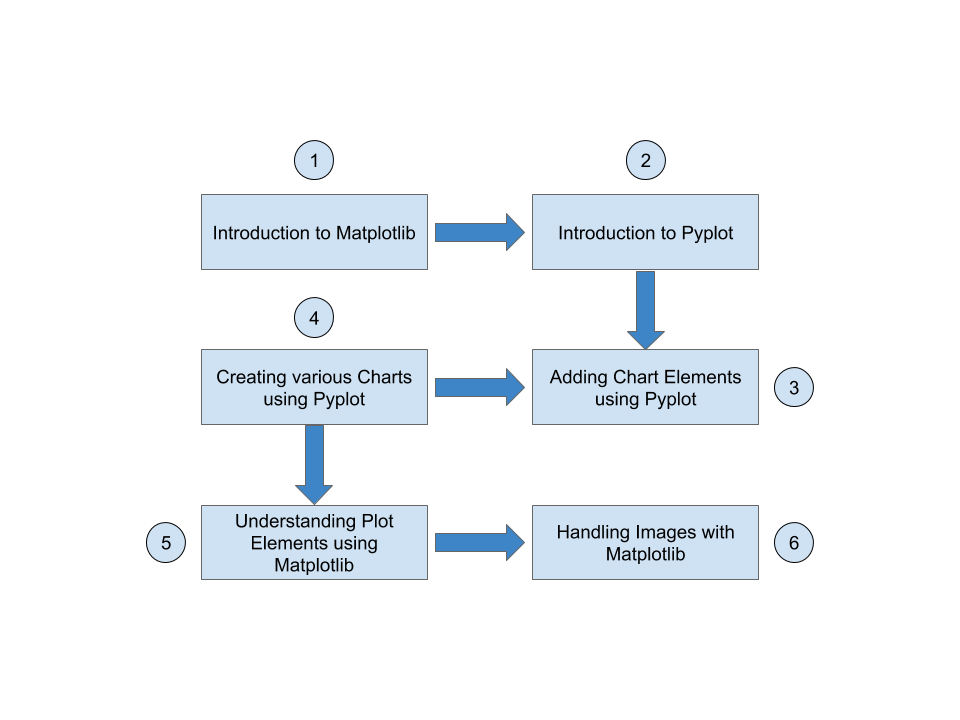
Getting Started with Matplotlib

Introduction to Matplotlib

* Matplotlib is one of the most popular Python packages used for **data visualization**.
* It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits.



* Several toolkits are available which extend Matplotlib functionality.

1. **Basemap:** This is a great tool for creating maps using python in a simple way.
2. **Cartopy:** It is a Python package designed for geospatial data processing in order to produce maps and other geospatial data analyses.
3. **Excel tools:** These are utilities for exchanging data with Microsoft Excel.
4. **GTK tools:** This is an interface to the GTK+ library
5. **Qt interface:** This is used for creating GUIs.
6. **Mplot3d:** Used for creating 3-D plots.
7. **Natgrid:** This is an interface to the natgrid library for gridding irregularly spaced data.
8. **matplotlib2tikz:** An export to Pgfplots for smooth integration into LaTeX documents.

We will look into all of this in a later while.

Let’s begin!

\*\*Go to Next Page \*\*

# Installation

These are the various methods using which you can install Matplotlib.

First, you need to check if you already have Matplotlib installed. To do that, you need to use the following command:

import matplotlib

If you don’t see any error messages, then Matplotlib is already installed on your system. If it is not installed, you can follow the below guidelines to install Matplotlib on your system.

* **Installing using Pip**: To install Matplotlib using Pip, simply use the below command.
* pip install --user matplotlib
* **Installing using Conda**: If you are using Anaconda, please use the command given below to install Matplotlib.
* conda install -c conda-forge matplotlib

If you get any error messages while installing, please refer to the respective user guides for Matplotlib installation. You can find them from the below links:

* Matplotlib: <https://pypi.org/project/matplotlib/>
* Conda Installation: <https://anaconda.org/conda-forge/matplotlib>

To check the Matplotlib version, we use the \_\_version\_\_ command. Please check the version of Matplotlib installed in the your PC:

import matplotlib

matplotlib.<< your code goes here >>

\*\*Go to Next Page \*\*

# Introduction to Pyplot

(Here we will learn more about Pyplot, and how it is associated to Matplotlib.)

* Pyplot is a module of Matplotlib which provides simple functions to add plot elements like lines, images, text, etc. to the current axes in the current figure.
* To try Pyplot, you need to import the pyplot module using the import function.
* Please import the pyplot module from Matplotlib:

import matplotlib.pyplot as plt

Creating a simple plot using Pyplot

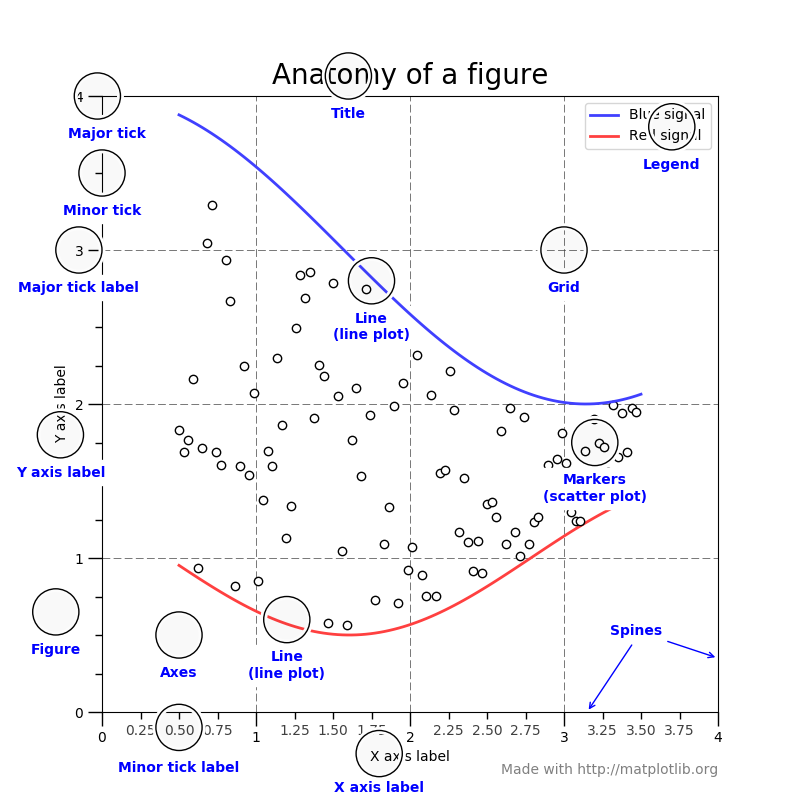
* First, we need to define 2 arrays X and Y of 5 values in each of them for the x- and y-coordinates respectively. To do this, we need to import Numpy as most of the data that we will be working with will be in the form of arrays only:
* import matplotlib
* import numpy as np
* Then we will declare the X and Y variables:
* X = np.array([1, 2, 3, 4, 5])
* Y = np.array([11, 12, 13, 14, 15])
* Now we will import Pyplot:
* import matplotlib.pyplot as plt
* Next, we will pass the X and Y arrays as input arguments to Pyplot’s plot method:
* plt.plot(X, Y)
* Finally, we will use the show method to display this plot we created:

plt.show()

\*\*Go to Next Page \*\*

# Adding Chart Elements using Pyplot

Every plot has some anatomy.



* We will start by adding a title and axis labels to our plot. To add the title, we use the title method:
* import numpy as np
* X = np.array([1,2,3,4,5])
* Y = np.array([11,12,13,14,15])
* import matplotlib.pyplot as plt
* plt.plot(X, Y)
* plt.title("Learning Pyplot")
* And, for the axis labels, we use the xlabel and ylabel methods respectively:
* plt.xlabel("X label")
* plt.ylabel("Y label")
* Finally, we use show to display our plot:

plt.show()

\*\*Go to Next Page \*\*

# Creating Charts using Pyplots

Now we will learn how to create bar graphs in Matplotlib using Pyplot.

Bar graphs are one of the most common types of graphs.

The goal is to show the relationship between the two axes.

They can also be used to track changes over time.

Pyplot provides a method bar to make bar graphs which take arguments: categorical variables, their values and color (if you want to specify any).

* First, we will import Pyplot:
* import matplotlib.pyplot as plt
* Let us first create 2 arrays; subjects, and their respective marks:
* subjects = ["Maths", "Biology", "Chemistry", "Physics", "English", "Computers"]
* marks = [97, 68, 59, 81, 77, 92]
* Now let us describe the plot as a bar graph:
* plt.bar(subjects, marks, color='blue')
* #plt.barh(marks, subjects, color='red')
* Next, we will define the title, and axis labels:
* plt.title("Bar Graph Example")
* plt.xlabel("Subjects")
* plt.ylabel("Marks")
* And finally, we will show the plot:

plt.show()

# We can create horizontal bar graphs using the barh method instead of the bar method.

\*\*Go to Next Page \*\*

# Creating Clustered Bar Graphs

Now we will create a Clustered bar graph.

The clustered bar graph extends the standard bar graph from looking at numeric values across one categorical variable to two or more categorical variables.

Each bar in a standard bar graph is divided into a number of sub-bars stacked side to side, each one corresponding to a level of the subsequent categorical variables.

* Let's start by importing Pyplot, and Numpy:
* import matplotplib.pyplot as plt
* import numpy as np
* Now let us define the subjects, and the marks for the 3 students:
* subjects = ["Maths", "Biology", "Chemistry", "Physics", "English", "Computers"]
* student1 = [97, 68, 59, 81, 77, 92]
* student2 = [88, 61, 80, 40, 62, 52]
* student3 = [54, 62, 77, 54, 71, 98]
* Next we will define an array called index, and a variable called width. Here we mention the index and width of our bar graphs in order to horizontally stack them together.
* index = np.arange(6)
* width = 0.03
* **Note:** Please write all the following codes till the end in a single cell. Now let's plot the bars. We will use the same bar method in this case, but a bit differently:
* Plt.bar(index, student1, width, color="aqua", label="Student 1")
* plt.bar(index + width, student2, width, color="green", label="Student 2")
* plt.bar(index + (width\*2), student3, width, color="blue", label="Student 3")
* Now let us set the title, and axis labels:
* plt.title("Stacked Bar Graph Example")
* plt.xlabel("Students")
* plt.ylabel("Marks")
* Next, we will use the xticks method to label our x-axis based on the position of our bars.
* plt.xticks(index + width/2, subjects)
* And we will plot the legends:
* plt.legend()
* Finally, we will show our plot:

plt.show

\*\*Go to Next Page \*\*

# Creating Pie Charts using Pyplot

Pie charts are generally used to show percentage or proportional data and usually the percentage represented by each category is provided next to the corresponding slice of pie. Graphs such as pie charts and bar graphs show descriptive data, or qualitative data.

We will now create a pie chart to show the market share of 5 different firms.

* First, we will import Pyplot:
* import matplotlib.plt as plt
* Now, let us define the data:
* firms = ["Firm A", "Firm B", "Firm C", "Firm D", "Firm E"]
* market\_shares = [10, 40, 30, 5, 15]
* Next, we will define an array called Explode. Please note that this is an optional step for creating pie charts. However, if you want to focus on one particular firm, you can explode a part of your pie chart. Here, we will focus on Firm D which has the lowest market shares. So, let us define Explode now:
* Explode = [0, 0, 0, 0.5, 0]

Please notice that in the above array, we have kept a non-zero value for the 3rd value, this index corresponds to Firm D.

* Like we did in the last assessment, you need to write all the following codes in a single cell. Now we will create the pie chart using the pie method:
* plt.pie(market\_shares, explode=Explode, labels=firms, shadow=True, startangle=45)

Here startangle=45 means that everything is rotated counter-clockwise by 45 degrees.

* Next, we will define the axis and legends:
* plt.axis('equal')
* plt.legend(title = "List of Firms")

In the above code, 'equal' denotes equal aspect ratio to ensures that pie is drawn as a circle. Also, the title in legend is the title of the legends and not the entire chart. Finally, we will show the chart:

plt.show()

# Creating Histograms using Pyplot

Histograms are a special form of bar chart where the data represent continuous rather than discrete categories.

This means that in a histogram there are no gaps between the columns representing the different categories.

Histogram’s data is plotted within a range against its frequency.

Histograms are very commonly occurring graphs in probability and statistics and form the basis for various distributions like the normal -distribution, t-distribution, etc.

* Let us start by importing Pyplot:
* import matplotlib.pyplo as plt
* Now let us create a variable x with the data that we will use to plot the histogram. Here we will use NumPy’s random.randn() method which generates data with the properties of a standard normal distribution i.e. mean = 0 and standard deviation = 1, and hence the histogram looks like a normal distribution curve.
* Import numpy as np
* x = np.random.randn(10000)
* Next, we will set up the title and the axis labels:
* plt.title("Histogram Example")
* plt.xlabel ("Random Data")
* plt.ylabel ("Frequency")
* Now we will plot our histogram using the hist method:
* plt.hist(x, 10, rwidth = 0.7)

We have used 10 here to divide the data into 10 equal strata or bins. We add ‘rwidth’ to add space between two bars.

Note that this value should be less than 1.

* And finally, we will show our plot:

plt.show()

\*\*Go to Next Page \*\*

# Creating 3D Projections

1. Import Axes3D:
2. from mpl\_toolkits.mplot3d import Axes3D
3. Create the projection:
4. x = np.linspace(-5, 5, 50)
5. y = np.linspace(-5, 5, 50)
6. X, Y = np.meshgrid(x, y)
7. R = np.sqrt(X\*\*2 + Y\*\*2)
8. Z = np.sin(R)

Here, linspace returns evenly spaced numbers over a specified interval. meshgrid returns coordinate matrices from coordinate vectors.

1. Finally, we will create the figure, the subplot, and call the plot\_surface with the coordinates we created earlier:
2. figure = plt.figure(1, figsize = (12, 4))
3. subplot3d = plt.subplot(111, projection='3d')
4. surface = subplot3d.plot\_surface(X, Y, Z, rstride=1, cstride=1, cmap=matplotlib.cm.coolwarm, linewidth=0.1)
5. plt.show()

The figure module provides the top-level Artist , the Figure , which contains all the plot elements. The figsize marks the width and height of the plot.

Axes3D: <https://matplotlib.org/stable/api/_as_gen/mpl_toolkits.mplot3d.axes3d.Axes3D.html>

\*\*Go to Next Page \*\*

# Creating Scatterplot using Pyplot

Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation.

* Import pyplot
* import matplotlib.pyplot as plt
* Import rand to generate random values for the x-, and y-coordinates
* from numpy.random import rand
* Now we will call the scatter module to generate the scatter plot.
* for color in ['red', 'green', 'blue']:
* n = 100
* x, y = rand(2, n)
* scale = 450.0 \* rand(n) \*\* 5
* plt.scatter(x, y, s=scale, c=color, alpha=0.3, edgecolors='blue')
* plt.grid(True)
* plt.show()

Here, s is a scalar or array-like shape, and is an option argument. c defines the color. alpha defines the alpha blending value, or the transparency. edgecolor is the line color of the shapes. grid displays the plot grid.

pyplot.scatter: <https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.scatter.html>

\*\*Go to Next Page \*\*

# Understanding Plot Elements

At times, we might want to display more than one chart at a time. We can achieve this by using subplots.

* First, import Pyplot
* import matplotlib.pyplot as plt
* Now we will create a sample plot
* plt.plot([1,4,7])

We did this to demonstrate that creating a subplot will delete any pre-existing subplot that overlaps with it beyond sharing a boundary.

* Now, in the same cell as before, we will create a subplot using the subplot() function with top plot of a grid with 2 rows and 1 column. Since this subplot will overlap the first, the plot (and its axes) previously created, will be removed
* plt.subplot(2,1,1)
* plt.plot(range(12))
* plt.subplot(2,1,2, facecolor='red')
* plt.plot(range(12))
* plt.show()

The third line creates a second subplot with red background.

* You can add an insert plot in the same figure by adding another axes object in the same figure canvas. The matplotlib.figure module contains the Figure class. It is a top-level container for all plot elements. The Figure object is instantiated by calling the figure() function of the pyplot module. You can create it as follows:
* import matplotlib.pyplot as plt
* import numpy as np
* import math
* x = np.arange(0, math.pi\*2, 0.09)
* fig=plt.figure()
* axes1 = fig.add\_axes([0.1, 0.1, 0.9, 0.9])
* axes2 = fig.add\_axes([0.62, 0.62, 0.3, 0.3])
* y = np.sin(x)
* axes1.plot(x, y, 'b')
* axes2.plot(x,np.cos(x),'r')
* axes1.set\_title('sine')
* axes2.set\_title("cosine")
* plt.show()

Here, axes1 is the main axes, and axes2 is the inset axes. You can test with different values to move and resize the charts.

\*\*Go to Next Page \*\*

# Working with Images Matplotlib

* The image module in Matplotlib package provides various functions required for loading, rescaling and displaying image.
* Loading image data is supported by the Pillow library. Natively, Matplotlib only supports PNG images.
* First, we will import pyplot and image from matplotlib. We will also import numpy
* import matplotlib.pyplot as plt
* import matplotlib.pyplot as mpimg
* import numpy as np
* Now we will read the image using the imread function
* path = 'C:/Users/Ajinkya/Documents/Python Projects/Getting Started with Matplotlib/what.png'
* img = mpimg.imread(path)
* And finally, we will display the image using imshow function

imgplot = plt.imshow(img)

mpimg: <https://matplotlib.org/stable/tutorials/introductory/images.html>

Conclusion: -

Congratulations! You have gone through Matplotlib tutorial successfully! There is still much to learn, but you’re definitely ready to go out on your own and create your own amazing plots.

If you have followed along since the beginning, then you have learned most of what Matplotlib has to offer. You might not believe it, but there's still a lot of other things Matplotlib can do! Moving forward, you can always head to [Matplotlib.org](http://matplotlib.org/), and check out the examples and the gallery section.

~JinlyaEinstien

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*THE END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*