Week 10- Modules and Packages

Why modules??

- Modules breaks the large programs into small manageable files.
- It helps in code reusability.
- Grouping of similar functions and classes, helps in easy understanding and use.
- Helps in debugging and maintenance of the code.

.

What is Module in Python?

A module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

TYPES OF MODULES

1) User defined modules: Created by the programmer.
2) Pre-defined or built-in modules: Modules that comes readily with the pythonsoftware.

   Ex: math, random, emoji

Module Creation:

- Create a python file in any editor (Idle, pycharm)
  Ex: demo.py
- Place the variables and functions of your choice in the demo.py file.
- Now a module called demo is created and ready for use.

Example:

```
#demo.py
a = 10
b = 20
```

```
def add(a, b):  # function for addition

        print(" Performing addition operation ")

        print(" The sum is:", a + b)

def product(a, b): # function for addition

         print(" Performing multiplication operation ")

        print(" The product is:", a * b)
```

IMPORTING A MODULE

- To use the variables and functions of a module, we should import it into ourprograms.
- import statement is used to import the modules
- Syntax : import module_name
- Ex: import demo

Accessing module members

Syntax : modulename.membername

Ex:

```
demo.a
demo.b
demo.add( )
demo.product( )
```

THE IMPORT STATEMENT

There different import statements we can use to import module

- import statement
- from import statement
- from import * statement

- Examples:

➢ import module1, module2, module3…

- Ex: import demo, test,

  shapesdemo.add(2,3)

  test.Test()

  shapes.rectangle()

➢ from module_name import member1, member2…

- Ex: from demo import a,

  addadd(2,3)

➢ from import * statement – import all the members of the module

- Ex: from demo import *

MODULE ALIASING

➢ Giving alternate name to module at the time of importing it, is termed asmodule aliasing.

- Ex: import demo as d
  d.a

  d.b
  d.add(10,20)
  d.product(5, 10)

The dir() function:

➢ The dir( ) is used to find all the members present in the module.
- Ex: import math
  print(dir(math))
- Ex: import random
  print(help(random))

Predefine Modules:

random module:

The random module is a built-in module to generate the pseudo-random variables. It can be used perform some action randomly such as to get a random number,selecting a random elements from a list, shuffle elements randomly, etc.

1) random() : To Generate Random Floats

   The random.random() method returns a random float number between 0.0to 1.0. The function doesn't need any arguments.

   Example:

   ```
   import random
   >>> random.random()  # Output: 0.645173684807533
   ```

2) randint(): Generate Random Integers

   The random.randint() method returns a random integer between thespecified integers.(both inclusive)

   Example:

   ```
       >>> import random
       >>> random.randint(1, 100)          #95
       >>> random.randint(1, 100)          # 49
   ```

3) **randrange(): Generate Random Numbers within Range**

   The random.randrange() method returns a randomly selected element from the range created by the start, stop and step arguments. The value of start is0 by default. Similarly, the value of step is 1 by default.

   Example:

   ```
   >>> random.randrange(1, 10)                        #2
   >>> random.randrange(1, 10, 2)                     #5
   >>> random.randrange(0, 101, 10)                   #80
   ```

4) Select Random Elements

The random.choice() method returns a randomly selected element from anon-empty sequence. An empty sequence as argument raises an IndexError.

Example:

```
>>> import random
>>> random.choice('computer')
't'
>>>random.choice([12,23,45,67,65,43])
45
>>>random.choice((12,23,45,67,65,43))
67
```

5) choices(): Selecting multiple random elements:

The random.choices() method returns specified number of randomly selectedelements from a non-empty sequence.

Example:

```
>>> import random
>>>random.choices('computer',k=2) # ['m', 'o']
```

6) **shuffle() : Shuffle Elements Randomly**

The random.shuffle() method randomly reorders the elements in a list.

Example:

```
numbers=[12,23,45,67,65,43]
>>> random.shuffle(numbers)
>>> numbers
[23, 12, 43, 65, 67, 45]
```

Python math Module

Python has a built-in module that you can use for mathematical tasks

List of Functions in Python Math Module

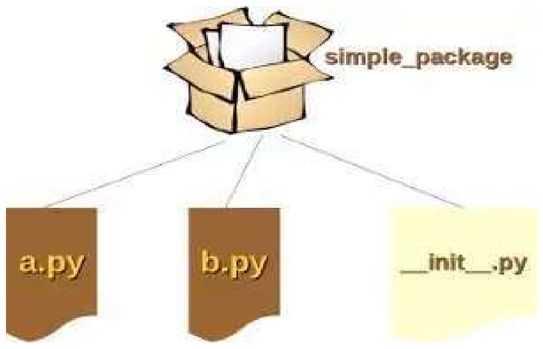
Function	Description
ceil(x)	Returns the smallest integer greater than or equal to x.

factorial(x)	Returns the factorial of x
floor(x)	Returns the largest integer less than or equal to x
trunc(x)	Returns the truncated integer value of x
exp(x)	Returns e**x
pow(x, y)	Returns x raised to the power y
sqrt(x)	Returns the square root of x
acos(x)	Returns the arc cosine of x
asin(x)	Returns the arc sine of x
atan(x)	Returns the arc tangent of x
cos(x)	Returns the cosine of x
sin(x)	Returns the sine of x
tan(x)	Returns the tangent of x
radians(x)	Converts angle x from degrees to radians
pi	Mathematical constant, the ratio of circumference of a circle to it's diameter (3.14159...)
e	mathematical constant e (2.71828...)

Packages

A package is basically a directory with Python files and a file with the name __init___.py. This means that every directory inside of the Python path, which contains a file named_init_.py, will be treated as a package by Python. We can put several modules into a Package.

A Simple Example



- Lets demonstrate with a very simple example how to create a package with somePython modules.
- First of all, we need a directory. The name of this directory will be the name of the package, which we want to create and name as "simple_package".
- This directory needs to contain a file with the name init__.py. This file can be empty, or it can contain valid Python code. This code will be executed when a package is imported, so it can be used to initialize a package,
- Now we can put all of the Python files which will be the submodules of ourmodule into this directory.
- We create two simple files a.py and b.py in simple_package

The content of a.py:

```
def bar():
    print("Hello, function 'bar' from module 'a' calling")
```

The content of b.py:

```
def foo():
    print("Hello, function 'foo' from module 'b' calling")
```

Now you can use the package in your program by importing it.

1) from simple_package import

```
a, ba.bar()

b.foo()
```

OUTPUT:

```
Hello, function 'bar' from module 'a' calling

Hello, function 'foo' from module 'b' calling
```

2). Another way of importing the package

```
import simple_package

simple_package.a.bar()

simple_package.b.foo()
```

OUTPUT:

```
Hello, function 'bar' from module 'a' calling

Hello, function 'foo' from module 'b' calling
```