Unit 4: Python Functions, Modules & Packages

Modules

- A module are primarily .py files that represents group of classes, methods and functions.
- We have to group them depending on their relationship into various modules and later use these modules in other programs.
- It means, when a module is developed it can be reused in any program that needs that module.

Advantages of modules –

• Reusability: Working with modules makes the code reusability a reality.

• Simplicity: Module focuses on a small proportion of the problem, rather than focusing on the entire problem.

• Scoping: A separate namespace is defined by a module that helps to avoid collisions between identifiers.

Writing Module:

- Writing a module means simply creating a file which contains python definition and statements. The file name is the module name with the extension .py. To include module in a file, use the import statement.

Follow the following steps to create modules:

1. Create a first file as a python program with extension as .py. This is your module file where we can write a function which performs some task.

2. Create a second file in the same directory called main file where we can import the module to the top of the file and call the function.

Example:

Create p1.py file and add the below code in the file

```
def add(a,b):
        result=a+b
        return result
def sub(a,b):
        result=a-b
        return result
def mul(a,b):
        result=a*b
        return result
def div(a,b):
        result=a/b
        return result
```

Create p2.py file I same directory where p1.py is created and add the below code in the file

```
import p1
print("Addition= ",p1.add(10,20))
print("Subtraction= ",p1.sub(10,20))
print("Multiplication= ",p1.add(10,20))
print("Division= ",p1.add(10,20))
```

Output:

```
Addition= 30
Subtraction= -10
Multiplication= 30
Division= 30
```

Importing Modules

- Import statement is used to import a specific module by using its name. Import statement creates a reference to that module in the current namespace. After using import statement we can refer the things defined in that module.

Example: using from x import *

```
from p1 import *
print("Addition= ",add(10,20))
print("Multiplication= ",mul(10,20))
```

Output:

```
Addition= 30
Multiplication= 200
```

Example:

```
from p1 import add,sub
print("Addition= ",add(10,20))
print("Subtraction= ",sub(10,20))
```

Output:

```
Addition= 30
Subtraction= -10
```

Import with renaming (Aliasing Modules):

- It is possible to modify the names of modules and their functions within python by using the ‘as’ keyword.
- We can make alias because we have already used the same name for something else in the program or we may want to shorten a longer name.

Syntax:

import module as another_name

e.g. if a module created with name p1 which has function named ‘add’

```
import p1 as m
print(‘addition is :’,m.add)
```

Python Built-in Modules

- A module is a collection of python objects such as functions, classes and so on.
- Python interpreter is bundled with a standard library consisting of large number of built-in modules.
- Built-in modules are generally written in C and bundled with python interpreter in precompiled form.
- A built-in module may be a python script(.py) containing useful utilities.

Numeric and Mathematical Modules

1. math module

The math module is a standard module in Python and is always available. To use mathematical functions under this module, you have to import the module using import math.

Example:

```
import math
print (math.sqrt(25))

print math.pi
print (math.degrees(2))
```

2. cmath module

The cmath module allows us to work with mathematical functions for complex numbers.

Example:

```
from cmath import *
c=2+2j
print(exp(c))
print(log(c,2))
print(sqrt(c))
```

Random module:

• Sometimes we want the computer to pick a random number in a given range or a list.

• To perform this random module is used. This function is not accessible directly, so we need to import random module and then we need to call this function using random static object.

Example:

```
import random
print(random.random())
print(random.randint(10,20))
```

Output: 0.8709966760966288

16

Statistics module:

• It provides access to different statistics functions such as mean, median, mode, standard deviation.

Example:

```
import statistics

print(statistics.mean([2,5,6,9]))

print(statistics.median([1,2,3,8,9]))

print(statistics.mode([2,5,3,2,8,3,9,4,2,5,6]))

print(statistics.stdev([1,1.5,2,2.5,3,3.5,4,4.5,5]))
```

Namespace and Scoping

- A namespace is a system to have a unique name for each and every object in python.
- An object might be a variable or a method.
- Python itself maintains a namespace in the form of a python dictionary.
- A namespace is a system to have a unique name for each and every object in python.
- An object might be a variable or a method.
- Python itself maintains a namespace in the form of a python dictionary.

Types of Namespace:

• Local Namespace: This namespace covers the local names inside a function. Python creates this namespace for every function called in a program.

• Global Namespace: This namespace covers the names from various imported modules used in a project. Python creates this namespace for every module included in the program.

Python Variable Scoping:

• Scope is the portion of the program from where a namespace can be accessed directly without any prefix.

• At any given moment there are atleast following three nested scope.

1. Scope of the current function which has local names.
2. Scope of the module which has global names.
3. Outermost scope which has built-in names.

Local scope variable- All variables which are assigned inside a function.

Global scope variable-All those variables which are outside the function termed

Python Packages

- A package is a hierarchical file directory structure that defines a single python application environment that comprises of modules and subpackages and so on.
- Packages allow for a hierarchical structuring of the module namespace using dot notation.
- Packages are a way of structuring many packages and modules which help in a well organized hierarchy of data set, making the directories and modules easy to access.
- A package is a collection of python modules i.e. a package is a directory of python modules containing an additional _ _init_ _.py file. E.g Phone/_ _init_ _.py

Writing Python Packages

To create a package in Python, we need to follow these three simple steps:

1. First, we create a directory and give it a package name, preferably related to its operation.
2. Then we put the classes and the required functions in it.
3. Finally we create an __init__.py file inside the directory, to let Python know that the directory is a package.

Example to write Python Packages-

- Create a directory named mypkg and create a __init__.py file and save in the mypkg directory so that mypkg will be considered as package.
- Create file p1.py in the mypkg directory and add this code

```
def m1():
    print("First Module")
```

- Create file p2.py in the mypkg directory and add this code

```
def m2():
    print("second module")
```

- Create file pkgsample.py and add this code

```
from mypkg import p1,p2
p1.m1()
p2.m2()
```

Output

```
First Module
second module
```

Using Standard Packages

Math:

- Some of the most popular mathematical functions are defined in the math module.
- These includes trigonometric functions, representation functions, logarithmic functions and angle conversion functions.
- Two mathematical constants are also defined in math module.

1. Pie(∏) is a well known mathematical constant which is defined as the ratio of the circumference to the diameter of a circle and its value is 3.141592653589793

```
import math
        print(math.pi)
```

o/p 3.141592653589793

Euler’s Constant

- Another well known mathematical constant defined in the math module is e.
- It is called Euler’s number and it is a base of the natural Logarithm.
- It value is 2.718281828459045
- e.g import math

```
print(math.e)
```

o/p 2.718281828459045

NumPy:

- NumPy is the fundamental package for scientific computing with python.
- NumPy stands for “Numeric Python”.
- It provides a high performance multidimensional array object and tools for working with these arrays.
- An array is a table of elements (usually numbers) all of the same type, indexed by a tuple of positive integers and represented by a single variable.
- NumPy’s array class is called ndarray.
- In NumPy arrays the individual data items are called elements. Dimensions are called axes.
- The size of NumPy arrays are fixed , once created it cannot be changed again.
- NumPy arrays are great alternatives to python lists.

Basic attributes of the ndarray class are as follow:

- shape- A tuple that specifies the number of elements for each dimension of array.
- size- The total number of elements in the array.
- ndim- Determines the dimension of an array.
- nbytes- Number of bytes used to store the data.
- dtype- Determines the datatype of elements stored in array.

```
import numpy as np
a=np.array([10,20,30])
 print(a) arr=np.array([[1,2,3],[4,5,6]])
print(arr)
print("No. of dimension: ",arr.ndim) #2
print("Shape of array: ",arr.shape) #(2,3)
print("Size of array: ",arr.size) # 6
print("Type of elements in array: ",arr.dtype) # int32
```

Basic Array Operations:

1. Unary operators: Many unary operations are provided as a method of ndarray class. This includes sum, min, max etc. These functions can also be applied row-wise or column-wise by setting an axis parameter.
2. Binary Operators: These operations apply on array elementwise and a new array is created. You can use all basic arithmetic operators like +,-,/, * etc. In case of +=, -=, *= operators existing array is modified.

```
import numpy as np
arr1=np.array([1,2,3,4,5])
arr2=np.array([2,3,4,5,6])
print(arr1)
print("Add 1 in each element ",arr1+1) # [2 3 4 5 6]
print("Subtract 1 from each element: ",arr-1) #[0 1 2 3 4]
print("Multiply 10 with each element in array: ",arr1*10) #[10 20 30 40 50]
print("Sum of all array elements: ",arr1.sum()) # 15
print("Array sum= ",arr1+arr2) # [3 5 7 9 11]
print("Largest element in array: ",arr1.max()) # 5
```

Reshaping of Array:

- We can also perform reshape operation using python numpy operation.
- Reshape is when you change the number of rows and columns which gives a new view to an object.

Example:

```
import numpy as np
    arr=np.array([[1,2,3],[4,5,6]])
    a=arr.reshape(3,2)
    print(a)
```

Output: ([[1,2], [3,4], [5,6]])

Slicing of Array:

- Slicing is basically extracting particular set of elements from an array.

Example:

```
import numpy as np
        a=np.array([(1,2,3,4),(5,6,7,8)])
                print(a[0,2])
```

Output:

#3

SciPy:

- SciPy is a library that uses NumPy for more mathematical functions.
- SciPy uses NumPy arrays as the basic data structure and come with modules for various commonly used tasks in scientific programming, including linear algebra, integration (calculus), ordinary differential equation solving and signal processing.
- SciPy is organized into subpackage covering different scientific computing domains

Sr. No.	Subpackage	Description
1.	cluster	Clustering algorithms.
2.	constants	Physical and mathematical constants.
3.	fftpack	Fast Fourier Transform routines.
4.	integrate	Integration and ordinary differential equation solvers.
5.	interpolate	Interpolation and smoothing splines.
6.	io	SciPy has many modules, classes, and functions available to read data from and write data to a variety of file formats.
7.	linalg	Linear algebra.
8.	ndimage	N-dimensional image processing.
9.	odr	Orthogonal distance regression.
10.	optimize	Optimization and root-finding routines.

- e.g. :using linalg sub package of SciPy

```
import numpy as np
from scipy import linalg
    a=np.array([[1.,2.],[3.,4.]])
    linalg.inv(a) # find inverse of array
```

o/p array([[-2.,1.],[1.5,-0.5]])

- e.g. Using special sub package of SciPy

```
from scipy.special import cbrt
        cb=cbrt([81,64])
        print(cb) # find cube root
```

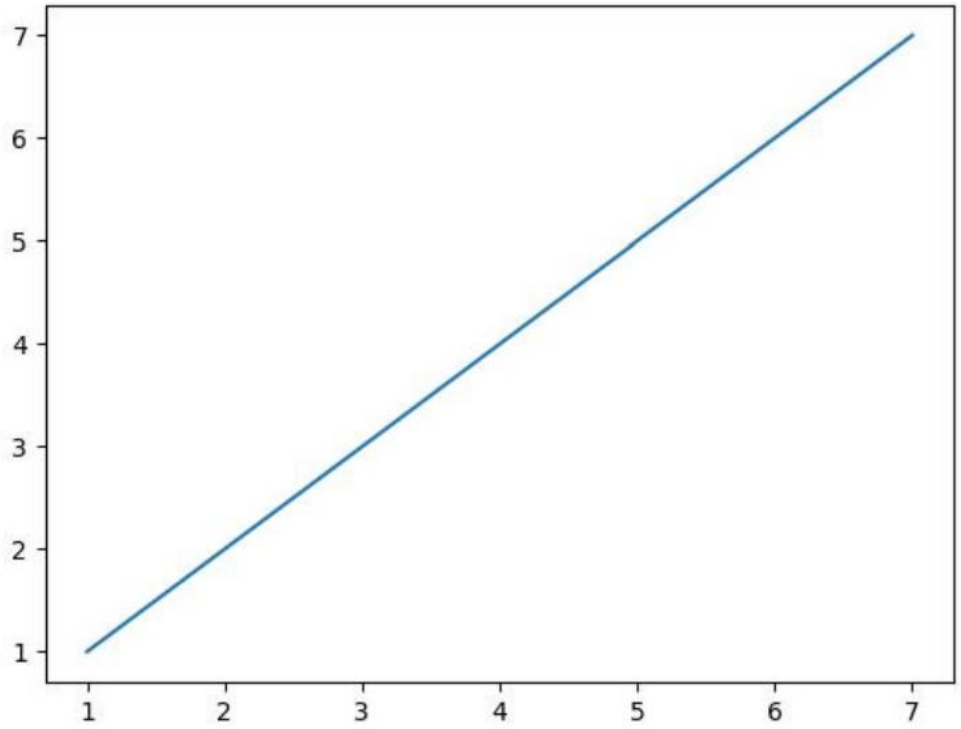
o/p array([4.32674871, 4.])

Matplotlib:

- It is a plotting library used for 2D graphics in python programming language.
- It can be used in python scripts, shell, web application servers and other graphical user interface.
- There are various plots which can be created using python matplotlib like bar graph, histogram, scatter plot, area plot, pie plot.

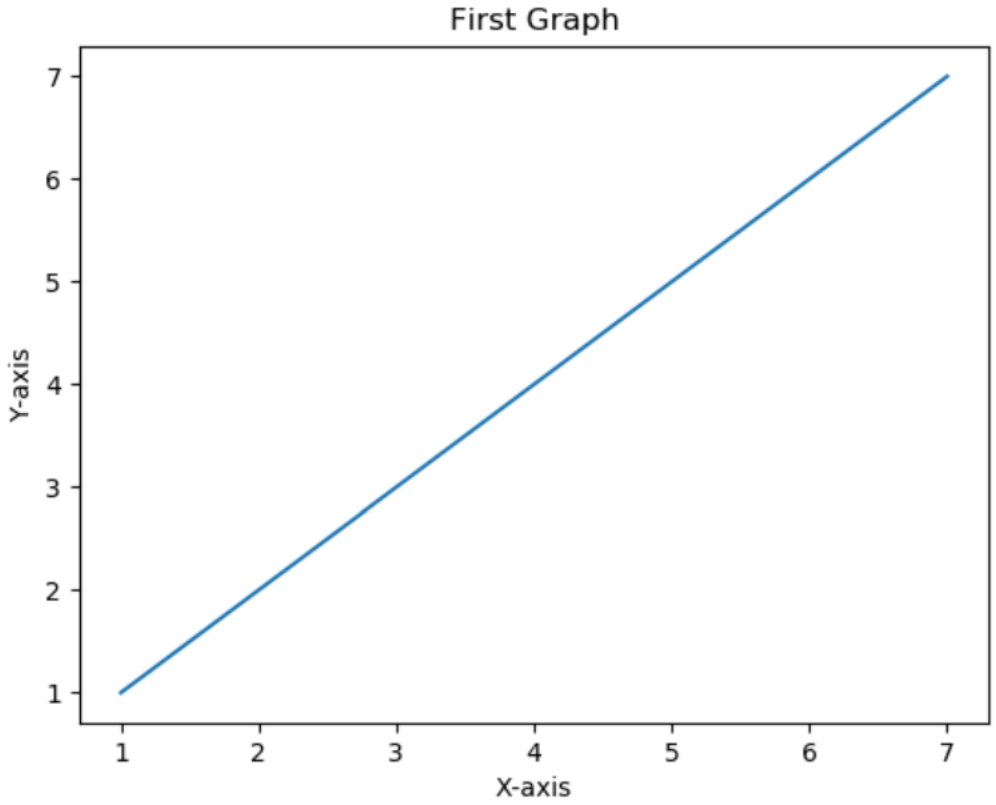
Example:

```
import matplotlib.pyplot as plt
x=[1,2,3,4,5,6,7]
y=[1,2,3,4,5,6,7]
plt.plot(x,y)
plt.show()
```



Example:

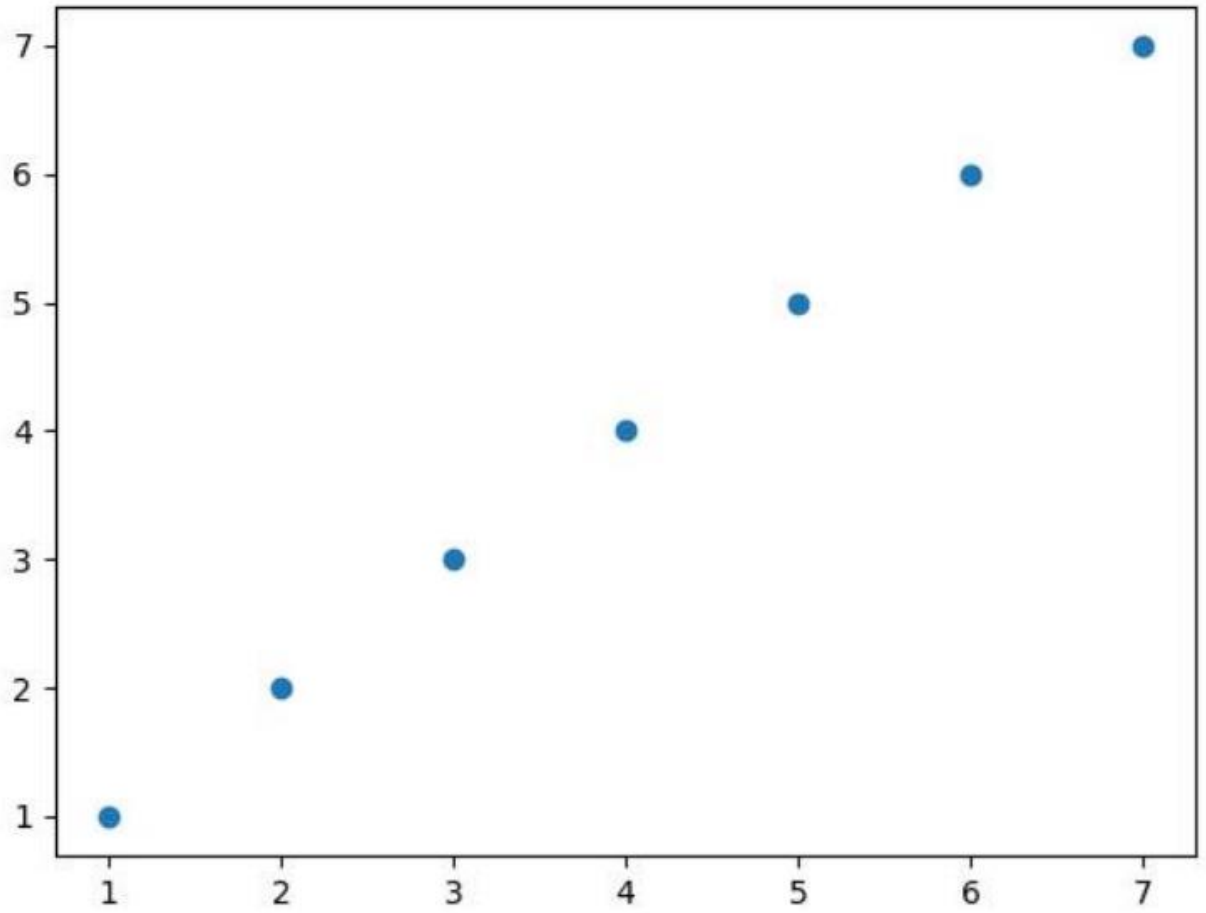
```
import matplotlib.pyplot as plt
x=[1,2,3,4,5,6,7]
y=[1,2,3,4,5,6,7]
plt.plot(x,y)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('First Graph')
plt.show()
```



Scatter Plot

e.g.

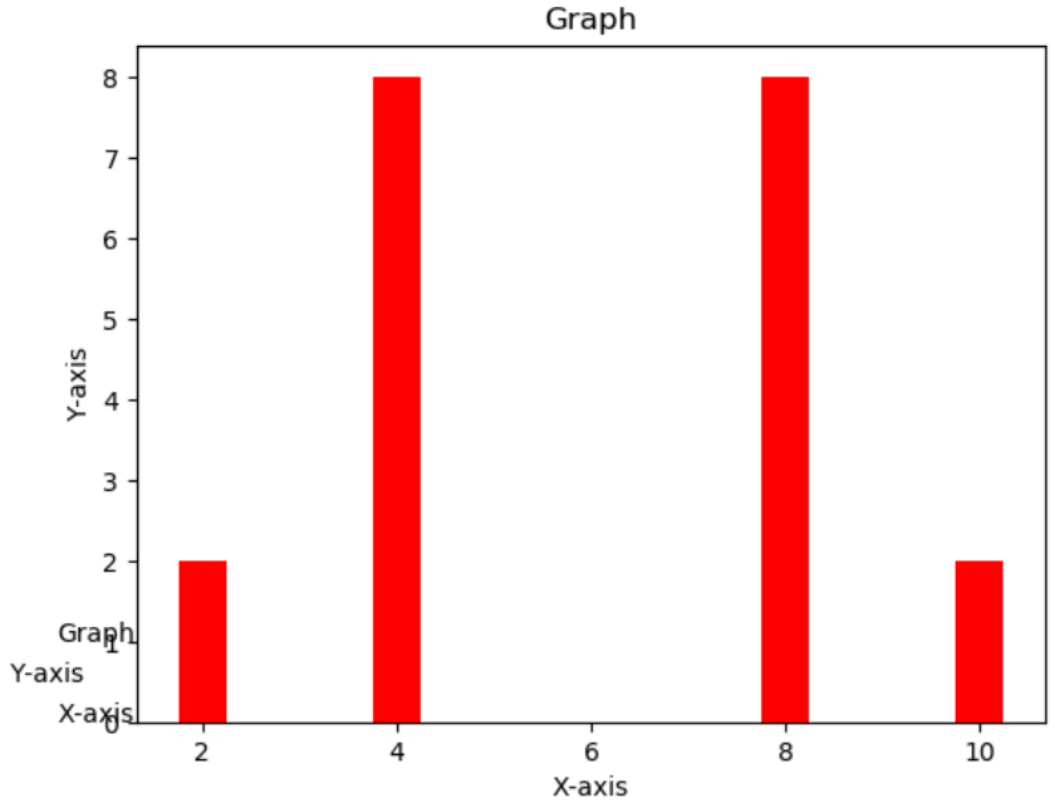
```
import matplotlib.pyplot as plt
x=[1,2,3,4,5,6,7]
y=[1,2,3,4,5,6,7]
plt.scatter(x,y)
plt.show()
```



Bar Graph

- A bar graph uses bars to compare data among different categories.
- It is well suited when you want to measure the changes over a period of time.
- It can be represented horizontally or vertically.

```
from matplotlib import pyplot as plt
x=[2,4,8,10]
 y=[2,8,8,2]
plt.xlabel('X-axis')
plt.text(0.5,0,'X-axis')
plt.ylabel('Y-axis')
plt.text(0,0.5,'Y-axis')
plt.title('Graph')
plt.text(0.5,1.0,'Graph')
plt.bar(x,y,label="Graph",color='r',width=0.5)
plt.show()
```



Pandas:

- It is an open source python library providing high performance data manipulation and analysis tool using its powerful data structures.
- Pandas is python library which is used for data analysis purpose.
- In pandas data can be represented in 3 ways:

1. Series
2. Data frames
3. Panel

Series:

It is one dimensional array like structure defined in pandas.

It can be used to store the data of any type.

e.g.

```
import pandas as pd data=[1.1,2.1,3.1,4.1,5.1,6.1,7.1,8.1]
df=pd.Series(data)
print('The series is:\n',df)
print('The type is:',type(df))
```

Output

The series is: 0 1.1 1 2.2 2 3.1 3 4.1 4 5.1 5 6.1 6 7.1 7 8.1

dtype: float64

• DataFrame-

• Dataframes are two dimensional structures which consist of columns and rows.

• It can be used to store the data of any type.

e.g.

```
import pandas as pd
dict1={'a':1.1,'b':2.1,'c':3.1,'d':4.1}
dict2={'a':5.1,'b':6.1,'c':7.1,'d':8.1}
data={'Col 1':dict1,'Col 2':dict2}
df=pd.DataFrame(data)
print(df)
```

Output

	Col1	Col2
a	1.1	5.1
b	2.1	6.1
c	3.1	7.1
d	4.1	8.1

Panel-

• Panel is a three dimensional data structure with homogeneous data. • It is hard to represent the panel in graphical representation.

• But a panel can be illustrated as a container of DataFrame.

It takes following arguments

1. data: The data can be of any form like ndarray, list, dict, map, DataFrame
2. item: axis 0, each item corresponds to a dataframe contained inside.
3. major_axis: axis 1, it is the index (rows) of each of DataFrames.
4. minor_axis: axis 2, it is the column of DataFrames.
5. dtype: The data type of each column.
6. copy: It takes a Boolean value to specify whether or not to copy the data. The default value is false.

```
import pandas as pd
import numpy as np
data = np.random.rand(2,4,5)
 p = pd.Panel(data)
print p
```

Output:

```
<class 'pandas.core.panel.Panel'>

Dimensions: 2 (items) x 4 (major_axis) x 5 (minor_axis)

Items axis: 0 to 1

Major_axis axis: 0 to 3

Minor_axis axis: 0 to 4
```

User defined Packages:

- We organize a large number of files in different folders and subfolders based on some criteria, so that we can find and manage them easily.
- In the same way a package in python takes the concept of the modular approach to next logical level.
- A package can contain one or more relevant modules.
- Physically a package is actually a folder containing one or more module files.

Example to create and access the packages-

• Create a directory named mypkg1 and create a __init__.py file and save in the mypkg1 directory so that mypkg1 will be considered as package.

message.py

```
def sayhello(name):
        print("Hello " + name)
        return
```

mathematics.py

```
    def sum(x,y):
    return x+y
def average(x,y):
        return (x+y)/2
def power(x,y):
return x**y
```

Create p1.py with the code given p1.py

```
from mypkg1 import mathematics
 from mypkg1 import message
        message.sayhello("Amit")
        x=mathematics.power(3,2)
        print("power(3,2) :",x)
```

Output

```
Hello Amit
power(3,2) : 9
```