS','MATHS']
R= r. randint(1,3)
OS = ""
for J in range(R, 0,-1):
    OS += SUB[J]+'$'

print(OS)
```

The possible output will be one of the options as listed below.

CS$CHEM$PHY$	CHEMPHY

```
#Program to illustrate math module
#To determine time of flight, maximum height and range of a projectile
#projected with a velocity ‘u and angle ‘th’ with horizontal
import math as m
def Projectile(u, th):
    g=9.8
    rth=m.radians(th)
```

```
    t = 2*u*m.sin(rth)/g
    h=u*u*m.sin(rth)*m.sin(rth)/(2*g)
    r = u*u*m.sin(2*rth)/g
    return t, h, r

#_Main_
u = float (input('Enter velocity = '))
th = float (input('Enter angle = '))
tm, mh, re = Projectile(u, th)
print('Time of flight = %.1f' %tm, 's')
print('Maximum height = %.1f' %mh, 'm')
print('Range = %.1f' %re, 'm')
```

OUTPUT

```
Enter velocity = 20
Enter angle = 30
Time of flight = 2.0 s
Maximum height = 5.1 m
Range = 35.3 m
```

References

(i) Introduction to Computer Science using Python by Charles Dierbach

(ii) Dhanpat Rai and Company publications

Atomic Energy Central School, Indore

Class : XII Subject : Computer Science Unit : 1, CTP – 2

Lesson : 4.1 Using Libraries in Python Module-1 Worksheet

Name:_____ Roll No.:____ Date: _____

Maximum Marks : 22 Marks Obtained: _____

1. In Python, define the term 'Module'. Is it possible to import main module into other modules. Justify your answer. (2)

2. Is it necessary to import module(s) for using built-in library functions in a Python program? Name any three such functions. (2)

3. In Python, name the modules to be imported while using the following stand library functions. (2)
   (i) fabs() (ii) ioad() (iii) randint() (iv) mean()

4. List and briefly explain the advantages of using modular programming in Python. (3)

5. With a suitable Python code, explain the importance of Namespace while using imported modules. . (2)

6. Write possible output(s) are expected to be displayed while executing following Python code. Also write maximum values each to be assigned to the variables START and STOP. (2)

```
import random
P = [30, 50, 40, 60, 20, 10, 70]
START = random.randint(1,4)
STOP  = random.randint(2,6)
for M in range(START, STOP+1):
        print(P[M], end='%')
```

(i) 40%60%20% (ii) 10%20%60%
(iii) 60%20%10%70 (iii) 40%20%10%

7. With suitable codes in Python, illustrate any three forms of importing modules into a program. (3)

8. Find the output of the following Python code. (3)

```
>>> import math as m
>>> print (m.pow(8, 1//3))
>>> print (m.pi)
>>> print(m.floor(4.96))
```

9. By importing necessary module(s), write a python program using a function PRISM() to calculate and return the refractive index and angle of dispersion of an equilateral prism. User has to supply necessary input(s) and display the outputs through the calling function (main module) (3)