Matplotlib Course Notes

You can install with using “pip install matplotlib”
You need the import pyplot module from matplotlib with using: from matplotlib import pyplot as plt

When you want to add data to plot you use: plt.plot() method and if you want to show your plot you have to use:plt.show() method

We need to have some labels to show what data actually represents
to give our entire plot a title : plt.title(‘name’)
to give x label a title: plt.xlabel(‘name’)
to give x label a title: plt.ylabel(‘name’)

To add a new line to the plot: we can simply create a new list.

You can create an new list for your plot but if you have common data in your plot you can pass your data to another graphic in plot.To specify to which data is you can use: plt.legend([‘firstname, ’secondname’])

We can pass a label argument to our plot methods, you can pass in a label argument and say label=’name’)
A format strings consists of a part for color,mark,line:
fmt=’[‘marker][line][color]’

example: plt.plot(ages,ages2,’k–’ ,label=”name”) its black and has dots in the line
also you can use like: plt.plot(ages,ages2, color=’k’, linestyle=’--’ ,label=”name”)
to add markers,you can check markerstyle and use like: marker=’.’(it has dots)
you can also use hex values for color
To make a line little bit wider : linewidth=3
to automatically padding : plt.python_layout()
To put in a grid:plt.grid(True)
These are the styles:

```
['seaborn-dark', 'seaborn-darkgrid', 'seaborn-ticks', 'fivethirtyeight', 'seaborn-whitegrid',
'classic', '_classic_test', 'fast', 'seaborn-talk', 'seaborn-dark-palette', 'seaborn-bright',
'seaborn-pastel', 'grayscale', 'seaborn-notebook', 'ggplot', 'seaborn-colorblind',
'seaborn-muted', 'seaborn', 'Solarize_Light2', 'seaborn-paper', 'bmh',
'tableau-colorblind10', 'seaborn-white', 'dark_background', 'seaborn-poster', 'seaborn-deep']
```

To style your to to ,

to use these styles:plt.style.use()
another styling method is : plt.xkcd() (it is like comics style graph)
To save as PNG in your directory :plt.savefig(‘plot.png’,path)

Bar Charts and Analyzing Data from CSVs

We can use the bar method instead of the plot method. With using that you can pass x and y values directly.And it will represent as a bar chart instead.If you use plot method at the same time it will overlay above the bar chart.

It is complicated to read data to avoid this we can use the numpy arange method.
If you want to share your data in the bar.
1. x_indexes=np.arange(len(ages_x))
2. Add width value: width=0.25
3. add bar parenthesis :x_indexes-width to first bar and for the second x_indexes and for the last one x_values+width

legend command used for the show your values in the bar chart.
And use this method to show the x values : plt.xticks(ticks=x_indexes,labels=ages_x)

To add data to bar chart from CSV file:

```
import csv
import numpy as np
from matplotlib import pyplot as plt

plt.style.use("fivethirtyeight")

with open('data.csv') as csv_file:
    csv_reader = csv.DictReader(csv_file)

    row = next(csv_reader)
```

To see the first row of the csv file .

print(row['LanguagesWorkedWith']) it will print the first column of LanguagesWorkedWith.
To see like the list you can use print(row['LanguagesWorkedWith'].split(' ; '))
How do counters actually work?
Open your terminal and write this: from collections import Counter
It will count to specific word you want in that list
print(language_counter.mostcommon(15)) it will print the most common words in that csv file like tuple.

```
for item in language_counter.most_common(15):
    languages.append(item[0])
    popularity.append(item[1])
```

```
'JavaScript', 'HTML/CSS', 'SQL', 'Python', 'Java', 'Bash/Shell/PowerShell', 'C#', 'PHP',
C++', 'TypeScript', 'C', 'Other(s):', 'Ruby', 'Go', 'Assembly']
59219, 55466, 47544, 30715, 29931, 31991, 27097, 23030, 20524, 18523, 18017, 7920, 7331,
201, 5833]
```

we can actually use these two lists for our plot

It will split the csv files using parameters.

You can use plt.bar(languages,popularity) to show in the bar chart.
plt.barh(y ax) : it will show like horizontal
In this example :plt.barh(languages, popularity)
For automatic padding:plt.tight_layout)

Instead of using the DictReader method to read data. You can use pandas method:
data=pd.read_csv('yourdocname.csv')
ids=data['Responder_id']
lang_resposes=data['LanguagesWorkedWith']

and :
for response in lang_responses:
language_counter.update(response.split(' ; '))

PİE CHARTS

```
slices = [60, 40]

plt.pie(slices)

plt.title("My Awesome
plt.tight_layout()
plt.show()

# Colors:
# Blue = #008fd5
```

Example of creating pie chart

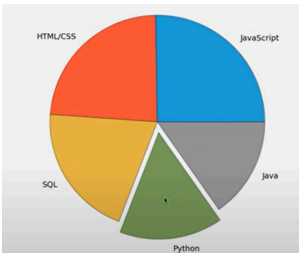
To label these colors: labels=[‘firstvalue’,’second value’]
plt.pie(slices,labels= labels)

to change the color: plt.pie(slices,labels=labels,wedgeprops={‘edgecolor’:’black’})
if you want to know more about wedge properties you can check their website

Change the color of slices: colors=[‘blue’, ‘red’,etc…] and you need to add in the pie method like colors=colors

You can also use hex colors

To explode an element :
1.create a list explode=[0,0,0.1,0]
2.Add to the pie method



We can add a shadow our pie: Add to your pie (shadow=True)
To rotate the pie: add startangle=90 (like this)
To add the percentage that slice, use that argument: `autopct='%1.1f%%'`

STACK PLOTS

-To use: plt.stackplot(minutes,player1,player2,player3)

```
plt.legend(loc='upper left')
```

It will give you clarity.

To change labels color: create color list and add to the arguments in stackplot
matplotlib.pyplot.legend(*args, **kwargs)
[source]

Place a legend on the Axe

FILLING AREA on LINE SPOTS

-

```
data = pd.read_csv('data.csv')
ages = data['Age']
dev_salaries = data['All_Devs']
py_salaries = data['Python']
js_salaries = data['JavaScript']
```

To pull the data from csv file

plt.fill_between(x,y1,y2=0,where=None,interpole=False)

xarray (length N)

The x coordinates of the nodes defining the curves.

y1array (length N) or scalar

The y coordinates of the nodes defining the first curve.

y2array (length N) or scalar, default: 0

The y coordinates of the nodes defining the second curve.

overall_median=yourvalue(int) it will start filling with this value.
Where argument provides a condition to fill.
interpolate=True: that will make sure that certain x ‘s dont clipped.
Also you can change condition using where argument

Histograms
- plt.hist(parameter,bins=5)
bins will apart parameters to 5 pieces
to see bins clearly,you can add edgecolor=’color name’
Bins lists are used to categorize data by the condition.
matplotlib.pyplot.hist(x, bins=None, range=None, density=False, weights=None, cumulative=False, bottom=None, histtype='bar', align='mid', orientation='vertical', rwidth=None, log=False, color=None, label=None, stacked=False, *, data=None, **kwargs)

If you use log=True:it will scale the data by logarithmic

plt.axvline(median_age,color=color,label=Age median)

matplotlib.pyplot.axvline(x=0, ymin=0, ymax=1, **kwargs

Add a vertical line across the Axes.

Parameters:

xfloat, default: 0

x position in data coordinates of the vertical line.

yminfloat, default: 0

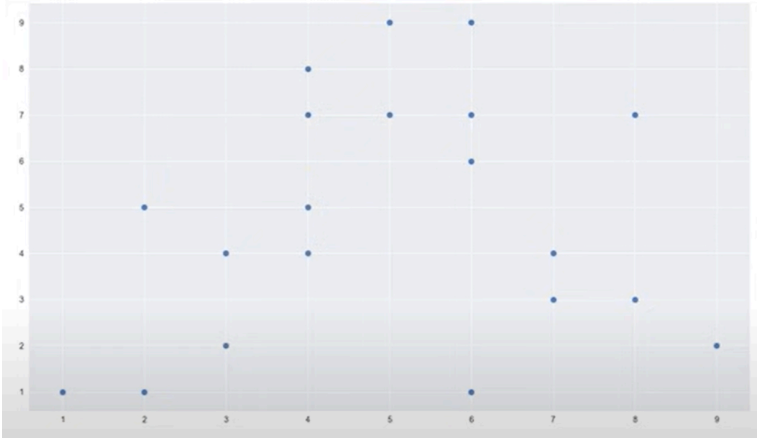
Should be between 0 and 1, 0 being the bottom of the plot, 1 the top of the plot.

ymaxfloat, default: 1

Should be between 0 and 1, 0 being the bottom of the plot, 1 the top of the plot.

SCATTER PLOT(Serpilme Diyagramı)

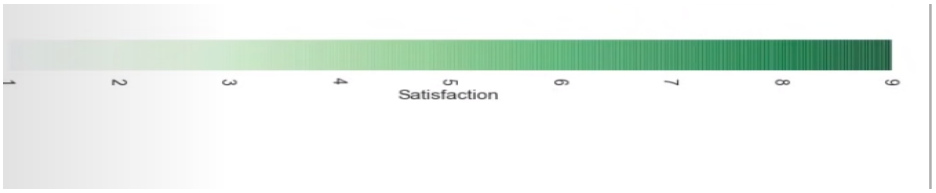
plt.scatter(x,y)



matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, *, edgecolors=None, plotnonfinite=False, data=None, **kwargs)

c color also you can assign to colors list
marker : shape
cmap= to decide to shade of which color is it

cbar=plt.colorbar()
cbar.set.label('your labelname')
it will add scale
s stands for size(circle size)



to scale data with using log scale:plt.xscale('log')

Axes.set_xscale(value, **kwargs)

[source]

Set the xaxis' scale.

Parameters:

value{"linear", "log", "symlog", "logit", ...} or ScaleBase

The axis scale type to apply.

**kwargs

Different keyword arguments are accepted, depending on the scale. See the respective class keyword arguments:

- matplotlib.scale.LinearScale
- matplotlib.scale.LogScale
- matplotlib.scale.SymmetricalLogScale
- matplotlib.scale.LogitScale
- matplotlib.scale.FuncScale

PLOTTING TIME SERIES DATA

-Before using this import these:

```
import pandas as pd
from datetime import datetime, timedelta
from matplotlib import pyplot as plt
from matplotlib import dates as mpl_dates
```

matplotlib.pyplot.plot_date#

matplotlib.pyplot.plot_date(x, y, fmt='o', tz=None, xdate=True, ydate=False, *, data=None, **kwargs)

you can change linestyle with using:linestyle='solid' with adding plot_date method

matplotlib.pyplot.gcf

matplotlib.pyplot.gcf()

[source]

Get the current figure

To display the date format: date_format=mpl_dates.DateFormatter('%b,%d,%Y')

Also use for csv files: data['Date']=pd.to_datetime(data['Date']) it is converting to datetime using panda

data.sort_values('Date',inplace=True)

PLOTTING LIVE DATA in REAL TIME

-Don't forget the import these:

```
import random
from itertools import count
import pandas as pd
import matplotlib.pyplot as plt
```

We are going to use this function:

```
def animate(i):
    x_vals.append(next(index))
    y_vals.append(random.randint(0, 5))
```

We can track our data with using animation module in matplotlib library :

```
from matplotlib.animation import FuncAnimation
```

ani=FuncAnimation(plt.gcf(),animate,interval=secondtime(int))

It is not clearing out the old time lines to solve this clear out axis

use this:plt.cla()

matplotlib.pyplot.cla()
[source]
Clear the current axes.

matplotlib.animation.FuncAnimation

class matplotlib.animation.FuncAnimation(fig, func, frames=None, init_func=None, fargs=None, save_count=None, *, cache_frame_data=True, **kwargs)

First change animate function like this

```
def animate(i):
    data = pd.read_csv('data.csv')
    x = data['x_value']
    y1 = data['total_1']
    y2 = data['total_2']
```

And you are going to use data from the csv file(it will create random data for your code)

SUBPLOTS

-Figure is the container holding our plots
-Axes are the actual plots
-plt.gcf():get the current figure
-plt.gca()
-Subplots create a figure and specify a certain number of rows

matplotlib.pyplot.subplots

matplotlib.pyplot.subplots(nrows=1, ncols=1, *, sharex=False, sharey=False, squeeze=True, width_ratios=None, height_ratios=None, subplot_kw=None, gridspec_kw=None, **fig_kw)
[source]
Create a figure and a set of subplots.
Share x will only label the bottom ticks for two rows
share y will only label the leftmost ticks

ax.set_xlabel()

Axes.set_xlabel(xlabel, fontdict=None, labelpad=None, *, loc=None, **kwargs)
[source]
Set the label for the x-axis.

If we want to 2 figures at the same time ⇒
fig1,ax1=plt.subplots()
fig2,ax2=plt.subplots()
Also you can save the figures as png:fig1.save('fig1.png')

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