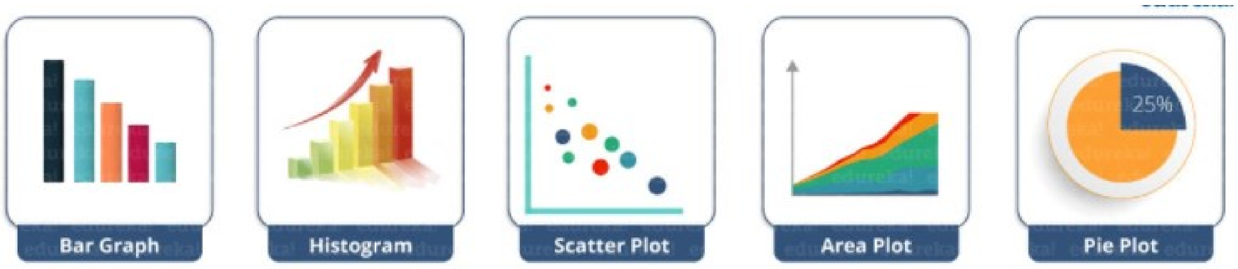
Help file for Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Introduction to Matplotlib:

- Environment Setup for Matplotlib
- Basic plots such as line, bar etc. There are various plots which can be created using python matplotlib. Some of them are listed below:



- General Concepts
- Examples and codes

Reading material:

You can refer to any of these links for step by step tutorial:

1. https://towardsdatascience.com/matplotlib-tutorial-learn-basics-of-pythons-powerful-plotting-library-b5d1b8f67596 (For the list of all the methods scroll to the end of the webpage)

2. https://www.edureka.co/blog/python-matplotlib-tutorial/

3. Python | Introduction to Matplotlib - GeeksforGeeks

For exercise refer:

4. Python | Introduction to Matplotlib - GeeksforGeeks

Introduction to Pyplot

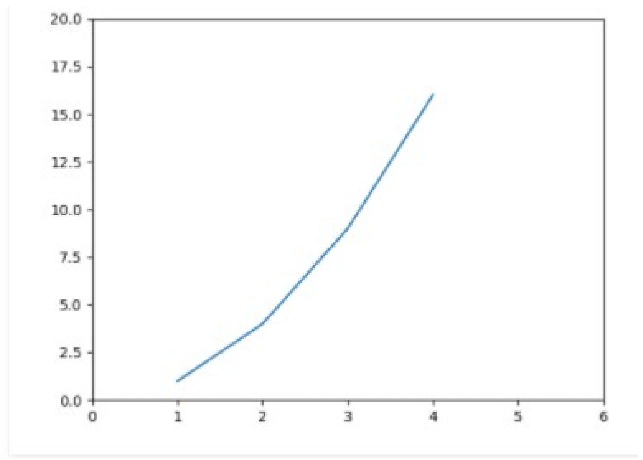
Pyplot is a Matplotlib module that provides a MATLAB like interface. Pyplot provides functions that interact with the figure i.e. creates a figure, decorates the plot with labels, creates plotting area in a figure.

Syntax:
*matplotlib.pyplot.plot(*args, scalex=True, scaley=True, data=None, **kwargs)*

Example:

```
# Python program to show pyplot module
   import matplotlib.pyplot as plt
   plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
   plt.axis([0, 6, 0, 20])
   plt.show()
```

Output:



The plot function marks the x-coordinates(1, 2, 3, 4) and y-coordinates(1, 4, 9, 16) in a linear graph with specified scales. [/caption]

Parameters: This function accepts parameters that enables us to set axes scales and format the graphs. These parameters are mentioned below :-

- **plot(x, y):** plot x and y using default line style and color.
- **plot.axis([xmin, xmax, ymin, ymax])**: scales the x-axis and y-axis from minimum to maximum values
- **plot.(x, y, color='green', marker='o', linestyle='dashed', linewidth=2, markersize=12):** x and y co-ordinates are marked using circular markers of size 12 and green color line with — style of width 2
- **plot.xlabel('X-axis')**: names x-axis
- **plot.ylabel('Y-axis')**: names y-axis
- **plot(x, y, label = 'Sample line ')** plotted Sample Line will be displayed as a legend

Summary of basic commands in Matplotlib:

1. Creating simple plots of sin(x) and cos(x)

```
import numpy as np
X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
```

```
C, S = np.cos(X), np.sin(X)
import matplotlib.pyplot as plt
plt.plot(X, C)
plt.plot(X, S)
plt.show()
```

Here, X is numpy array with 256 values ranging from -π to +π. C is cosine (256 values) and S is
sine (256 values).

2. Exploring all the figure settings that influence the appearance of the plot.

```
# Create a figure of size 8x6 inches, 80 dots per inch
plt.figure(figsize=(8, 6), dpi=80)
# Create a new subplot from a grid of 1x1
plt.subplot(1, 1, 1)
# Plot cosine with a blue continuous line of width 1 (pixels)
plt.plot(X, C, color="blue", linewidth=1.0, linestyle="-")
# Plot sine with a green continuous line of width 1 (pixels)
plt.plot(X, S, color="green", linewidth=1.0, linestyle="-")
# Set x limits
plt.xlim(-4.0, 4.0)
# Set x ticks
plt.xticks(np.linspace(-4, 4, 9, endpoint=True))
# Set y limits
plt.ylim(-1.0, 1.0)
# Set y ticks
plt.yticks(np.linspace(-1, 1, 5, endpoint=True))
# Save figure using 72 dots per inch
plt.savefig("exercise_2.png", dpi=72)
```

3. Adding legends

```
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosine")
plt.plot(X, S, color="red", linewidth=2.5, linestyle="-", label="sine")
plt.legend(loc='upper left')
```

4. Regular Plots

```
n = 256
X = np.linspace(-np.pi, np.pi, n, endpoint=True)
Y = np.sin(2 * X)
plt.plot(X, Y + 1, color='blue', alpha=1.00)
plt.plot(X, Y - 1, color='blue', alpha=1.00)
```

5. Scatter Plot

```
n = 1024
X = np.random.normal(0,1,n)
Y = np.random.normal(0,1,n)
```

```
plt.scatter(X,Y)
```

6. Bar Chart

```
n = 12
X = np.arange(n)
Y1 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)
plt.bar(X, +Y1, facecolor='#9999ff', edgecolor='white')
plt.show()
Y2 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)
plt.bar(X, -Y2, facecolor='#ff9999', edgecolor='white')
plt.show()
```