What is Matplotlib?

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.
Matplotlib was created by John D. Hunter.
Matplotlib is open source and we can use it freely.
Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compat
ibility.

Installation of Matplotlib ¶

If you have Python and PIP already installed on a system, then installation of Matplotlib is very easy.

Install it using this command:

In [ ]:
```
!pip install matplotlib
```

In [ ]:
```
import matplotlib as mp;  # Import
print(mp.__version__)   ## Check Version
```

Matplotlib Pyplot

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:

In [ ]:
```
import matplotlib.pyplot as plt    ## importing pyplot
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```

Plotting x and y points

The plot() function is used to draw points (markers) in a diagram.
By default, the plot() function draws a line from point to point.
The function takes parameters for specifying points in the diagram.
Parameter 1 is an array containing the points on the x-axis.
Parameter 2 is an array containing the points on the y-axis.
If we need to plot a line from (1, 3) to (8, 10), we have to pass two arrays [1, 8] and [3, 10] to the plot function.

In [ ]:
```
xpoints = np.array([1, 8])   # (x1, x2)   -- Horizontal Axis
ypoints = np.array([3, 10])  # (y1, y2)   -- Vertical Axis

plt.plot(xpoints, ypoints)
plt.show()
```

In [ ]:
```
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints, 'o')    # 'o' means rings, no line
plt.show()
```

In [ ]:
```
# Draw a line in a diagram from position (1, 3) to (2, 8) then to (6, 1) and finally to position (8, 10):
xpoints = np.array([1, 2, 6, 8])   # (x1, x2, x3, x4)
ypoints = np.array([3, 8, 1, 10])  # (y1, y2, y3, y4)

plt.plot(xpoints, ypoints)
plt.show()
```

Default X-Points

If we do not specify the points in the x-axis, they will get the default values 0, 1, 2, 3,
(etc. depending on the length of the y-points.

So, if we take the same example as above, and leave out the x-points, the diagram will look like this:

In [ ]:
```
ypoints = np.array([3, 8, 1, 10, 5, 7])

plt.plot(ypoints)     # The x-points in the example above is [0, 1, 2, 3, 4, 5].
plt.show()
```

Matplotlib Markers

Markers You can use the keyword argument marker to emphasize each point with a specified marker:

In [ ]:
```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = '*')
plt.show()
```

Format Strings fmt

You can use also use the shortcut string notation parameter to specify the marker.

This parameter is also called fmt, and is written with this syntax:

marker|line|color

In [ ]:
```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, 'o--g')  # Colors - r,b,g,c,m,y,w and
plt.show()                 # Pattern - :,--,-.
```

In [ ]:
```
plt.plot(ypoints, marker = 'o', ms = 20, mec = 'r', mfc ='g')   ## Marker Size(ms), Marker Color(mec)Markerface Color(mfc)
plt.show()
```

Matplotlib Line

Linestyle You can use the keyword argument linestyle, or shorter ls, to change the style of the plotted line:

In [ ]:
```
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linestyle = 'dotted', color = 'g', linewidth = '10.5')   # linestyle = 'dashed/dotted' or linestyle = ': / --'
plt.show()
```

In [ ]:
```
x1 = np.array([0, 1, 2, 3])    # Mulitple Line
y1 = np.array([3, 8, 1, 10])
x2 = np.array([0, 1, 2, 3])
y2 = np.array([6, 2, 7, 11])

plt.plot(x1, y1)
plt.plot(x2, y2)
plt.show()
```

Matplotlib Lables

Create Labels for a Plot

With Pyplot, you can use the xlabel() and ylabel() functions to set a label for the x- and y-axis.

In [ ]:
```
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.plot(x, y)

font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}

plt.title("Sports Watch Data", fontdict = font1)
#plt.title("Sports Watch Data", loc = 'right')   # Position the title to the "left/right"
plt.xlabel("Average Pulse", fontdict = font2)
plt.ylabel("Calorie Burnage", fontdict = font2)

plt.show();
```

Matplotlib Adding Grid Lines

Add Grid Lines to a Plot

With Pyplot, you can use the grid() function to add grid lines to the plot.

In [ ]:
```
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.plot(x, y)

plt.grid()

#plt.grid(axis = 'x')  # Only on axis = 'x or y'
#plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)   # Set Grid Properties

plt.show()
```

Matplotlib Subplot

Display Multiple Plots

With the subplot() function you can draw multiple plots in one figure:

In [ ]:
```
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1)    # the figure has 1 row, 2 columns, and this plot is the first plot.
plt.title("Sales")      # Sub Title
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2)    # the figure has 1 row, 2 columns, and this plot is the first plot.
plt.title("Income")     # Sub Title
plt.plot(x,y)

plt.suptitle("My Business")     # Super Title
plt.show()
```

Matplotlib Scatter

Creating Scatter Plots

With Pyplot, you can use the scatter() function to draw a scatter plot.

The scatter() function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:

In [ ]:
```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.xlabel("Car")
plt.ylabel("Speed")
plt.show()
```

In [ ]:
```
#day one, the age and speed of 13 cars:
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y)

#day two, the age and speed of 15 cars:
x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x, y)

# Comparision for 2 Days
plt.xlabel("Car")
plt.ylabel("Speed")
# plt.scatter(x, y, color = 'red')   # By default 'blue' and 'Orange'
plt.show()
```

Matplotlib Bars

Creating Bars

With Pyplot, you can use the bar() function to draw bar graphs

In [ ]:
```
x = np.array(["Apple", "Orange", "Banana", "Grapes"])
y = np.array([3, 8, 1, 10])

#plt.bar(x,y,width = 0.5)   # Vertical Bars along with width and height(only for Horizontal Bars)
plt.barh(x,y, height=0.3, color='green')  # Horizontal Bars, Colors, Width
plt.show()
```

Matplotlib Histograms

Histogram

A histogram is a graph showing frequency distributions.

It is a graph showing the number of observations within each given interval.

Example: Say you ask for the height of 250 people, you might end up with a histogram like this:

Create Histogram

In Matplotlib, we use the hist() function to create histograms.

The hist() function will use an array of numbers to create a histogram, the array is sent into the function as an argument.

For simplicity we use NumPy to randomly generate an array with 250 values, where the values will concentrate around 170, and the standard deviation is 10.

In [ ]:
```
import numpy as np

x = np.random.normal(170, 10, 250)    #(loc=0.0, scale=1.0, size=None)

print(x)

#plt.hist(x)
#plt.show()
```

Matplotlib Pie Charts

Creating Pie Charts

With Pyplot, you can use the pie() function to draw pie charts:

In [ ]:
```
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

#plt.pie(y)
#plt.pie(y, labels = mylabels)
#plt.pie(y, labels = mylabels, startangle = 90)
myexplode = [0.2, 0, 0, 0]
mycolors = ["black", "hotpink", "b", "#4CAF50"]
plt.pie(y, labels = mylabels, explode = myexplode, shadow = True, colors = mycolors)
#plt.legend(title = "Four Fruits:")
plt.show()
```