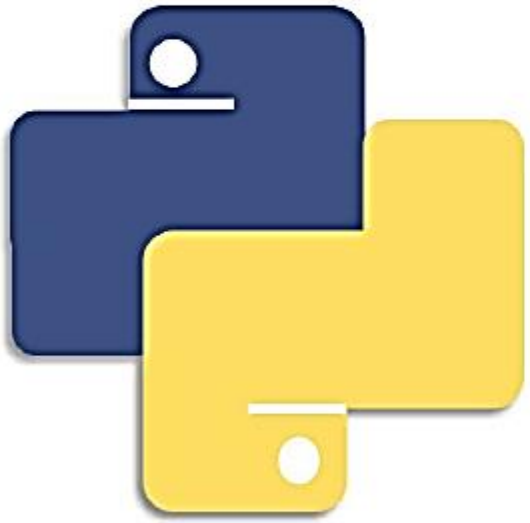
Python

By

Dr. rabih ali sbera

Department of Computer Engineering Techniques-2 nd

2024-2025



Python Programming

Python library

What Are Python Libraries?

- A Python library refers to a collection of related modules linked together. It contains a code bundle that developers can use repeatedly in different programs, making programming faster and more seamless as there is no need for writing the same code again and again for varied projects. Moreover, each library in Python has its source code

Numpy

- Numpy stands for Numerical Python, and it is among the most popular open-source libraries in Python. It supports multi-dimensional data, built-in mathematical functions, and large matrices, making it easier for mathematical and scientific computations. We can also use it as an N-dimensional container for different types of data and use the NumPy array Python object to define an N-dimensional array with rows and columns. TensorFlow also uses Numpy internally for various operations. We can also use it as a random number generator.

Matplotlib

- We use Matplotlib to plot numerical data, making it a crucial library for data analysis. This open-source library in Python plots highly defined figures, such as graphs, pie charts, scatterplots, and histograms.

Pandas

- Pandas is an open-source machine learning library in Python authorized under the Berkeley Programming Conveyance (BSD). Data scientists commonly use it as it offers flexible data structures and varied analysis tools. In data science, it is used for control, examination, data manipulation, information cleaning, and other similar tasks. It also supports iteration, visualization, sorting, conversion, re-indexing, aggregation, and concatenation. With Pandas, you can perform data modeling and analysis easily without switching to other programming languages.

SciPy

- SciPy stands for Scientific Python, and it is built over an extension of Numpy. Therefore, it works smoothly with Numpy to handle complex tasks, where Numpy focuses on array data sorting and indexing, whereas SciPy stores numerical data code. This open-source library in Python is used for information processing, scientific computing, and other high-level computations, as it contains various easy-to-use methods and functions that make computing tasks faster and easier. A few common subpackages of SciPy include io, odr, cluster, optimize, special, sparse, constants, stats, interpolate, spatial, integrate, etc.

Keras

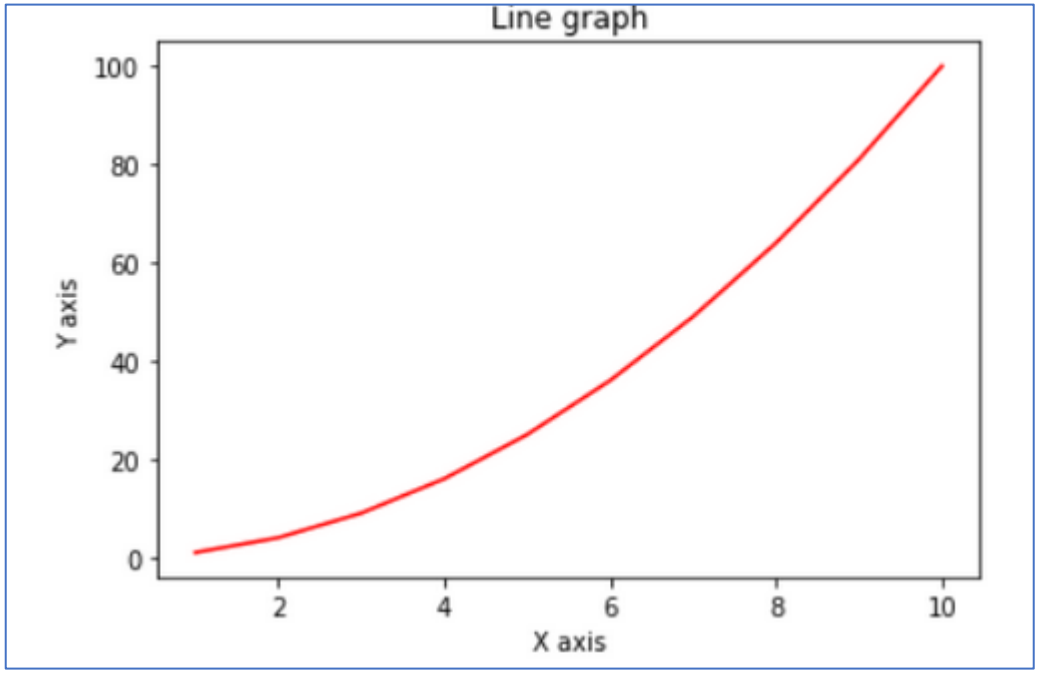
- We can’t complete our list of libraries in Python without Keras. It is an open-source neural network library that allows users to conduct extensive research into deep neural networks. With deep learning becoming a common concept, Keras is considered a viable option as it is an API designed for humans and not machines. The research community and industry prefer Keras over Theano and TensorFlow. Please note that you must download the TensorFlow backend engine before installing Keras.

Scikit-learn

- This popular open-source Python library is used to work with complex data. It works with SciPy and Numpy libraries. It supports machine learning and several supervised and unsupervised algorithms, including classification, regression, and clustering.

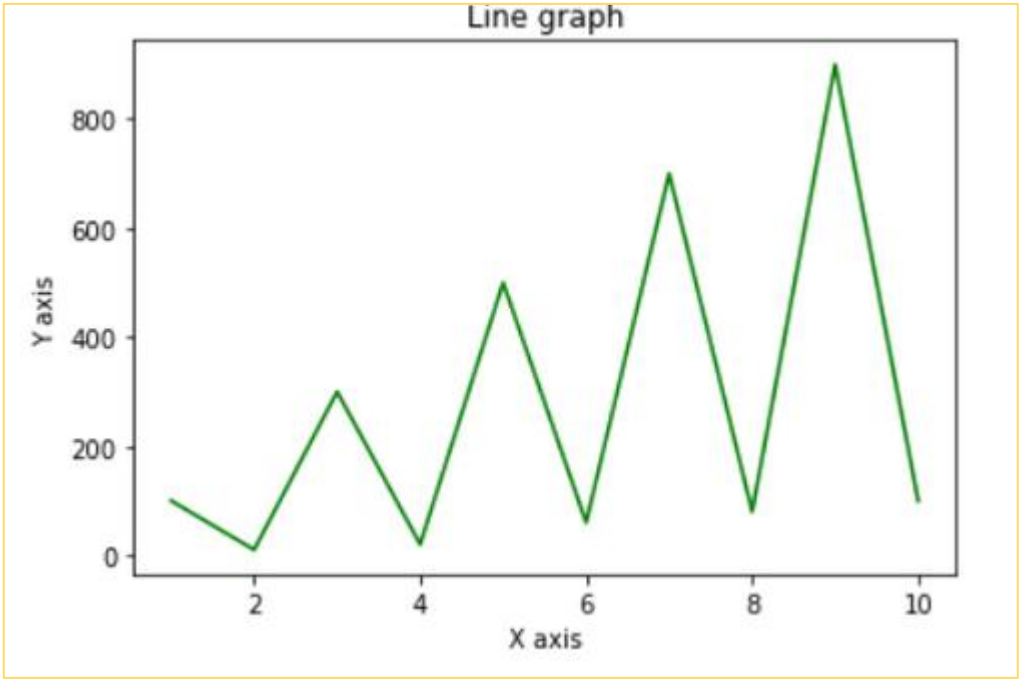
Examples

```
# importing the modules
import numpy as np
import matplotlib.pyplot as plt

# data to be plotted
x = np.arange(1, 11)
y = x * x

# plotting
plt.title("Line graph")
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.plot(x, y, color ="red")
plt.show()
```



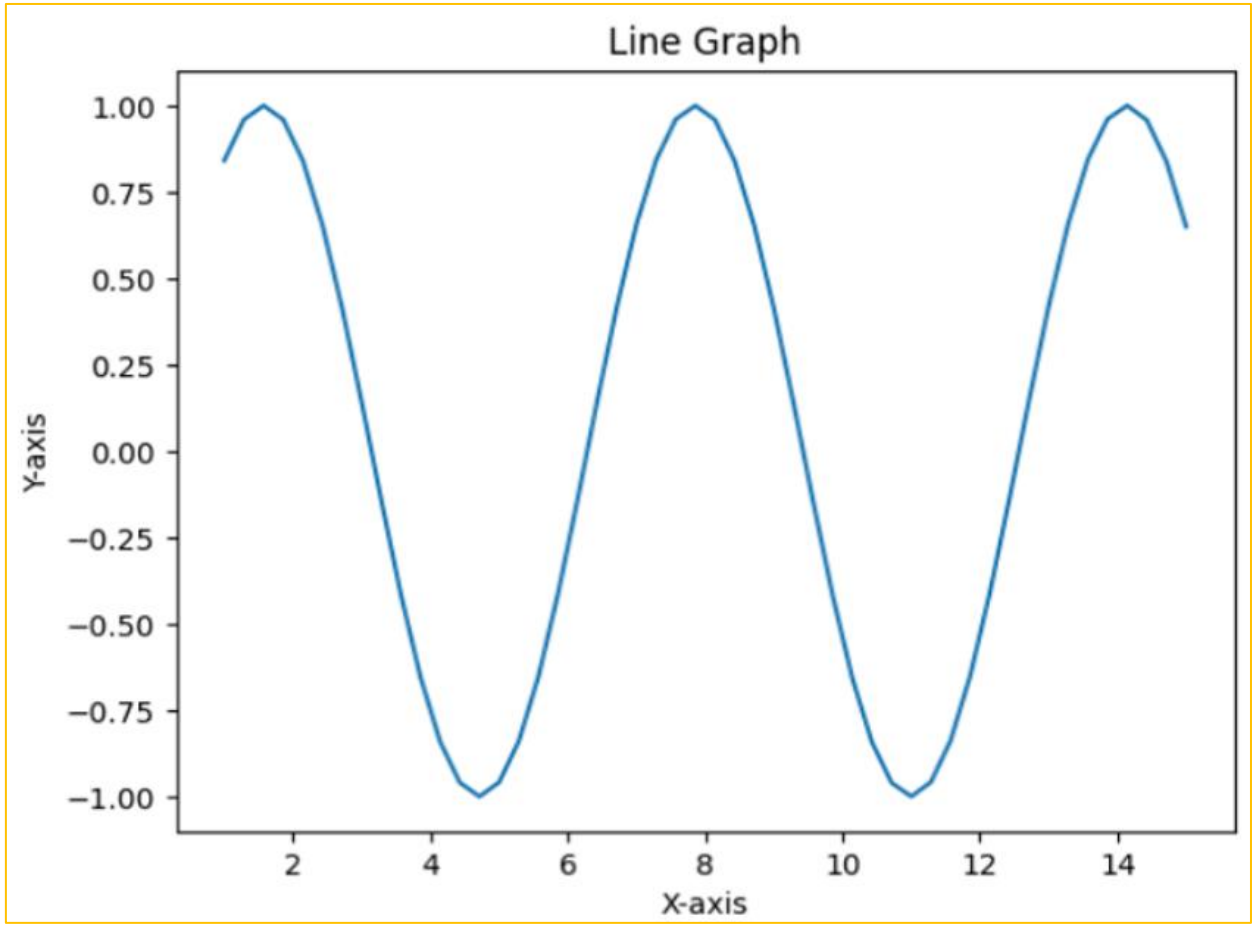
Examples

```
# importing the library
import numpy as np
import matplotlib.pyplot as plt
# data to be plotted
x = np.arange(1, 11)
y = np.array([100, 10, 300, 20, 500, 60, 700, 80, 900, 100])
# plotting
plt.title("Line graph")
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.plot(x, y, color ="green")
plt.show()
```



Examples

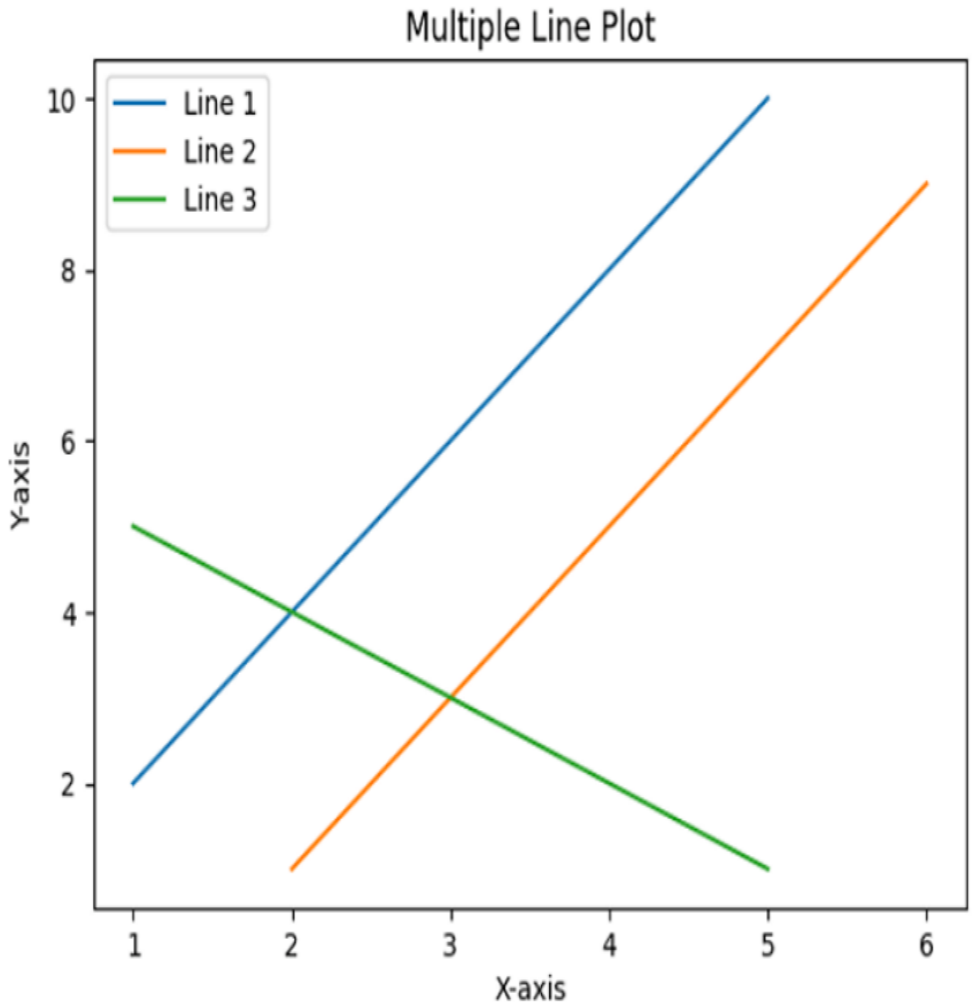
```
import numpy as np
import matplotlib.pyplot as plt
# To generate random data for plotting
axisX = np.linspace(1, 15, 50)
axisY = np.sin(axisX)
# To create the line graph from above data
plt.plot(axisX, axisY)
# Adding title and labels for the graph
plt.title('Line Graph')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
# to show the final graph
plt.show()
```



Examples

```
import numpy as np
import matplotlib.pyplot as plt
# Data points of line 1
x1 = np.array([1, 2, 3, 4, 5])
y1 = np.array([2, 4, 6, 8, 10])
# Data points of line 2
x2 = np.array([2, 3, 4, 5, 6])
y2 = np.array([1, 3, 5, 7, 9])
# Data points of line 3
x3 = np.array([1, 2, 3, 4, 5])
y3 = np.array([5, 4, 3, 2, 1])
```

```
# Plotting all lines with labels
plt.plot(x1, y1, label='Line 1')
plt.plot(x2, y2, label='Line 2')
plt.plot(x3, y3, label='Line 3')
# Adding legend, x and y labels,
and title for the lines
plt.legend()
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Multiple Line Plot')
# Displaying the plot
plt.show()
```



Examples

```
import matplotlib.pyplot as plt
import numpy as np
# Data points of line 1
x1 = np.array([1, 2, 3, 4, 5])
y1 = np.array([2, 4, 6, 8, 10])
# Data points of line 2
x2 = np.array([1, 2, 3, 4, 5])
y2 = np.array([1, 3, 5, 7, 9])
```

```
# Plotting all lines with specifying labels
plt.plot(x1, y1, label='Line 1')
plt.plot(x2, y2, label='Line 2')
# Adding legend, x and y labels, and titles for the lines
plt.legend()
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Plotting Multiple Lines in single graph')
# Displaying the plot
plt.show()
```

