11. NumPy Pandas Matplotlib SciPy

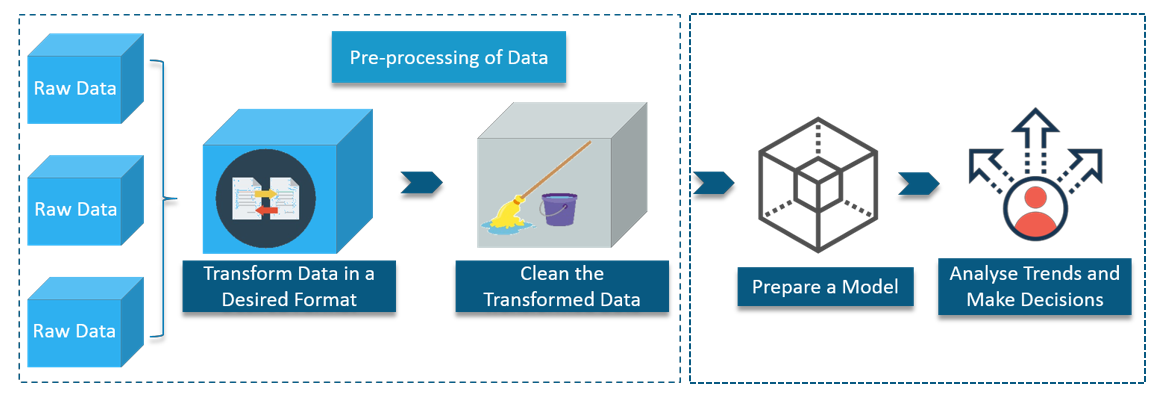
Table of Contents

|  |  |  |
| --- | --- | --- |
| Sl. No | Topics | Page No |
| 11.1 | **NumPy** |  |
| 11.2 | **Pandas** |  |
| 11.3 | **Matplotlib** |  |
| 11.4 | **SciPy** |  |

## 11.1 NumPy

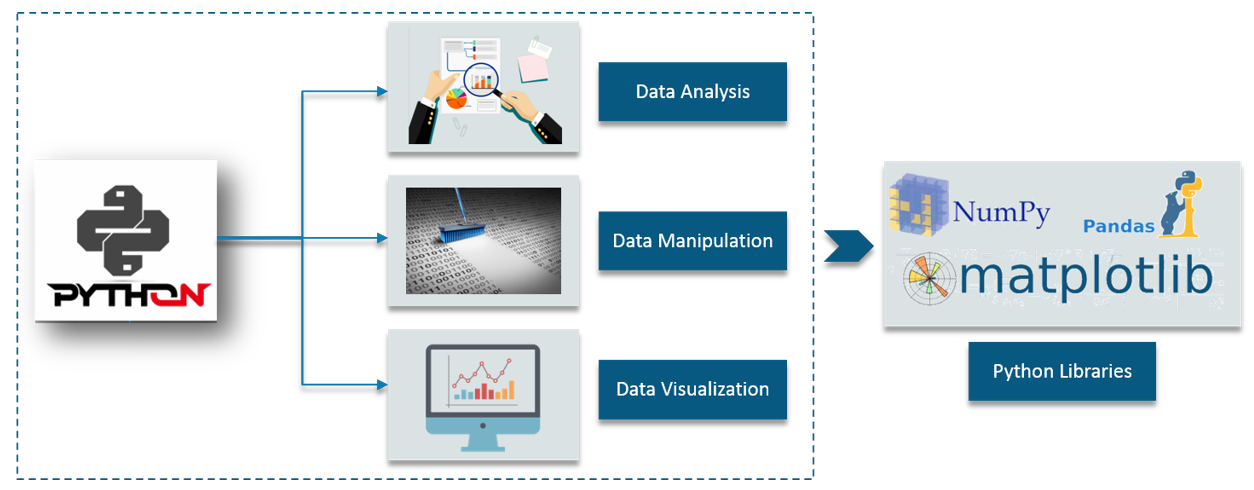
**What is Data Analysis?**

Data Analysis is a process of inspecting, cleansing, transforming, and modelling **data** with the goal of discovering useful information, suggesting conclusions, and supporting decision-making.



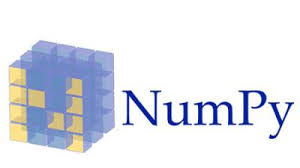
**Why Python used for Data Analysis?**

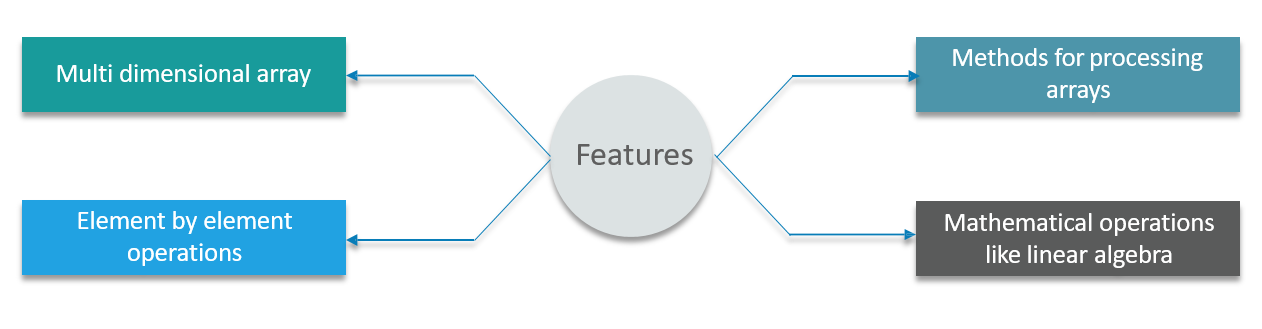
Python provides various methods for data analysis, manipulation and visualisation



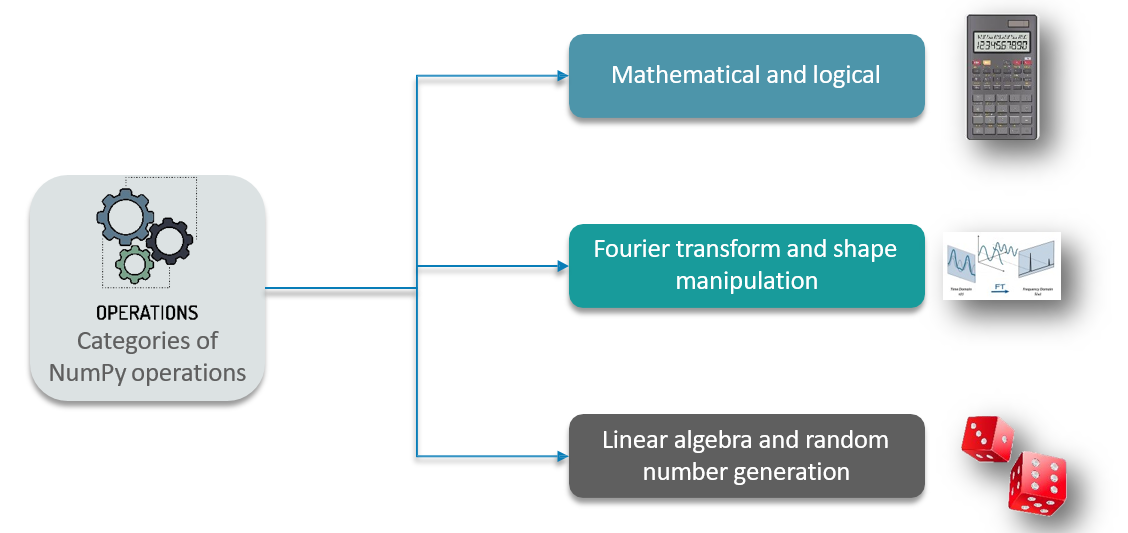
**What is NumPy?**

NumPy is a package for scientific computing.





**What are the Operations in NumPy?**



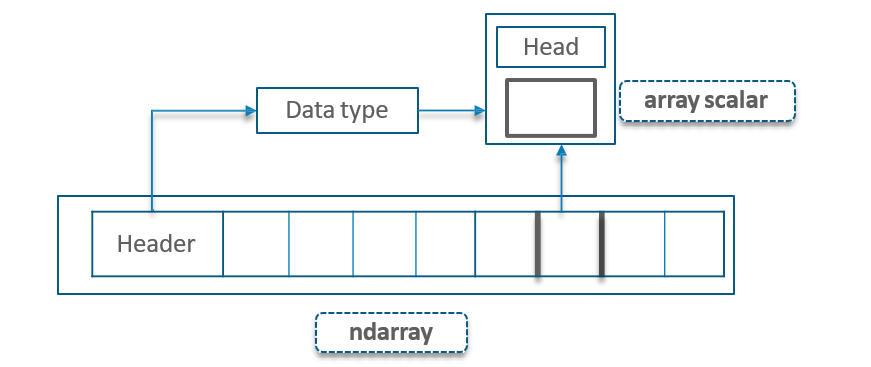
**NumPy Array**

**Ndarray – NumPy Array**

The ndarray is a multi-dimensional array object consisting of two parts -- the actual data, some metadata

which describes the stored data. They are indexed just like sequences are in Python, starting from 0.

* Each element in ndarray is an object of data-type object (called **dtype**)
* An item extracted from **ndarray**, is represented by a Python object of an array scalar type

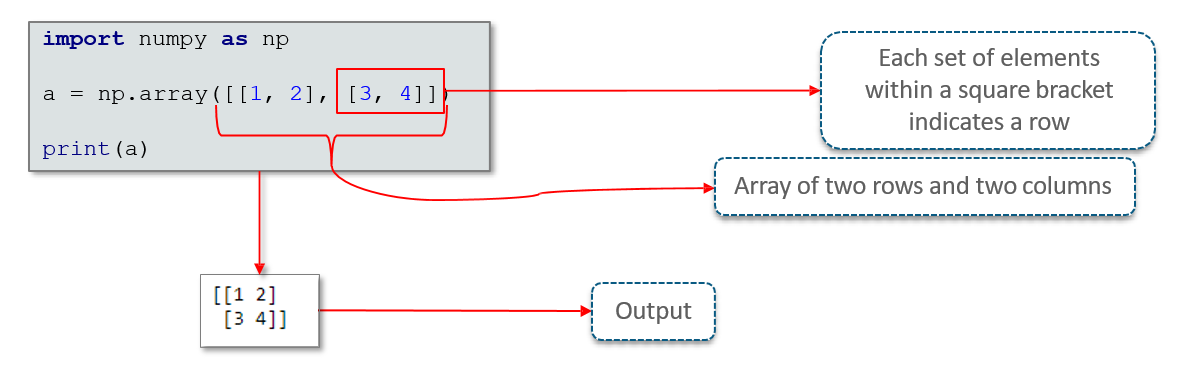


**Creating a NumPy Array**

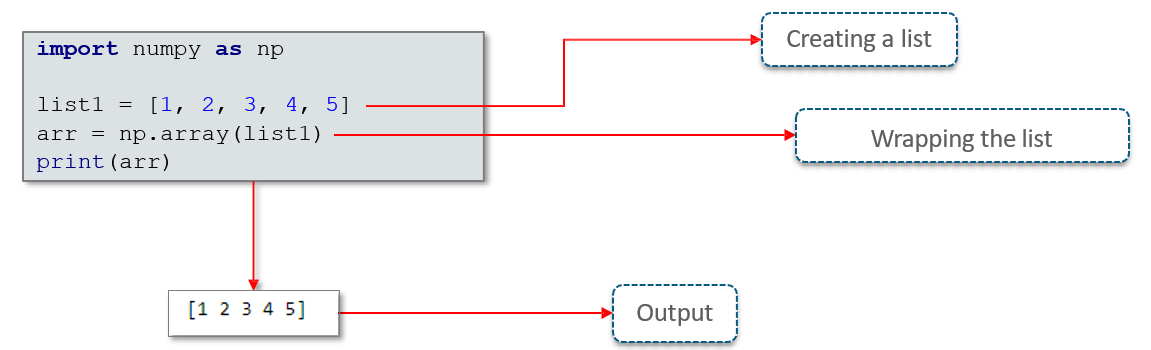
**Creating a NumPy Array – Single-Dimensional Array**



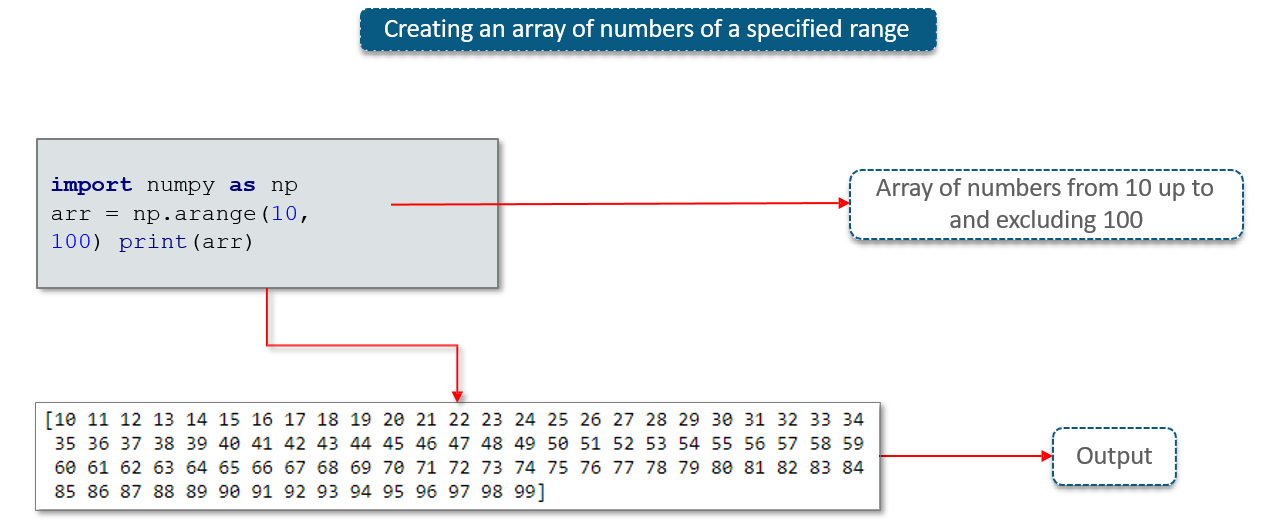
**Creating a NumPy Array – Multi-Dimensional Array**

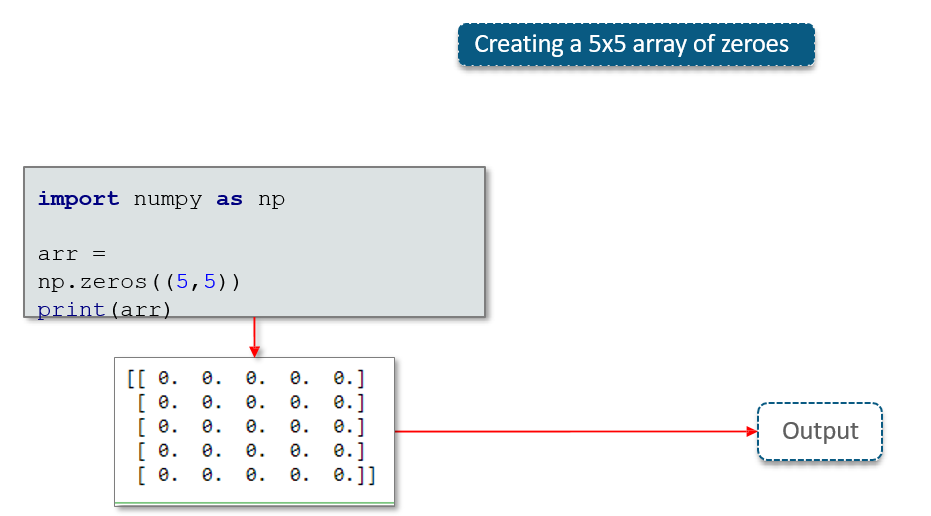


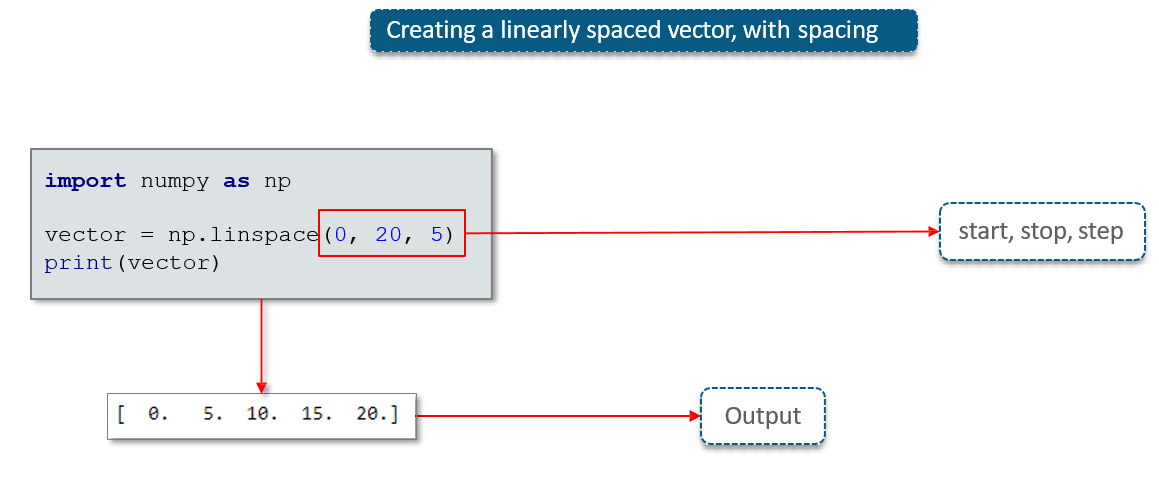
**Creating a NumPy Array - List Wrapping**

****

**Example: Creating a NumPy Array**

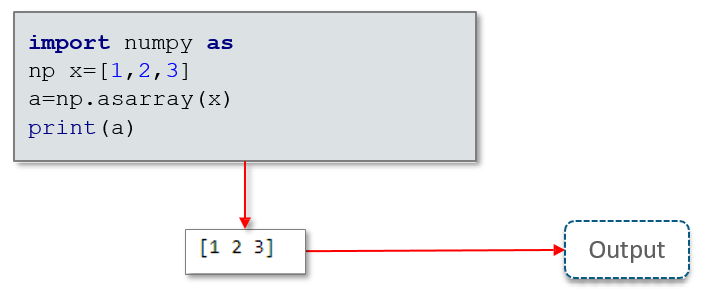




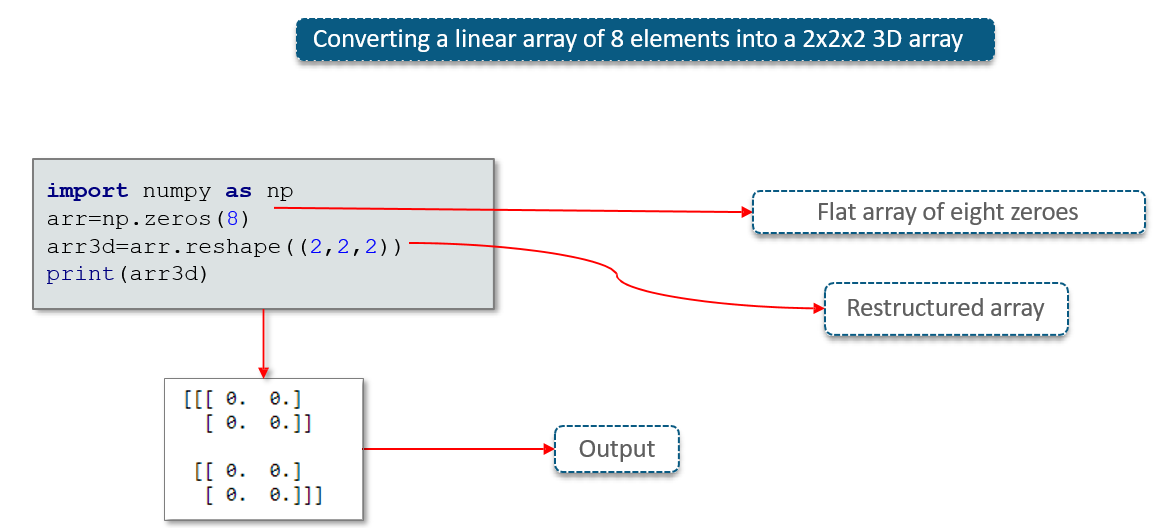


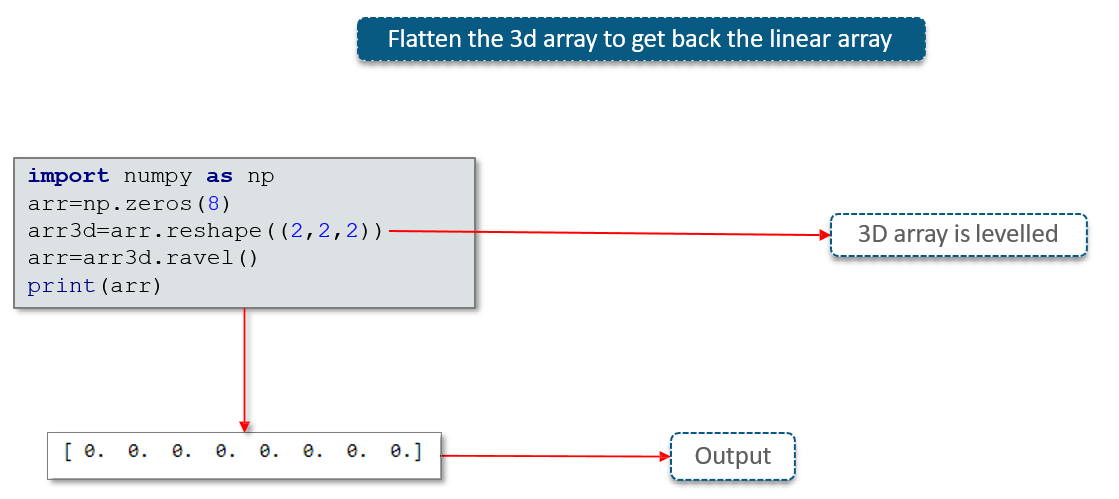
**Creating Arrays from Existing Data**

* **numpy.asarray** – used for converting Python sequences into ndarrays
* Syntax *- numpy.asarray(a, dtype = None, order = None)*
* The following piece of code converts a python list into an array



**Restructuring a NumPy Array**

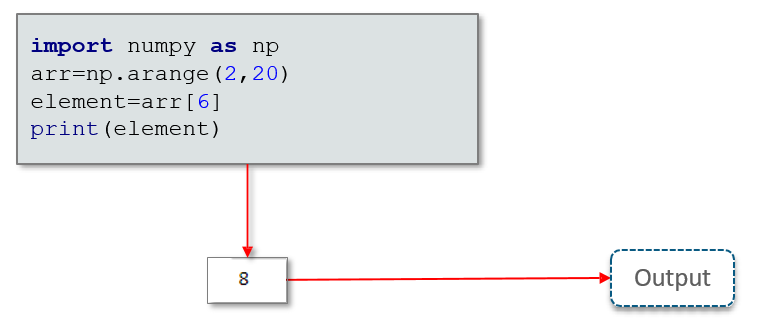




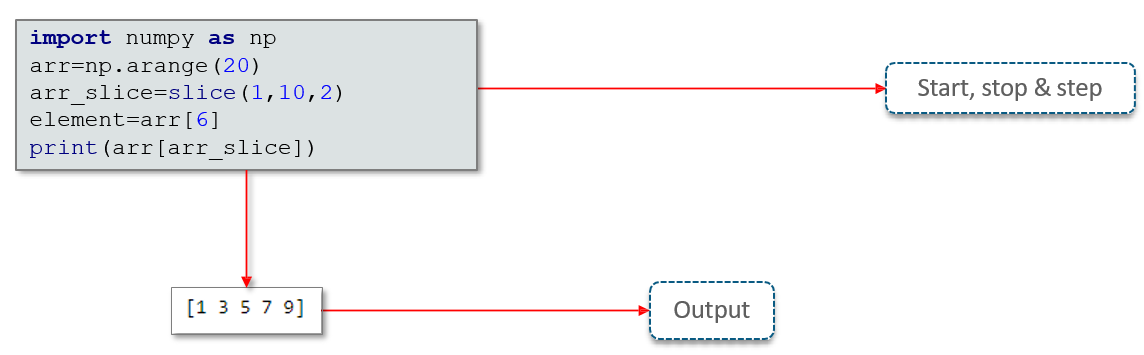
**Indexing NumPy Arrays**

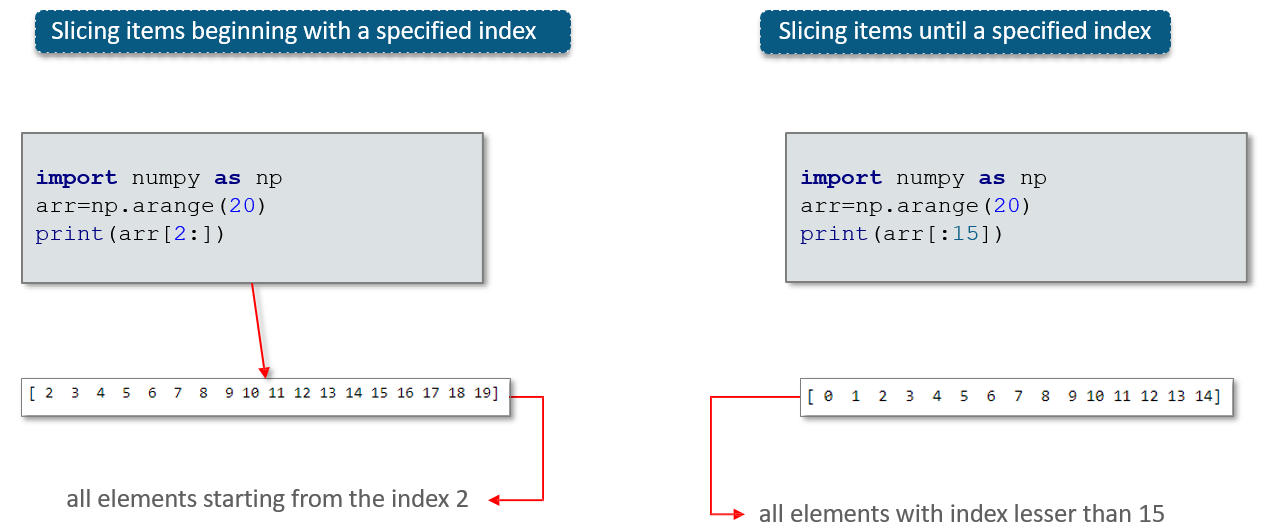
**Indexing and Slicing**

NumPy array indexing is identical to Python’s indexing scheme.

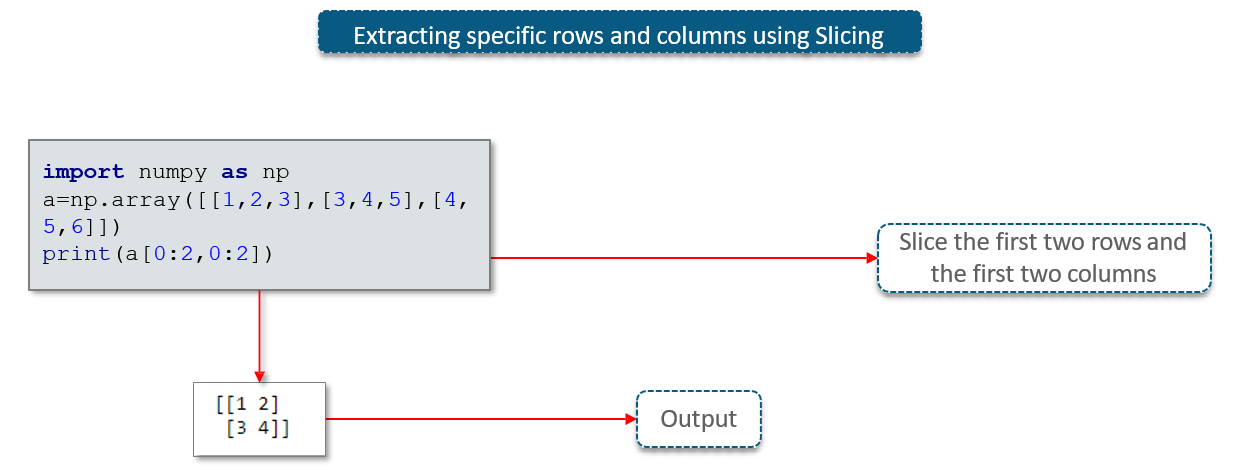


The slice object is constructed by providing start, stop, and step parameters to slice()

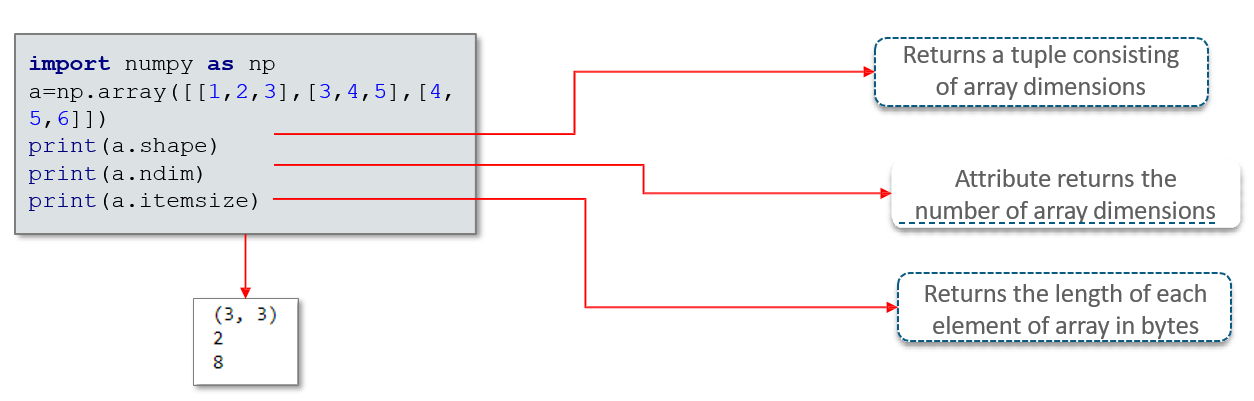




**Indexing and Slicing Multi-dimensional Arrays**



**NumPy Array Attributes**

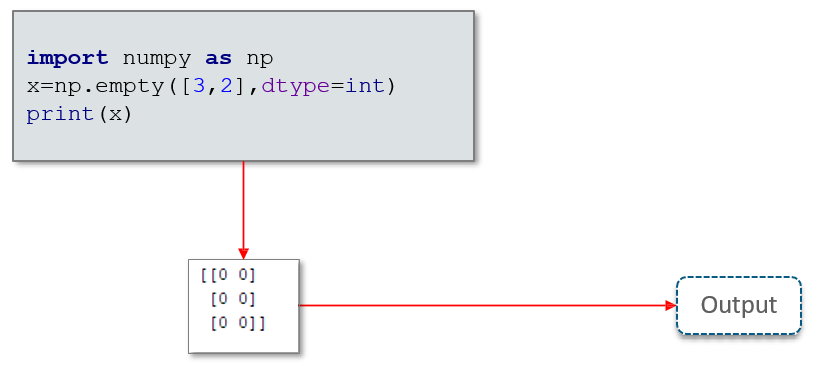


**NumPy Array Creation Routines**

**numpy.empty** – creates an uninitialized array of specified shape and dtype

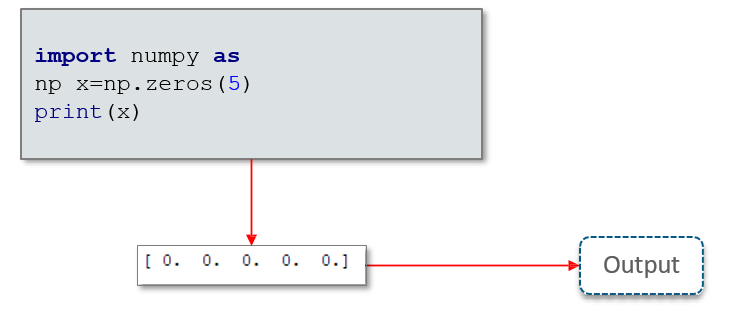
It uses the following constructor − *numpy.empty(shape, dtype = float)*

The constructor parameters are as follows – *shape, dtype*



**numpy.zeros** - returns a new array of specified size, filled with zeros.

It takes the following syntax - *numpy.zeros(shape, dtype = float)*

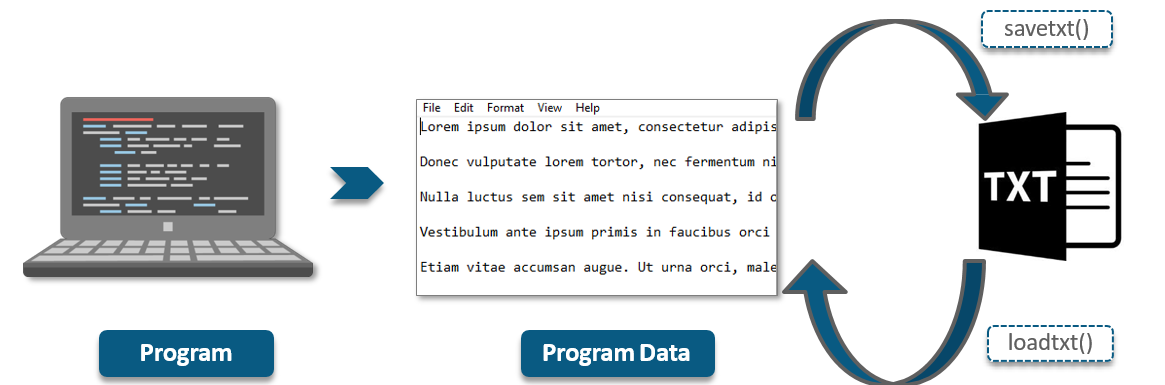


**How will you Read and Write from Text Files?**

* NumPy provides the option of importing data from files directly into ndarray using the **loadtxt** function
* The **savetxt** function can be used to write data from an array into a text file

arr = np.loadtxt('filex.txt’)

np.savetxt('newfilex.txt’,arr)



* NumPy arrays can be dumped into CSV files using the **savetxt** function and the comma delimiter
* The **genfromtxt** function can be used to read data from a CSV file into a NumPy array

arr = genfromtxt('my\_file.csv', delimiter=',’)

np.savetxt('newfilex.csv’, arr, delimiter = “,”)

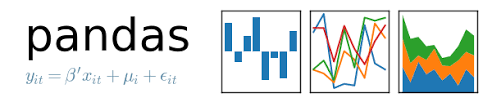


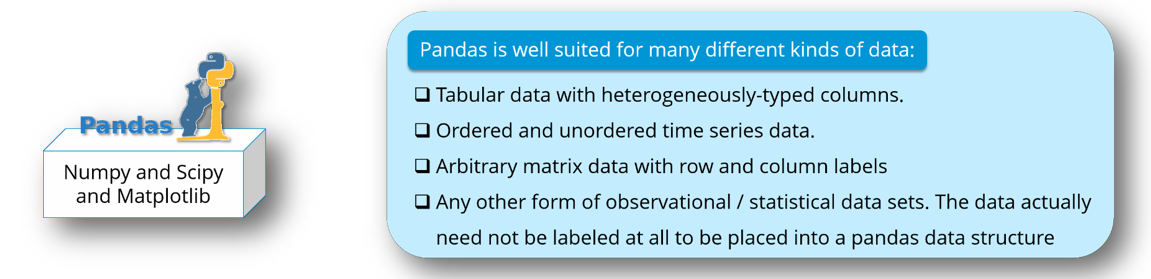
## 11.2 Pandas

**What is Pandas?**

Pandas is an open-source Python library providing efficient, easy-to-use data structures and data analysis tools.

The name Pandas is derived from “Panel Data” – an Econometrics from Multidimensional data.





**Data Structures in Pandas**

Pandas provides three data structures – **Series**, **DataFrames**, and **Panels**; all of which are built on top of the NumPy array.

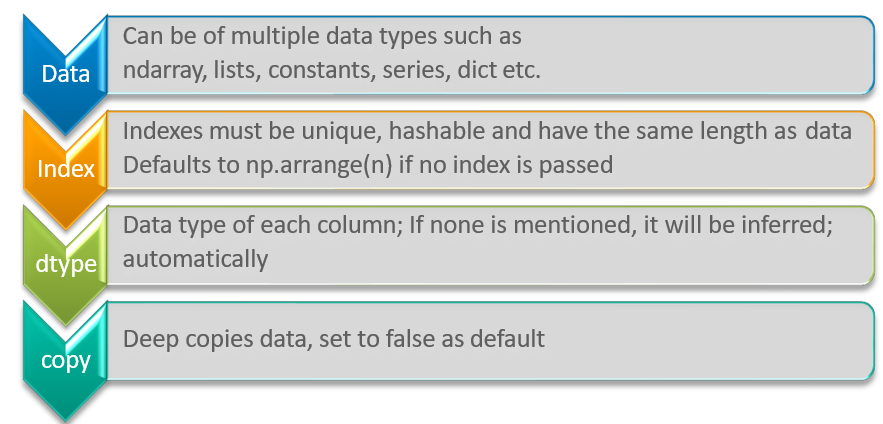
|  |  |  |
| --- | --- | --- |
| Data Structure | Dimensions | Description |
| Series | 1 | Labeled, homogenous array of immutable size |
| DataFrames | 2 | Labeled, heterogeneously typed, size-mutable tabular data structures |
| Panels | 3 | Labeled, size-mutable array |

All the above data structures are **value-mutable**

**Series**

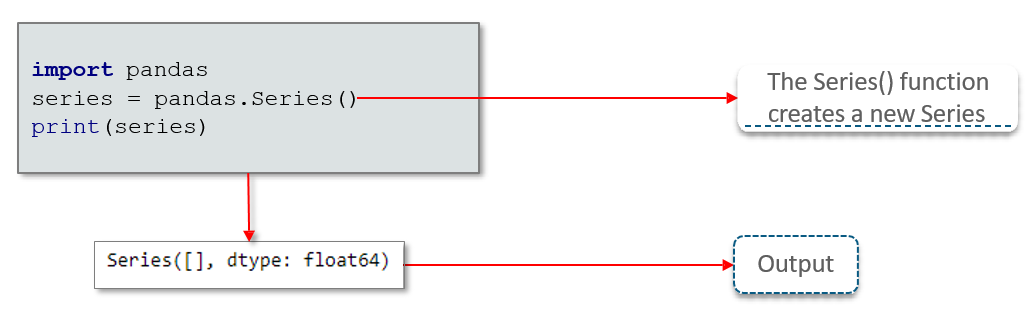
**What is Series?**

* A Series is a single-dimensional array structures that stores homogenous data i.e., data of a single type
* All the elements of a Series are **value-mutable** and **size-immutable**

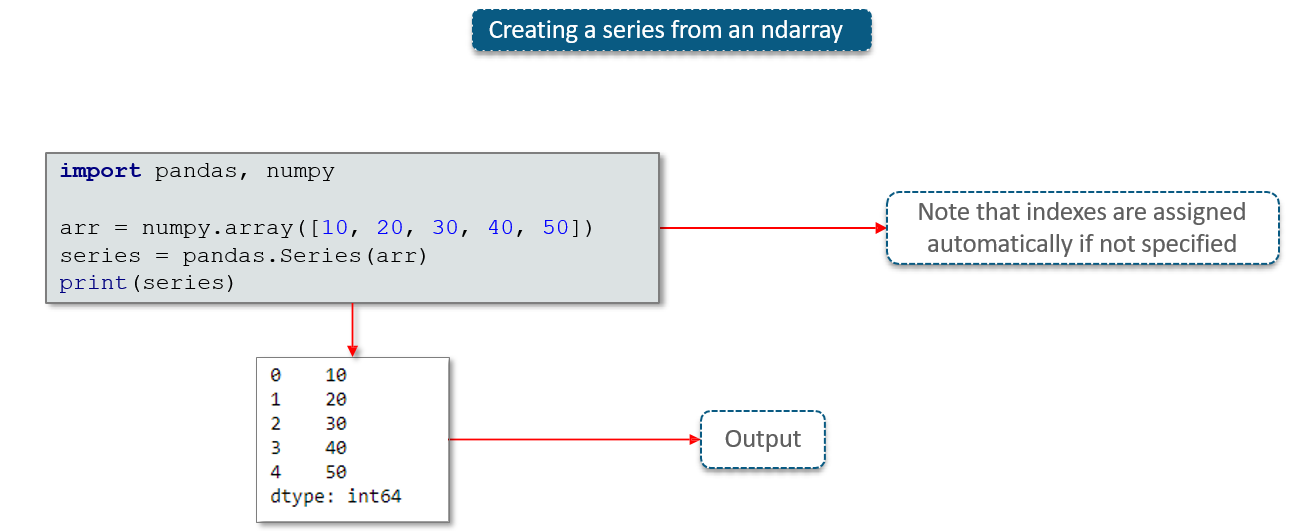


**Creating a Series**

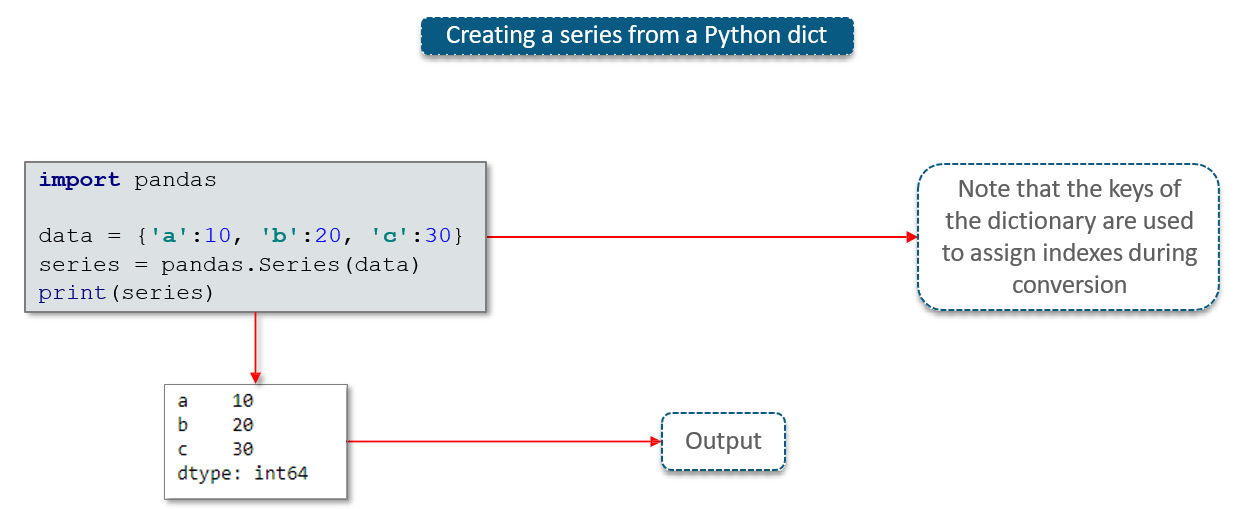
**Creating empty series**



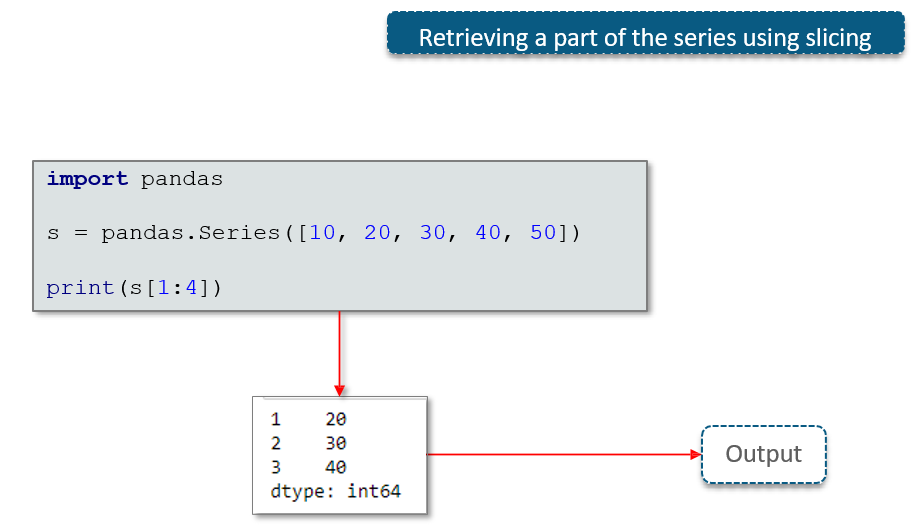
**Creating a series from an array**



**Creating a series from a Python Dict**



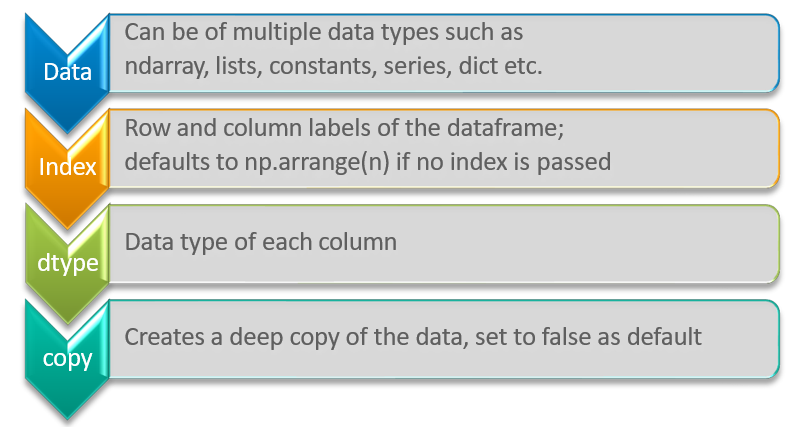
**Accessing data from a Series**

****

**DataFrames**

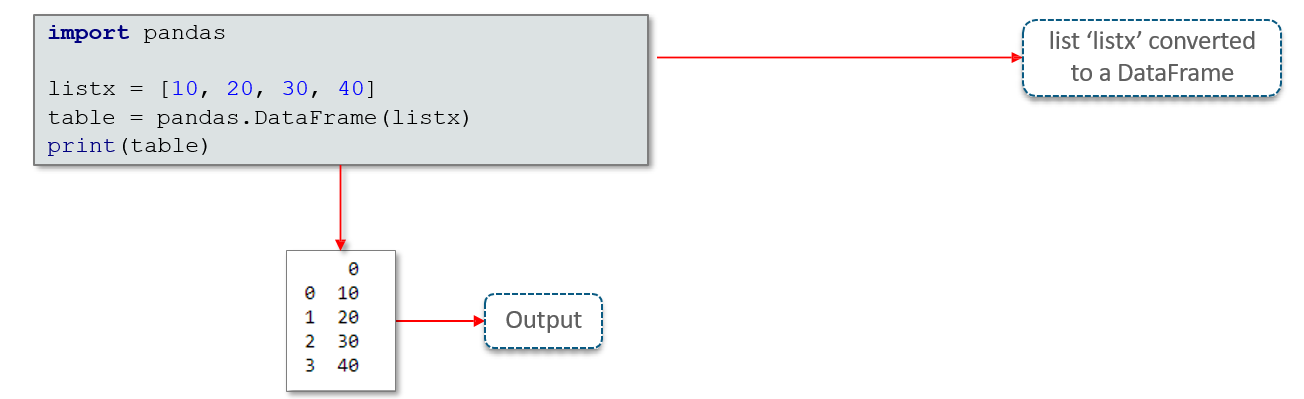
**What is Data Frames?**

* A DataFrame is a 2D data structure in which data is aligned in a tabular fashion consisting of rows & columns
* A DataFrame can be created using the following constructor − *pandas.DataFrame( data, index, dtype, copy)*

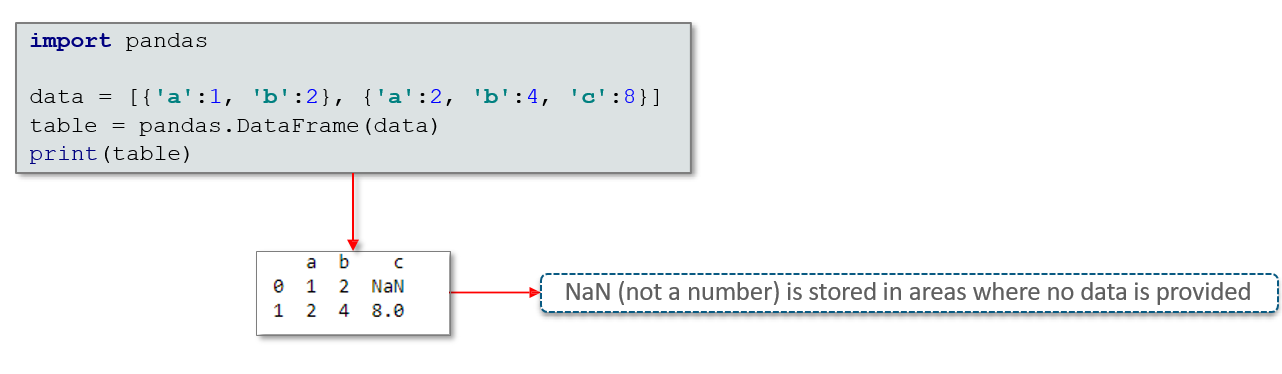


**Creating a DataFrame**

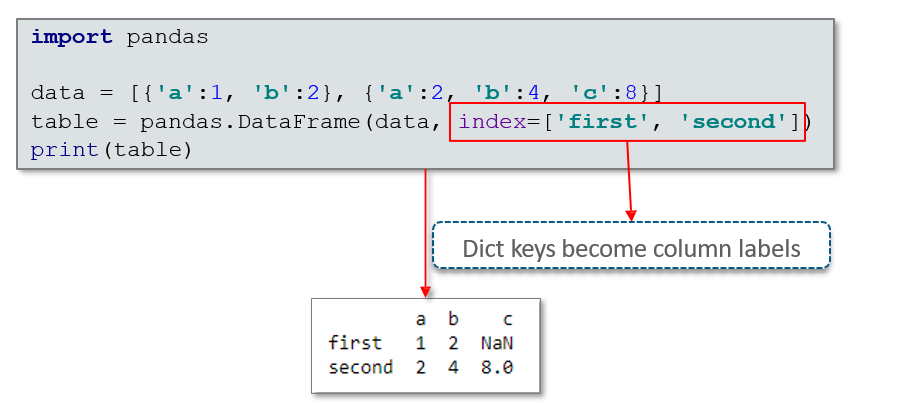
**Converting a list in to a DataFrame**



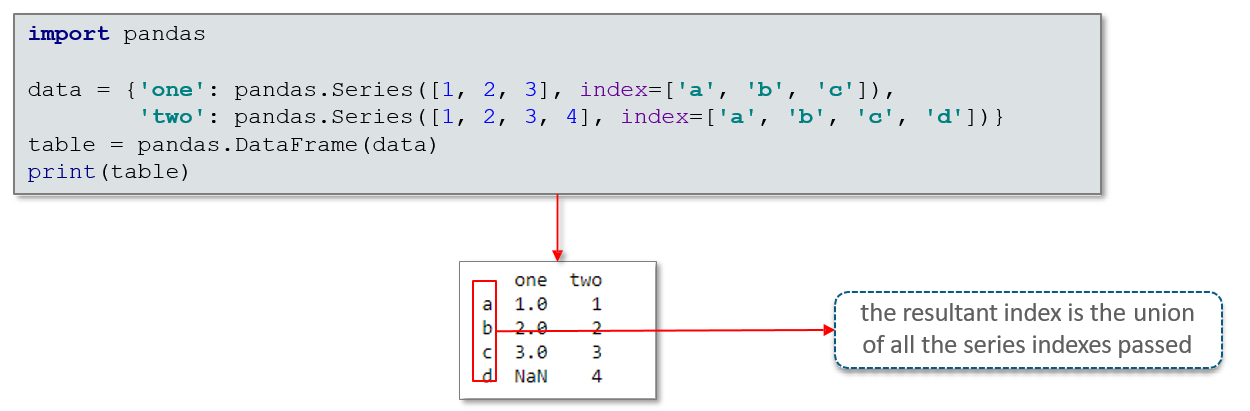
**Creating a DataFrame from a list of dictionaries**



**Creating a DataFrame from a list of dictionaries and accompanying row indices**

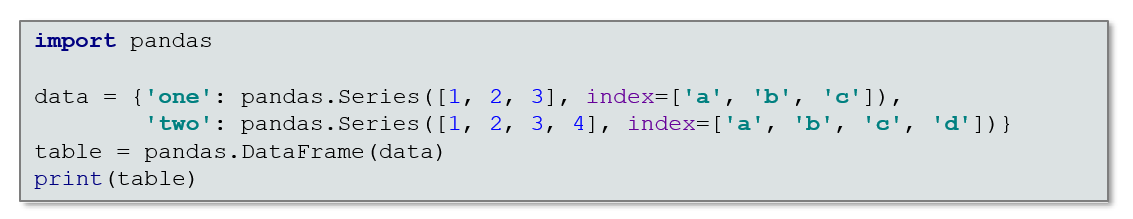


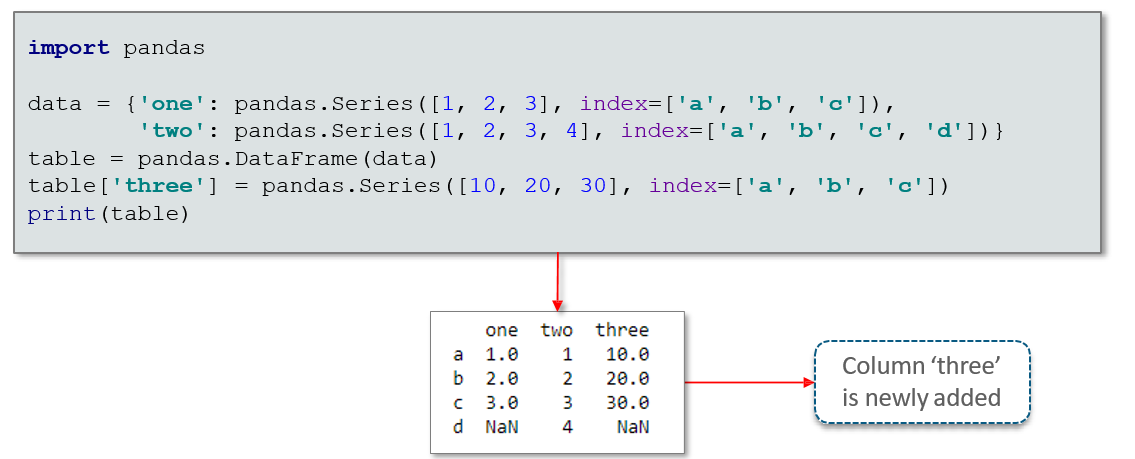
**Converting a dictionary of series into a DataFrame**



**DataFrame – Addition & Deletion of Columns**

**DataFrame – Column Addition**

A new column can be added to a DataFrame when the data is passed as a Series. 

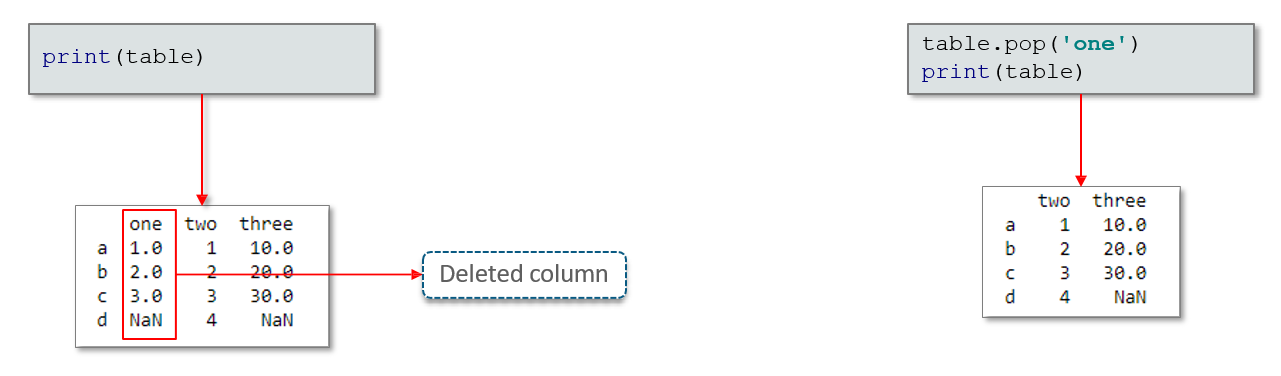


**DataFrame – Column Deletion**

DataFrame columns can be deleted using the del() function



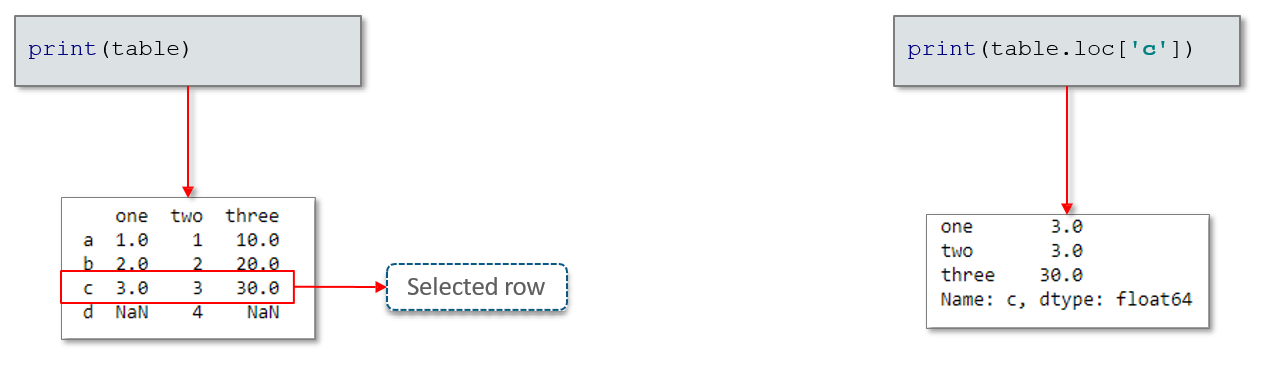
**DataFrame columns can be deleted using the pop() function**



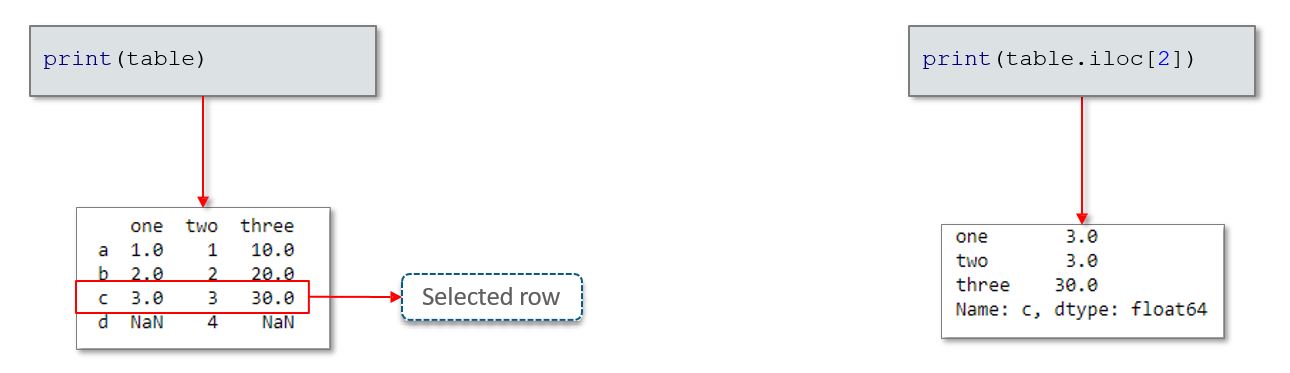
**DataFrame – Addition & Deletion of Rows**

**DataFrame – Row Selection**

DataFrame rows can be selected by passing the row label to the loc() function

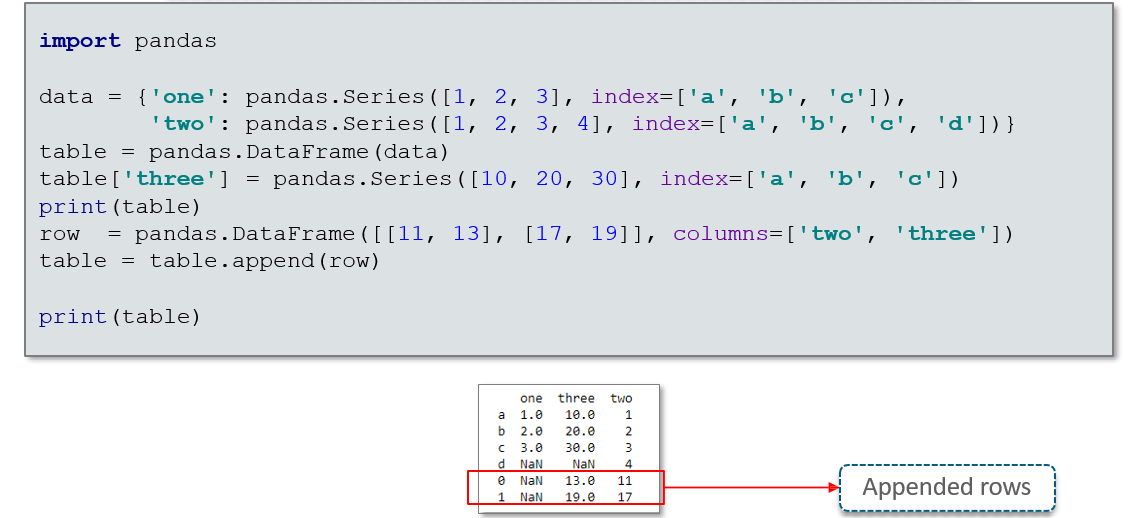


Row selection can also be done using the row index



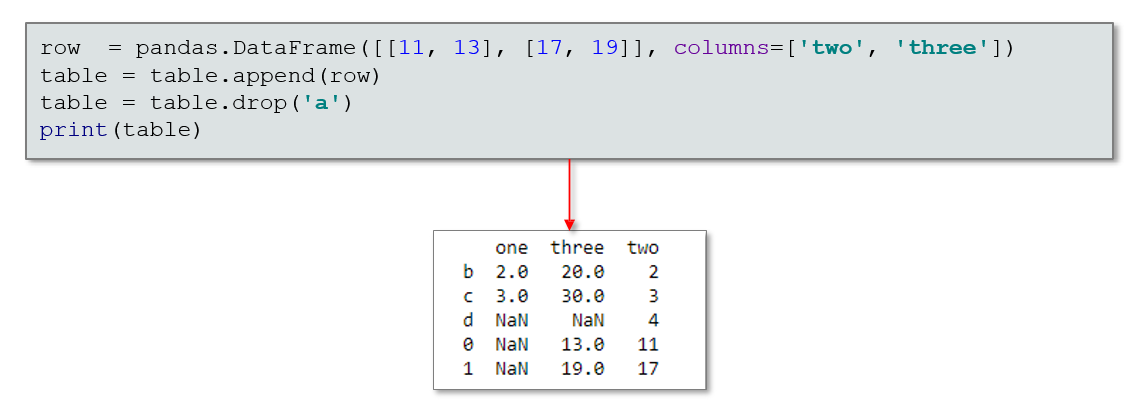
**DataFrame – Row Addition**

The append() function can be used to add more rows to the DataFrame.



**DataFrames – Row Deletion**

The drop() function can be used to drop rows whose labels are provided.



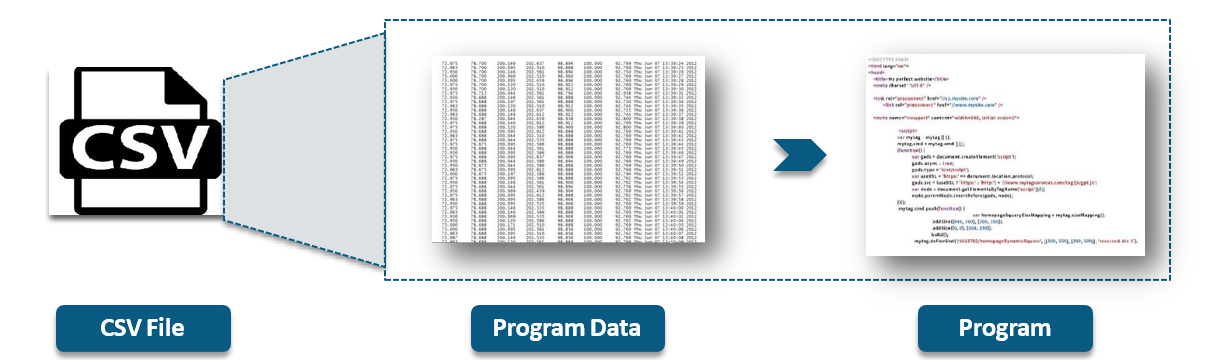
**Importing & Exporting Data**

**Loading CSV Data into DataFrames**

Data can be loaded into DataFrames from input data stored in the CSV format using the **read\_csv()** function.

|  |  |  |
| --- | --- | --- |
| table = pandas.read\_csv( | "/home/sriram/Datasets/USArrests.csv" | ) |

----------🡪 Path to file



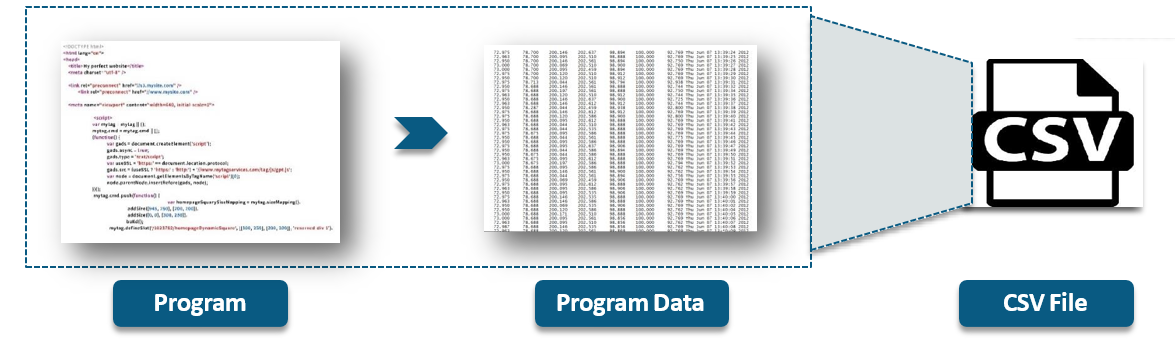
**Storing Data in CSV Files**

Data present in DataFrames can be written to a CSV file using the **to\_csv()** function

If the specified path doesn’t exist, a file of the same name is automatically created

|  |  |  |
| --- | --- | --- |
| table = pandas.read\_csv( | "/home/sriram/Datasets/USArrests2.csv" | ) |

----------🡪 Path to file

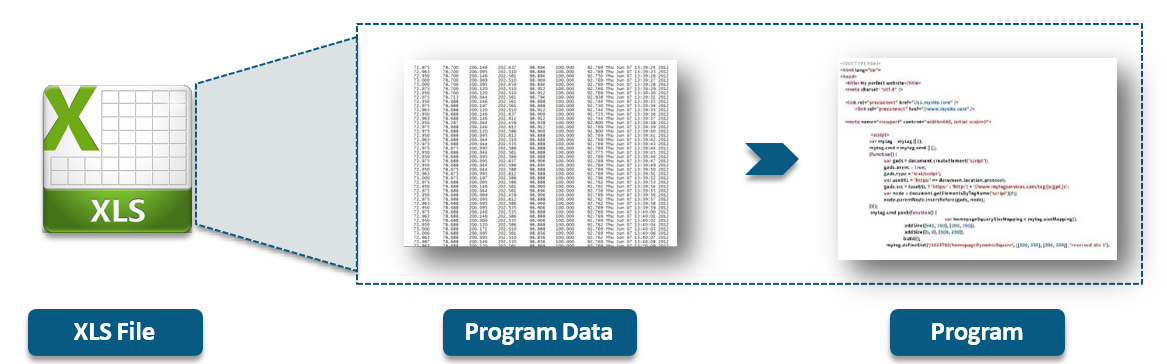


**Loading Excel sheet Data into Pandas**

Data can be loaded into DataFrames from input data stored in the Excelsheet format using **read\_excel()**

|  |  |  |
| --- | --- | --- |
| sheet = pandas.read\_excel( |  | ) |
| "/home/sriram/Datasets/USArrests.xlsx" |

----------🡪 Path to file

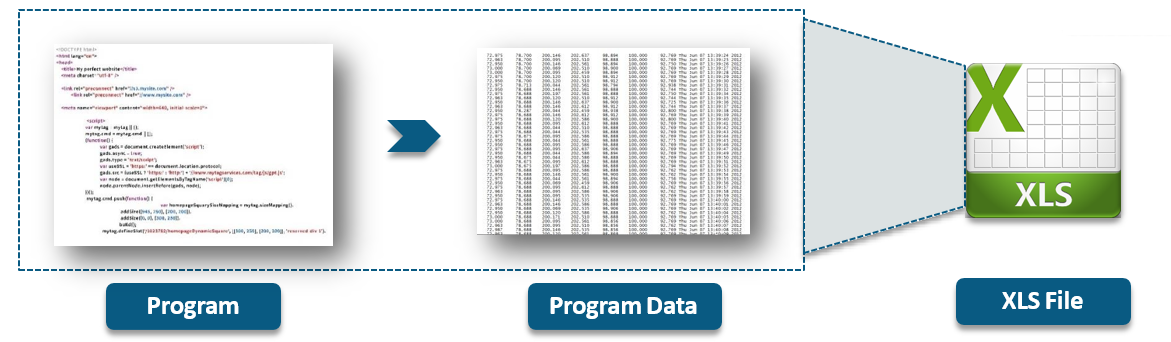


**Storing Data in Excel Files**

* Data present in DataFrames can be written to a spreadsheet file using **to\_excel()**
* If the specified path doesn’t exist, a file of the same name is automatically created

|  |  |  |
| --- | --- | --- |
| sheet = pandas.read\_excel( |  | ) |
| "/home/sriram/Datasets/USArrests2.xlsx" |

----------🡪 Path to file



## 11.3 Matplotlib

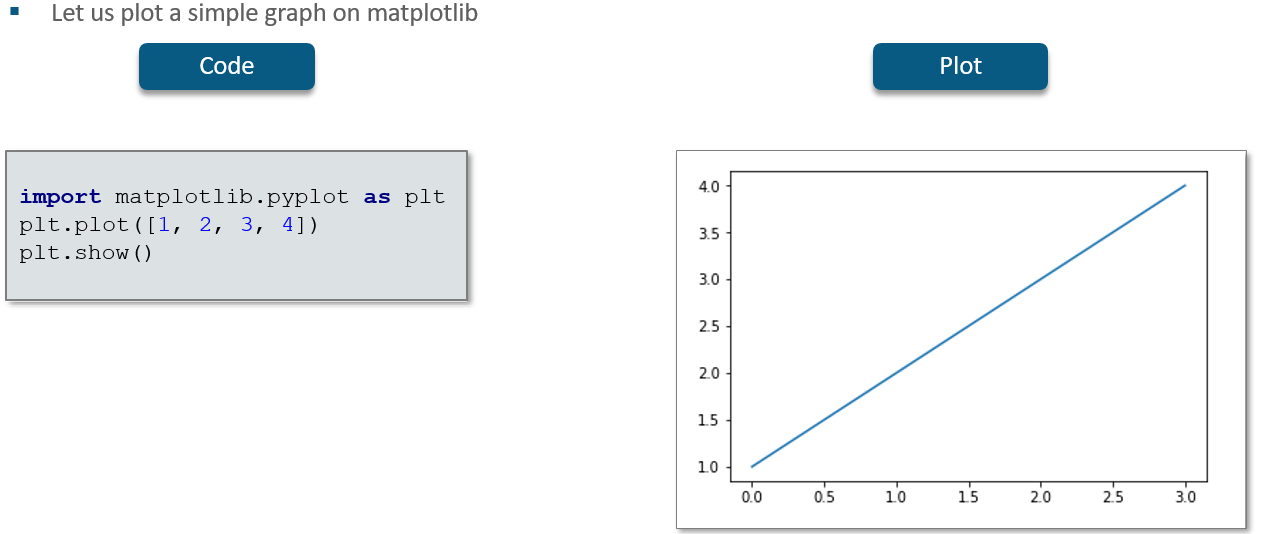
**What is Matplotlib?**

* Matplotlib is a Python library that is specially designed for the development of graphs, charts etc., in order to provide interactive data visualisation
* Matplotlib is inspired from the MATLAB software and reproduces many of its features

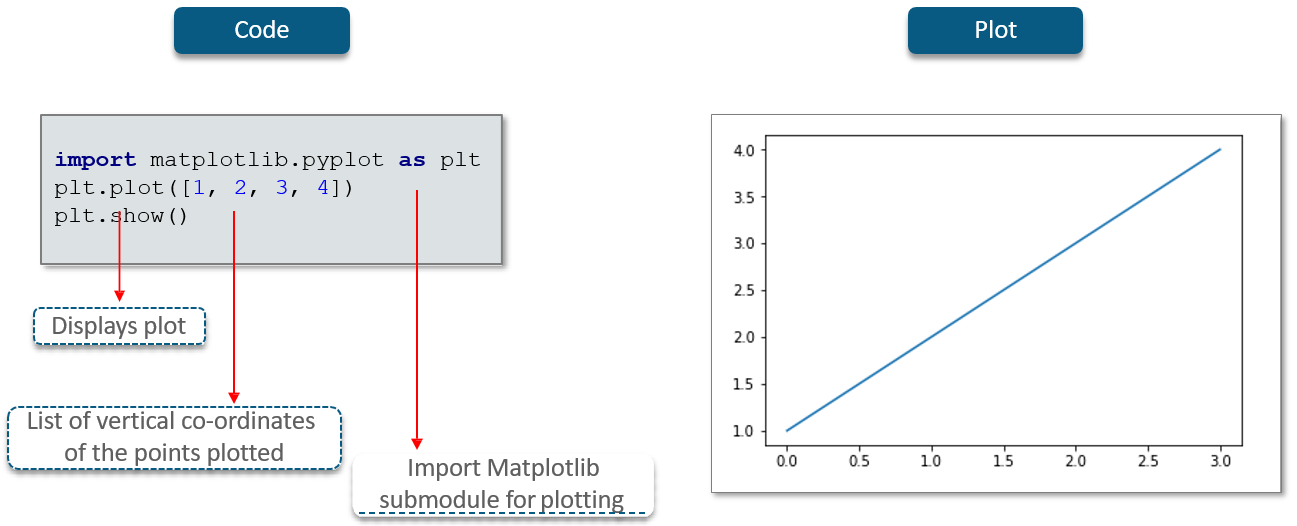


**Plotting in Matplotlib**

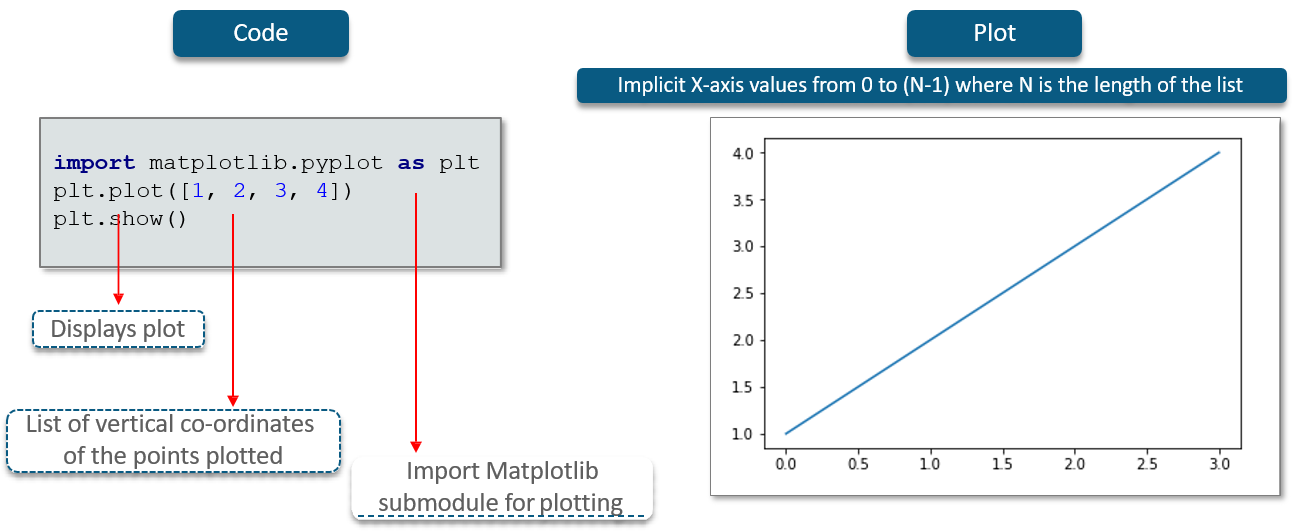
**First Plot with Matplotlib**



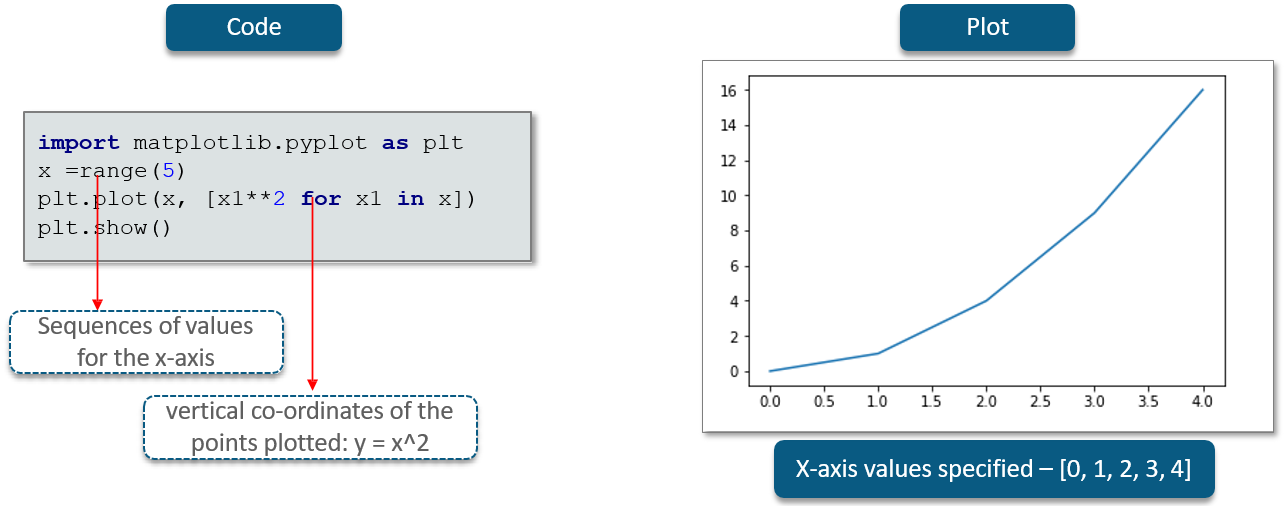
Let us plot a simple graph on matplotlib



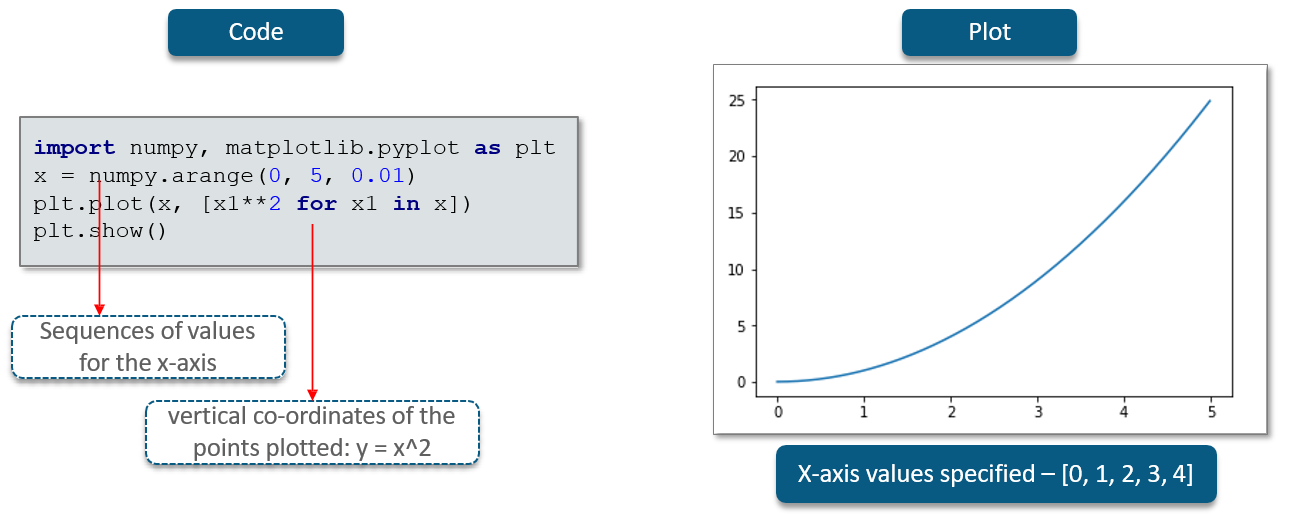
Let us plot a simple graph on matplotlib



We can specify both the axes



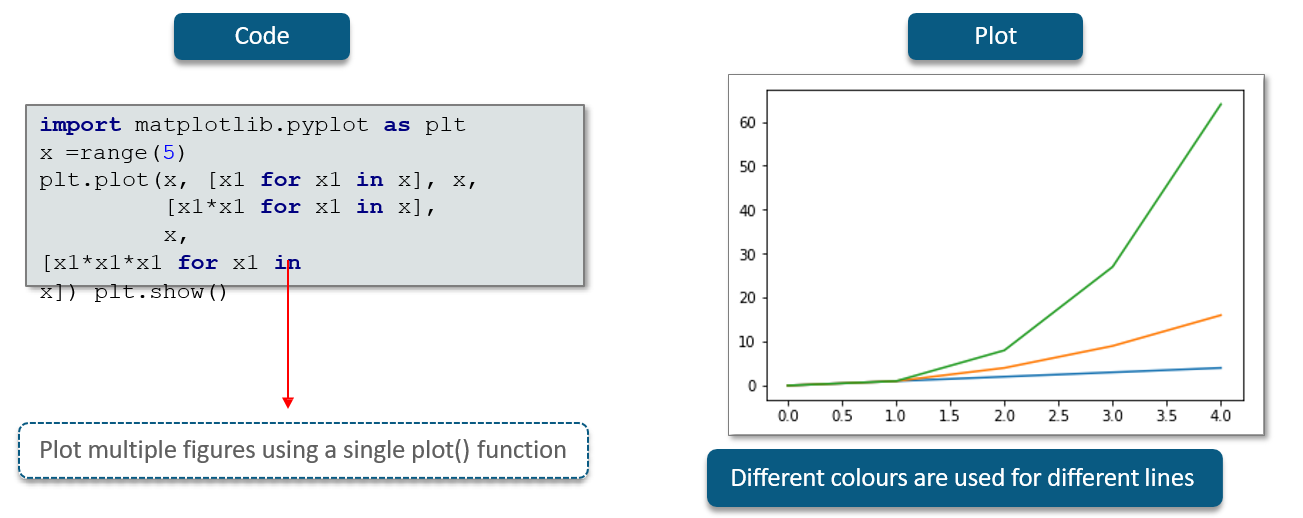
We can use NumPy to specify the values for both axes with greater precision



**Multiline Plots**

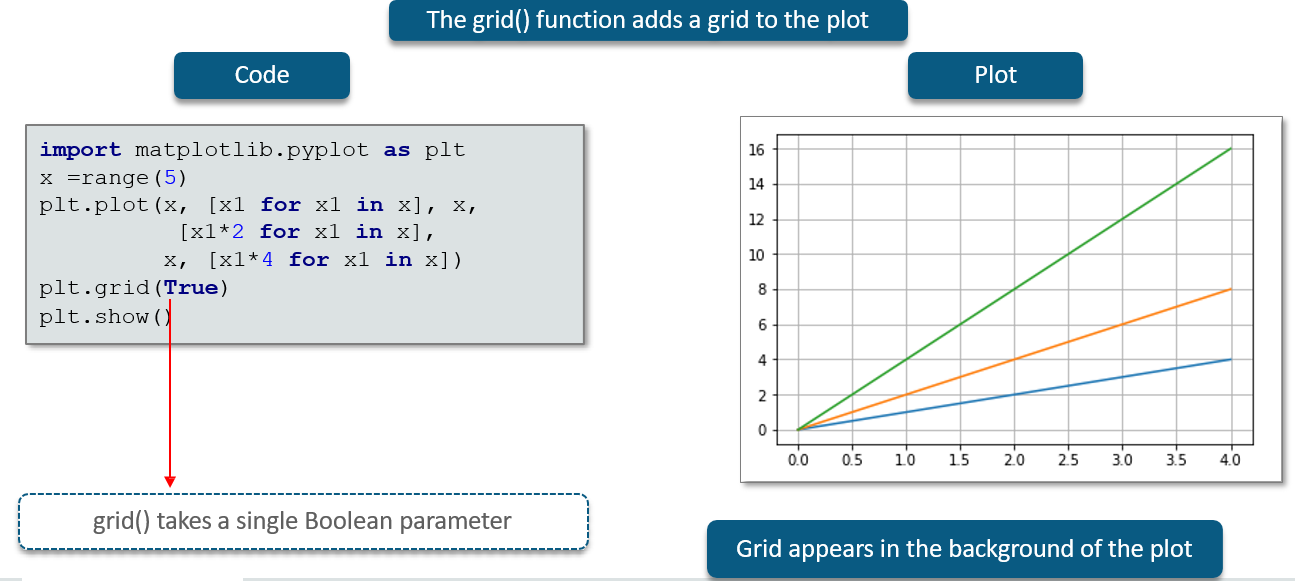
* Multiple functions can be drawn on the same plot



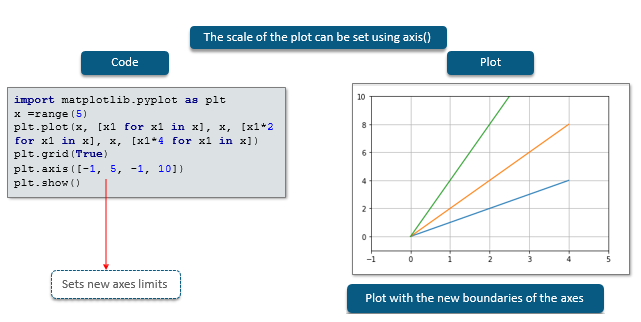


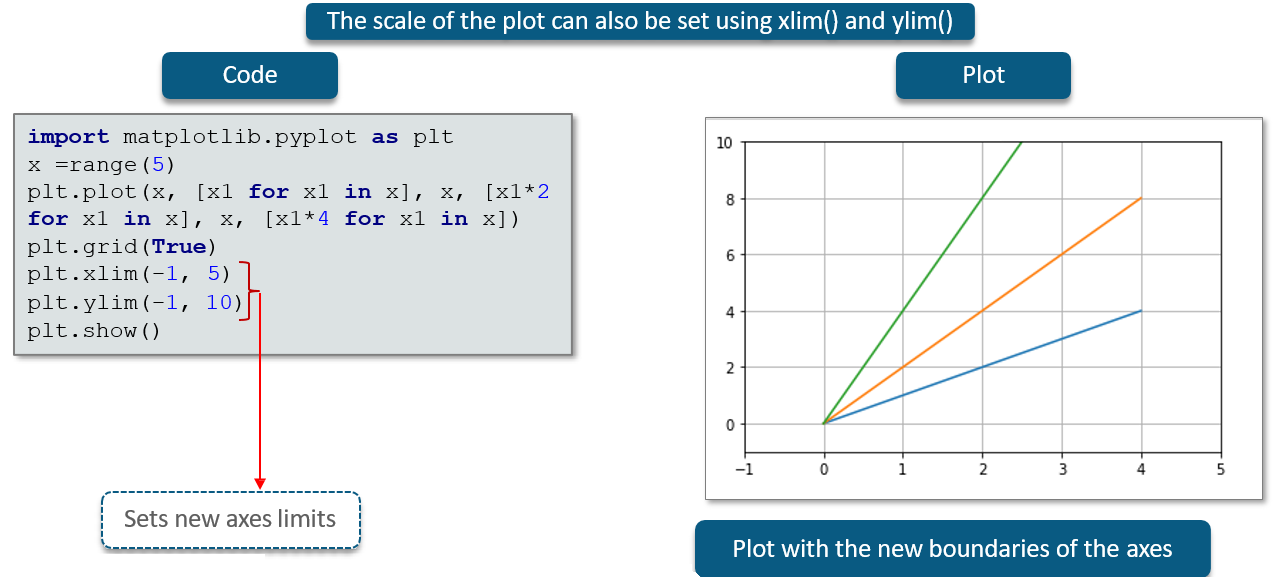
**Grids**

**Adding a Grid**

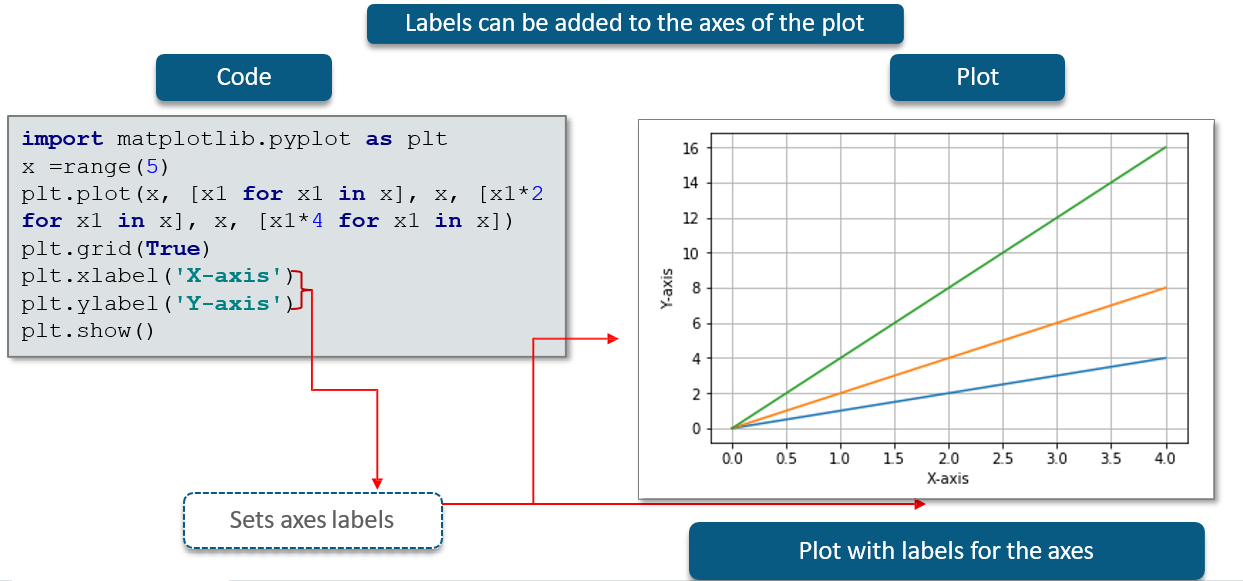


**Limiting the Axes**

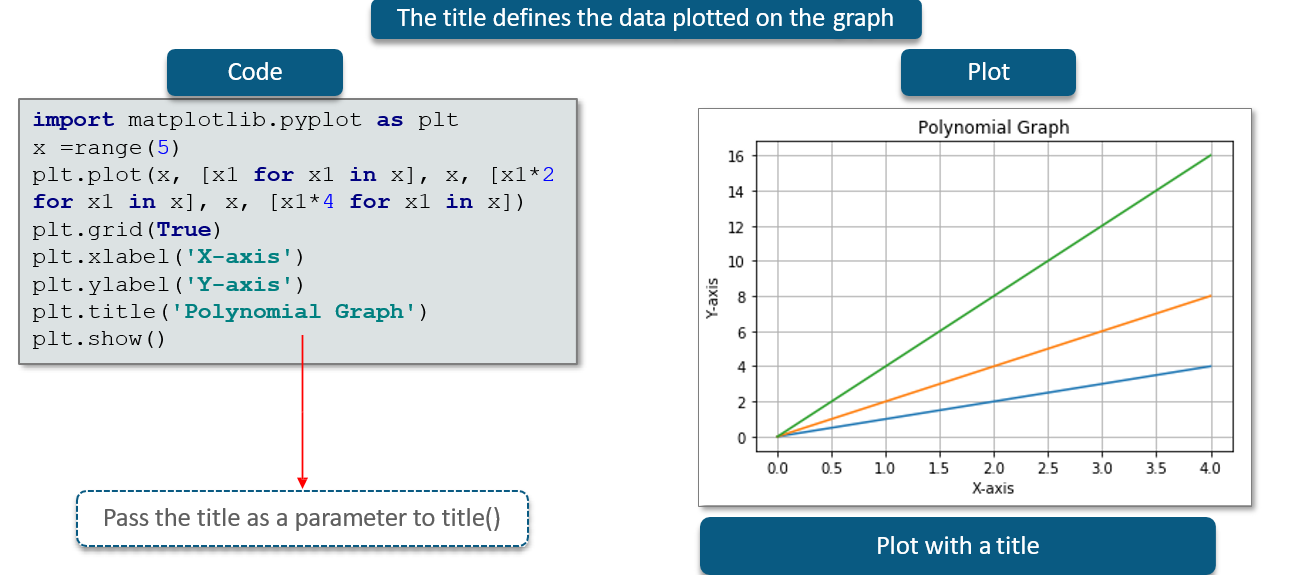




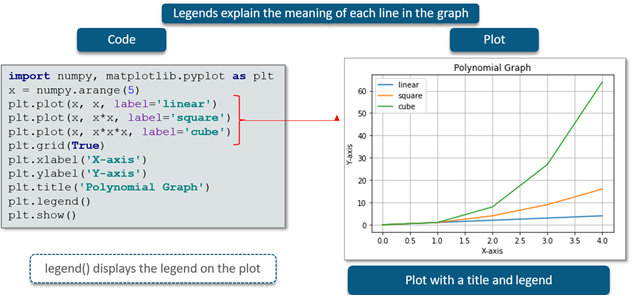
**Adding Labels**



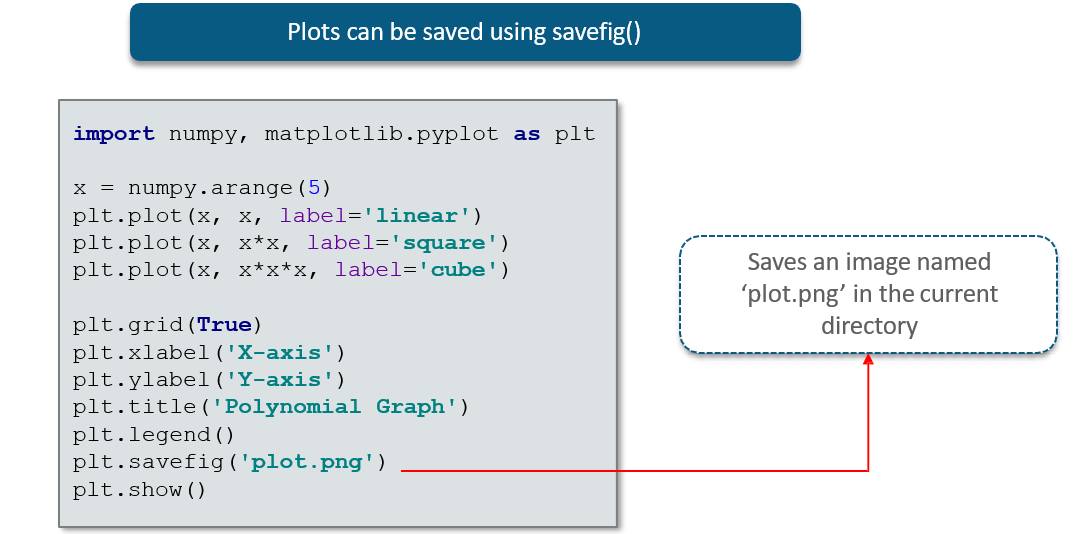
**Adding the Title**

****

**Adding a Legend**



**Saving Plots**

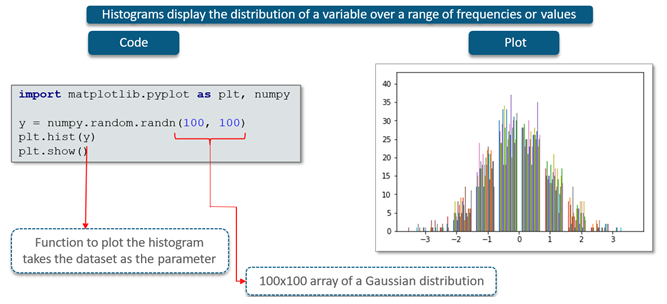


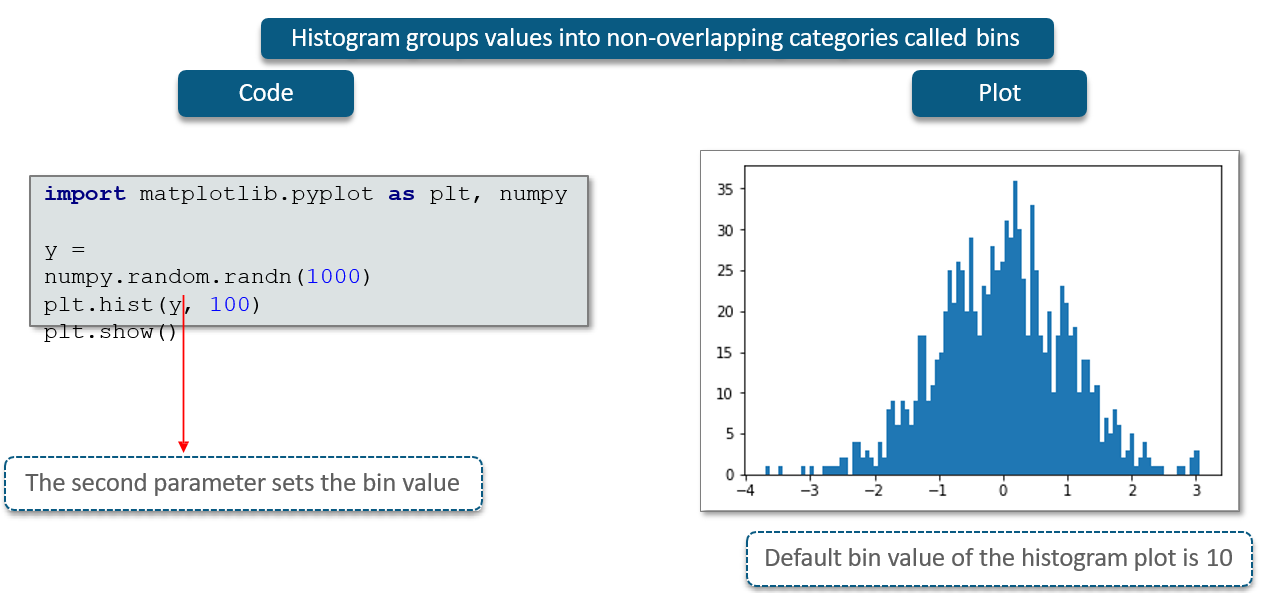
**Plot Types**

**Types of Plots**

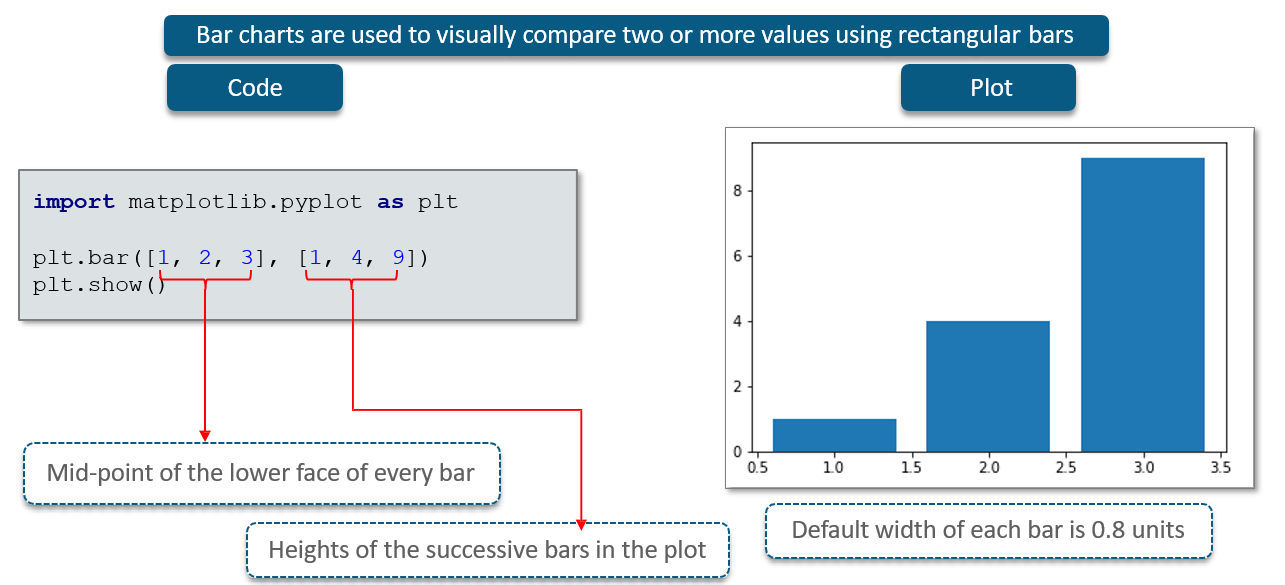


**Histogram**

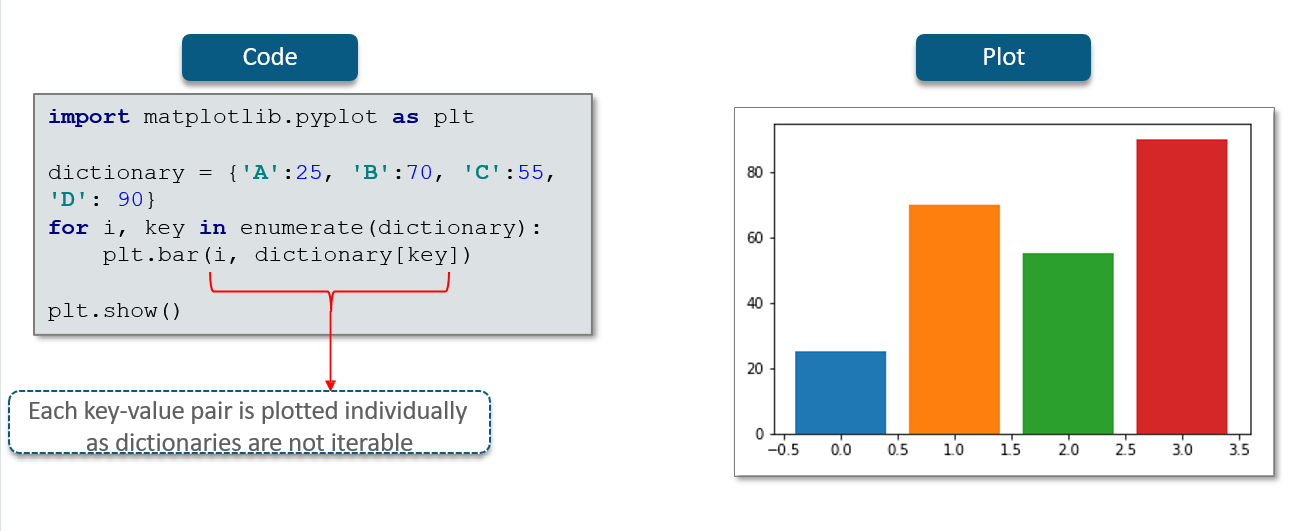


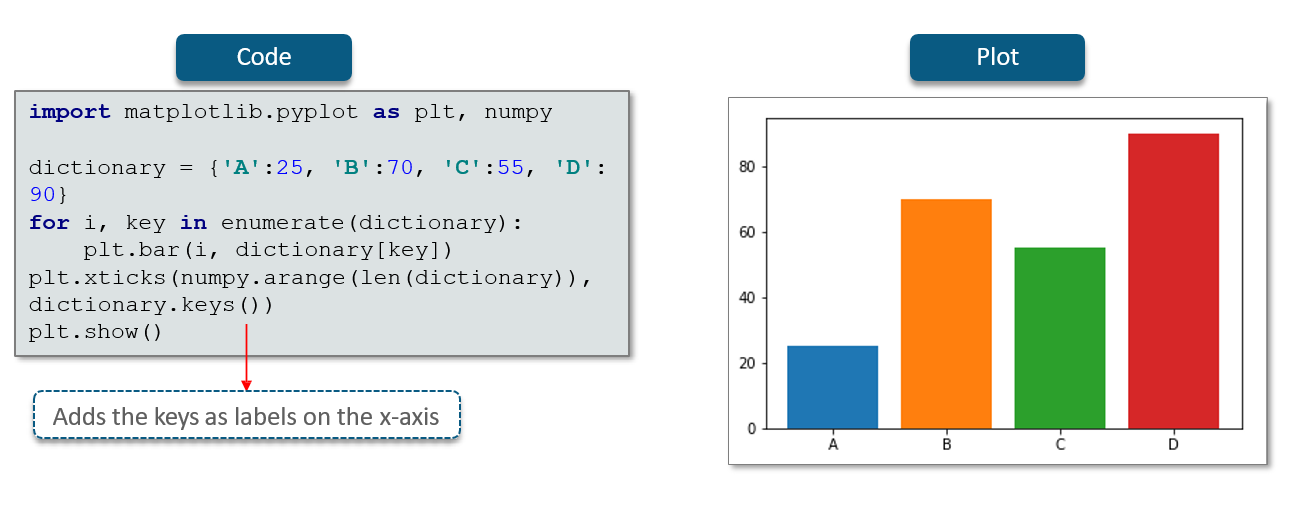


**Bar Chart**

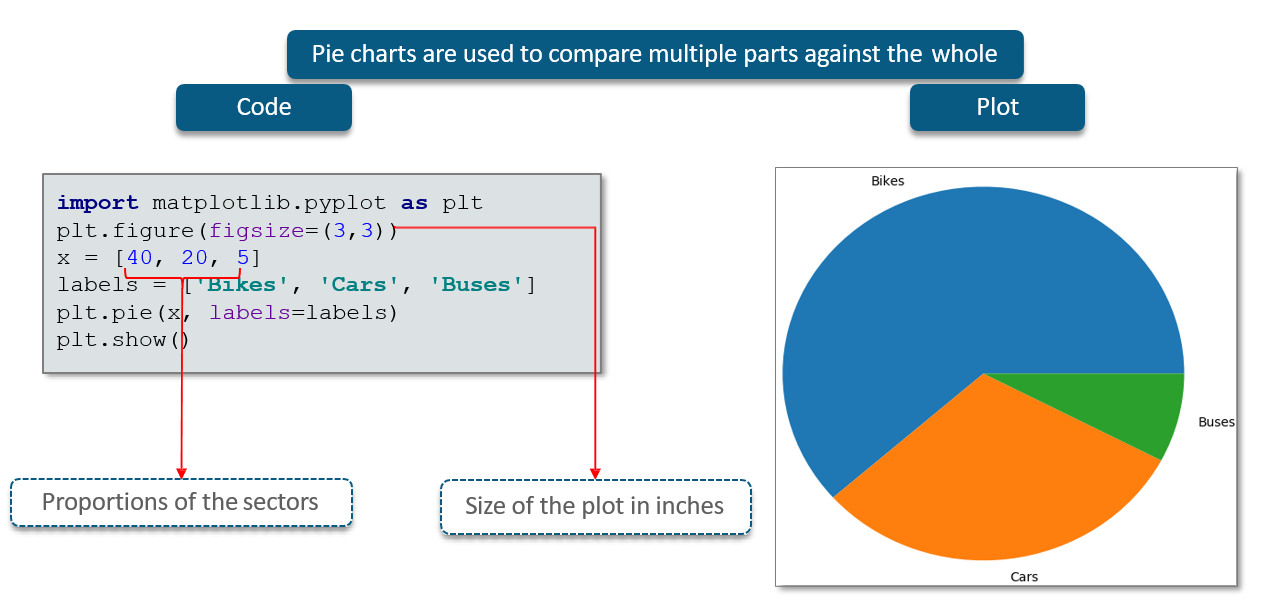


**Plotting a Dictionary using Bar Chart**

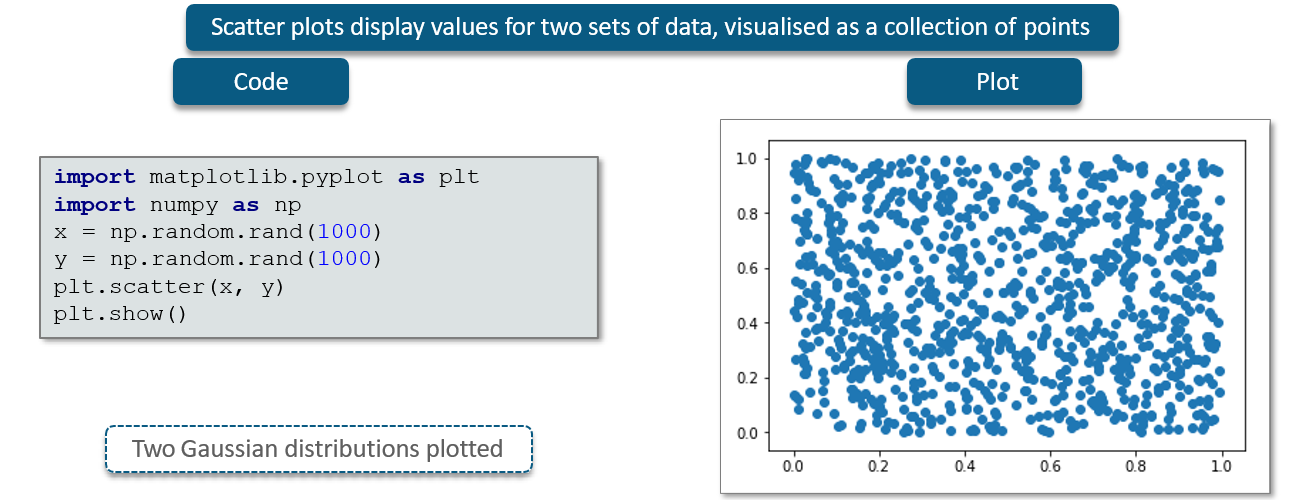




**Pie Chart**

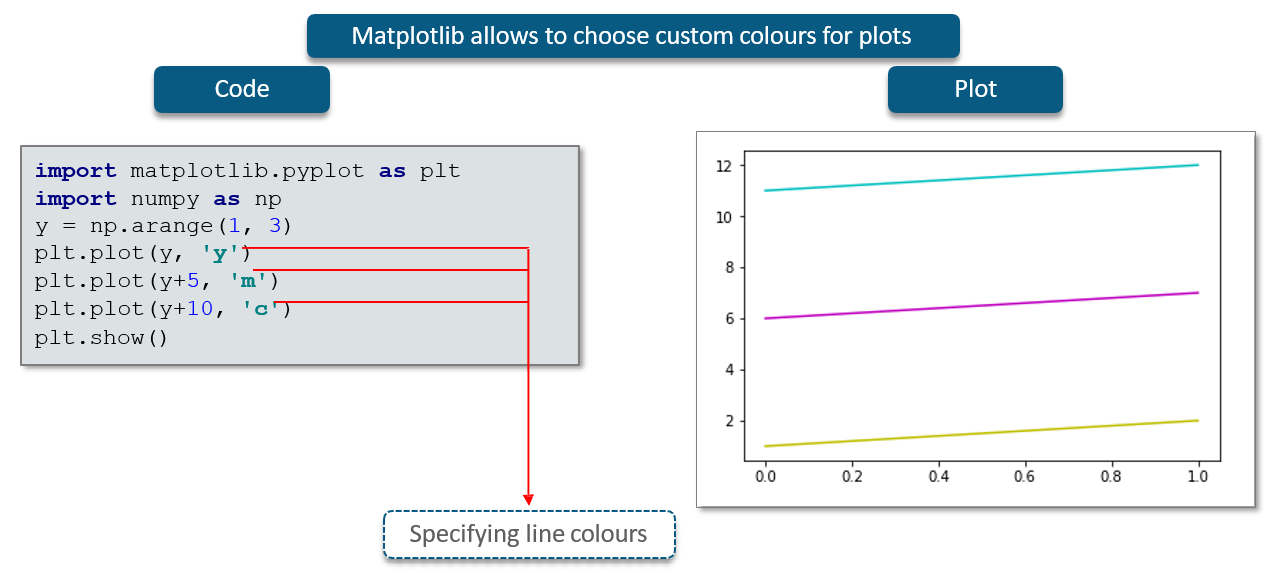


**Scatter Plot**



**Styling**

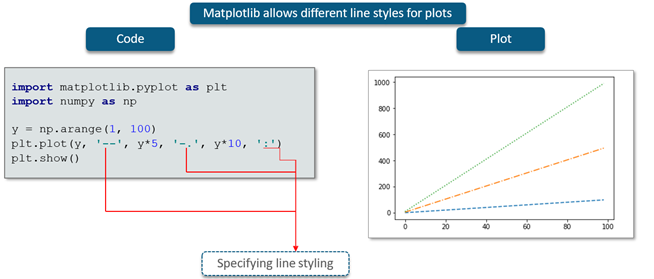
**Control Colours**

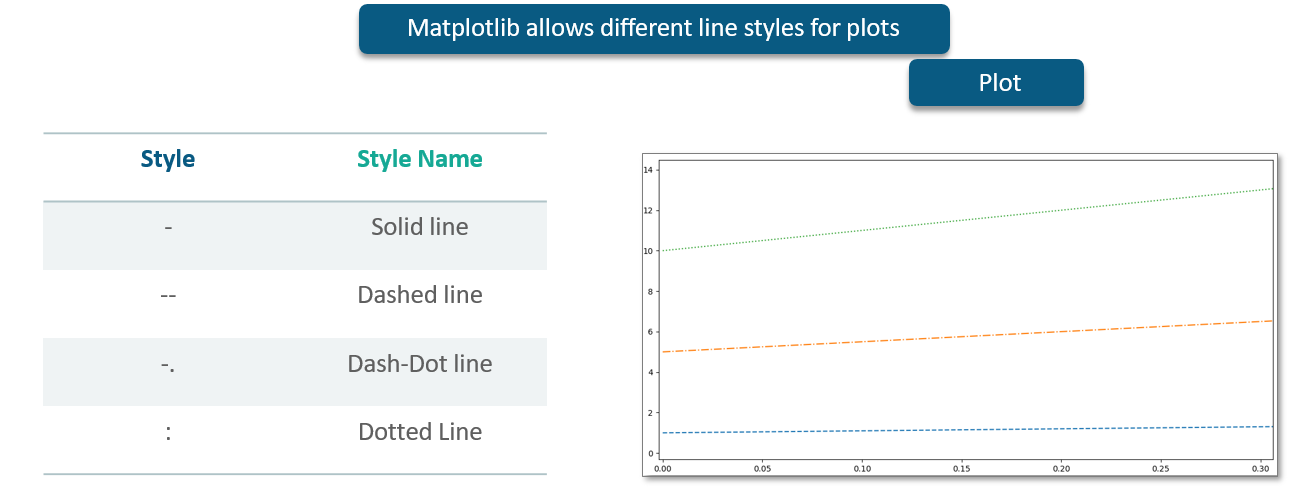


**Control Colors – Codes**

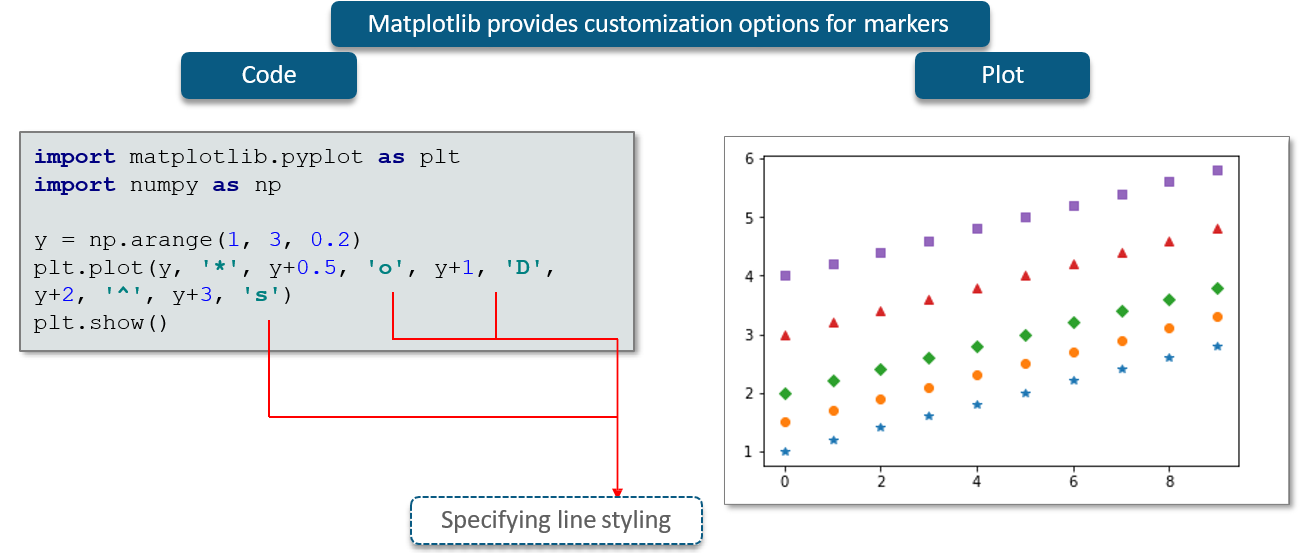
|  |  |
| --- | --- |
|  |  |
| **Color code** | **Color** |
| **b** | **Blue** |
| **c** | **Cyan** |
| **g** | **Green** |
| **k** | **Black** |
| **m** | **Magenta** |
| **r** | **Red** |
| **w** | **White** |
| **y** | **Yellow** |

**Control Line Styling**

****

****

**Control Marker Styling**

****

## 11.4 SciPy

**What is SciPy?**

SciPy is a scientific library for Python.

SciPy is an Open Source.

SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation.

The main reason for building the SciPy library is that, it should work with NumPy arrays.

**What are the SciPy Sub Packages?**

SciPy is organized into sub-packages covering different scientific computing domains.

Scipy.ndimage n-dimensional image package.

**What is SciPy NdImage?**

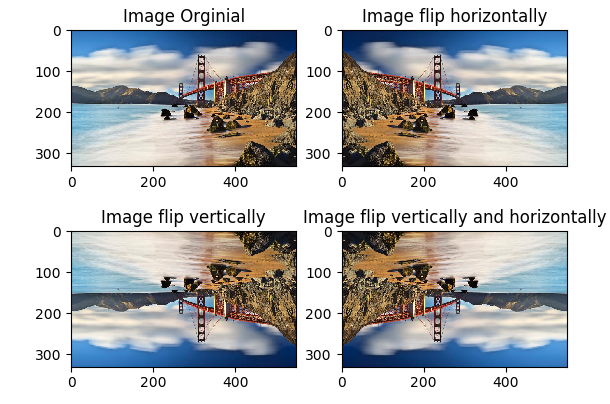
Scipy.ndimage n-dimensional image package.

Scipy.ndimage sub module is dedicated to image processing.

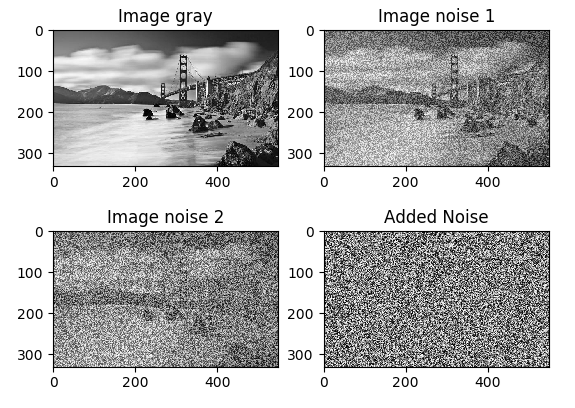
**Basic Image Processing Operations**

|  |  |
| --- | --- |
| from scipy import misc  image=misc.imread('D:\\san.jpg')  print(type(image))  print(image.shape)  print(width)  print(height)  print(dim)  import matplotlib.pyplot as plt  plt.imshow(image)  plt.show() |  |

**Flipping**



**Noise in Images**



**Gray Scale**



**Conversion of RGB color model to HSV model**

