Matplotlib

`import matplotlob.pyplot as plt`

`plt.savefig('plot.png or some other')` Saving our plot

`plt.title('something')`

`plt.xlabel('something')`

`plt.ylabel('something')`

`plt.ylabel('Price', fontsize = 16, fontweight = 'bold')`

`plt.grid(True)` to make the plots have a grid

`plt.style.use('fivethirtyeight')` we have so many styles to chose from, you can choose your own by looking a `plt.style.available`, If we are defining a style here, we don't really need to specify colors and other features as below

`plt.plot(x1, y1, label = 'label1' , color = 'b', linestyle = '--', linewidth = 3)` 1st lineplot, we can give our line our customized features as shown

`plt.plot(x1, y2, label = 'label2' )` 2nd lineplot

`plt.legend()` if we are plotting more than one line, it's important to know which one is which

`plt.show()`

BARCHARTS

`plt.bar(x1, y1 , we can give our own attributes like above)` we can also plot lineplot of other data with this plot simply by writing plt.plot `plt.barh(x1, y1)` for horizontal bar chart, to print something like popularity of a player with player name in y and no. of followers in x

to plot multiple bar charts side by side we not to use offsetting otherwise they will be stacked on each other

`plt.xticks( ticks = x.index, labels = ages)` It will give our x axis values contained in ages list

Pie Charts

`plt.pie(column_name,` `autopct = '%1.1f%%'` `, labels = give a list of labels , explode = a list specifying how much apart from radius we want our slice to be)`

Stackplots(similar as piecharts)

`plt.stackplot(x1, y1, y2, y3, labels = list of labels`

`)`

Histograms

`bins = [10, 20, 30, 40, 50, 60 ]`

`plt.hist(ages ,` `bins = 5` `, edgecolor = 'black')` , it will divide the data into 5 bins

`plt.hist(ages , bins = bins, edgecolor = 'black')`, if we pass on above list of bins there will be 6 bins created with class interval as list elements, we can even exclude some data by specifying our class interval as needed.

if some of datapoints are too big that is making those very small on our plot , we can use log on our data

`plt.hist(ages , bins = 5, edgecolor = 'black', log = True)`

Scatterplots

`plt.scatter(x, y or only x)`

`plt.scatter(x, y or only x, c = just like hue put a column here,  )`

`cbar = plt.colorbar()`

`cbar.set_label('give a label on which we plotted hue')`

`plt.xscale('log')`: to change the scale of plot

`plt.yscale('log')`

Time series data

`plt.style.use('seaborn')`

`plt.plot_date(` `dates` `, y, linestyle = 'solid')`

`plt.gcf.autofmt_xdate()` makes dates on x axis readable

Subplots

`fig, (ax1, ax2) = plt.subplots(nrows = 2, ncols = 1,` `sharex= True` `)` , we can changes no. of rows and columns in our plot axes

`ax1.plot(x, y1, label = ''something')`, everything will be same as we did earlier

`ax2.plot(x, y2)`

`ax1.set_title()`

`ax1.xlabel()`

`ax1.ylabel()`

`ax12.ylabel()`

`ax.legend()`

TO get 2 figures we just need to separate

```
fig, ax1 = plt.subplots()
fig, ax2 = plt.subplots()
```

Plotting Live data

```
from matplotlib.animation import FuncAnimation


def animate(i):
    data  = pd.read_csv('data.csv')
    x = data['x_val']
    y1 = data['t1']
    y2 = data['t2']

    plt.cla()
    plt.plot(x, y1, label = 'Channel 1')
    plt.plot(x, y2, label = 'Channel 2')
    plt.legend(loc = 'upper left')
    plt.tight_layout()


ani = FuncAnimation(plt.gcf(), animate, interval = 1000)
plt.tight_layout()
plt.show()
```