Data visualization with Matplotlib

Matplotlib is a *multiplatform* visualization library built on NumPy arrays. More modern packages and tools have been developed on top of it (e.g. Seaborn, ggplot), but it is still useful to know Matplotlib to be able to adjust finer details.

```
import matplotlib as mlp
import matplotlib.pyplot as plt
```

in an interactive shell:

```
>>> %matplotlib
Using matplotlib backend: Qt5Agg
>>> import matplotlib.pyplot as plt
```

Matplotlib has 2 interfaces: object-oriented and MATLAB-style (function-based). We will be using the function-based interface here. Function reference:

https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html#module-matplotlib.pyplot

DISPLAYING THE PLOT

- from a **python script**:

`plt.show()` –
  - should be used once per python session, most often, **in the end of the script**
  - starts an event loop
  - looks for all active figure objects
  - opens one or more interactive windows displaying the figure(s)

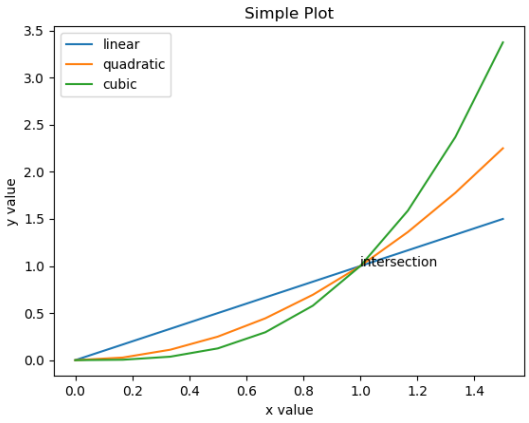
- from an **interactive shell**:

`plt.plot()` –
  - causes a figure window to open
  - further commands can be run to update the plot (some, automatically, some, requiring a `plt.draw()`

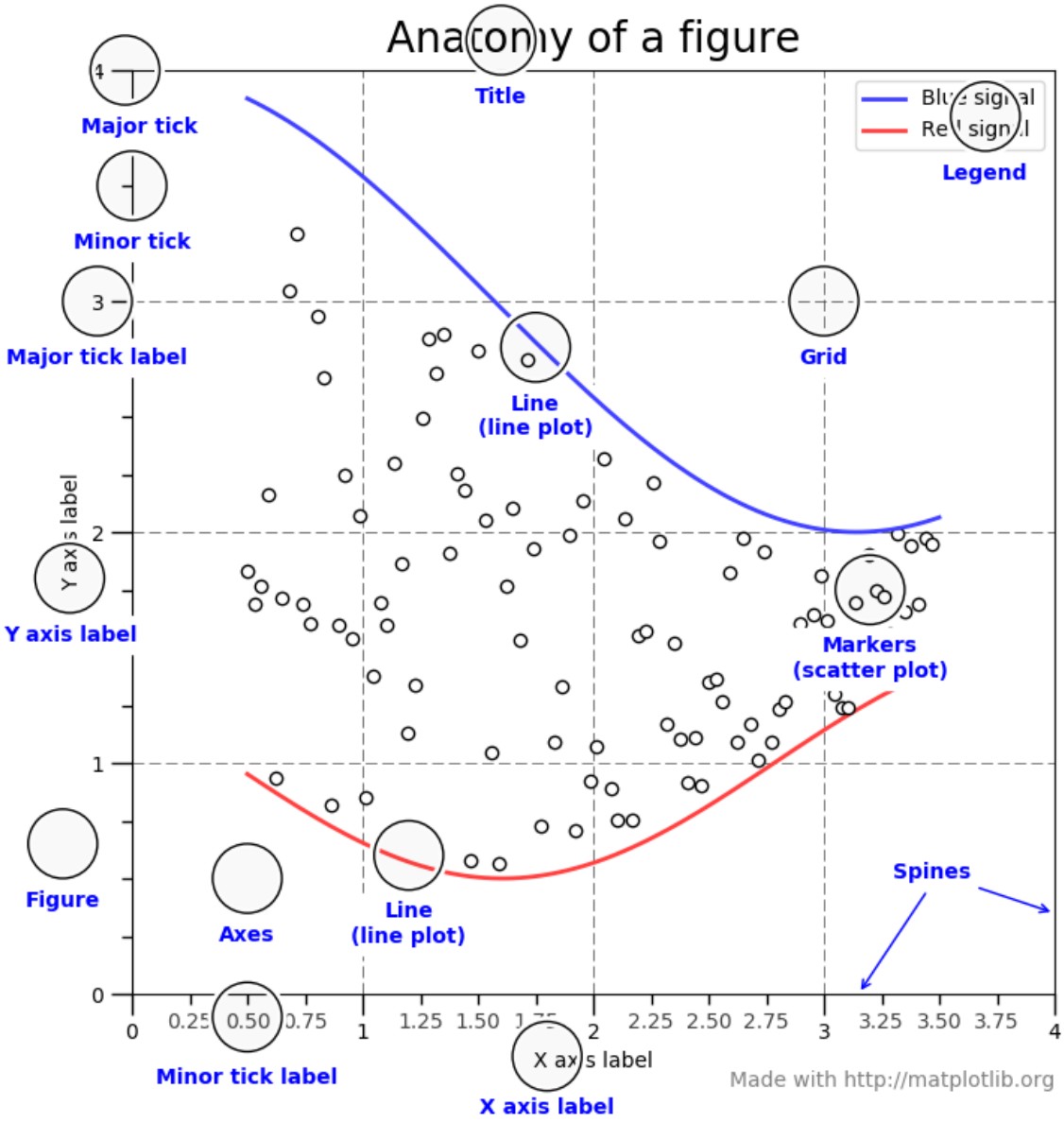
Example: assuming the required imports are made

```
x = np.linspace(0, 1.5, 10)
plt.plot(x, x, label='linear')
plt.plot(x, x**2, label='quadratic')
plt.plot(x, x**3, label='cubic')

plt.annotate('intersection', (1,1))

plt.xlabel('x value')
plt.ylabel('y value')
plt.title("Simple Plot")
plt.legend()
# plt.show()
```

From



https://matplotlib.org/tutorials/introductory/usage.html#sphx-glr-tutorials-introductory-usage-py :



CONTROLLING PARAMETERS OF A PLOT

The following parameters can be specified as a part of the argument list of the plot function:

- `color='blue', color = 'g'`
- `linestyle = 'solid' # or '-', 'dashed', '–' 'dashdot', '-.', 'dotted', ':'`
- the above two may be combined in the single third parameter

```
plt.plot(x, x**3, '--r')  # -- is line type, r  is red
```

- scatter plots – instead of lines, can display just the points. Marker parameter indicates the point type: o,x,.o+,>

```
plt.plot(x, x, 'rx', label='linear')  # x – point marker,r - re
```

- adjust axis limits

```
plt.xlim(-5,10);        plt.ylim(-5,10)
```

- include a grid

```
plt.grid(True)
```

- set the ticks

```
plt.yticks(range(20, 100, 10) )
def formatXYBetter():
    current_values = plt.gca().get_xticks()
    plt.gca().set_xticklabels(['{:,.0f}'.format(x) for x in current_values])
    current_values = plt.gca().get_yticks()
    plt.gca().set_yticklabels(['{:,.0f}'.format(x) for x in current_values])
```

- get the ticks, change tick label display angle

```
locs, labels = plt.xticks()
plt.setp(labels, rotation=45) # rotate the tick labels
```

- adjust the grid parameters

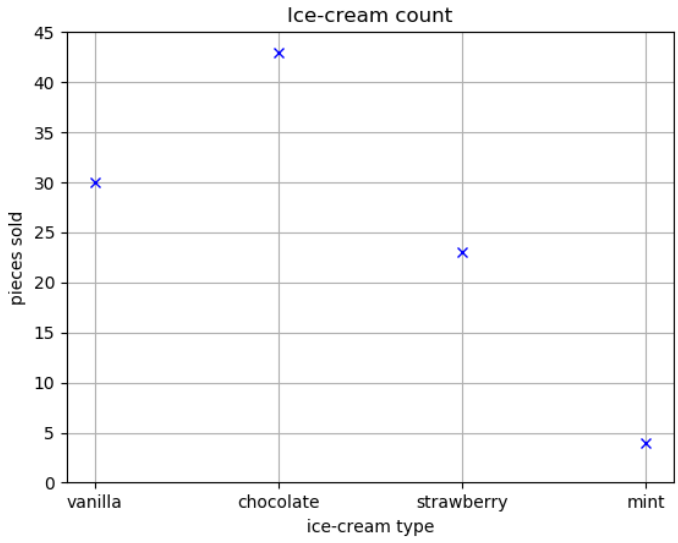
```
plt.grid(color = 'lightgrey', linestyle = '--', linewidth = .5)
```

- adjust the style

```
plt.style.use('ggplot') # other styles available as plt.style.available
```

Practice problems

1. Produce the following picture:



2. Plot the cubic function $x^3 + 53x^2 - 400x + 25$ from x = -30 to x = 30, in green. Include the grid and mark each value of the function for an integer argument with a point marker. Annotate the graph as 'cubic polynomial'
3. Describe the output of the following code, annotating it with comments. What title would you give this visualization?

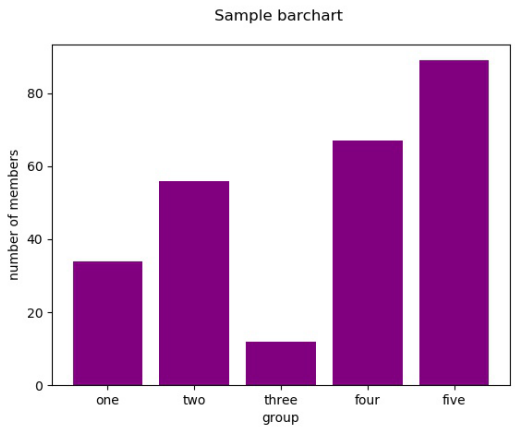
```
def main():
    moviesDF = pd.read_csv('IMDB.csv')

    moviesDF.TotalVotes = moviesDF.TotalVotes.str.replace(',','')
    moviesDF.TotalVotes = pd.to_numeric(moviesDF.TotalVotes)

    portion = moviesDF[moviesDF.Genre1=='Drama'].loc[:,('Title', 'TotalVotes')]
    portion.sort_values(by='TotalVotes', inplace = True)

    plt.plot(portion.Title, portion.TotalVotes, 'bo')

    positions,titleLabels = plt.xticks()
    plt.setp(titleLabels, rotation = 30, ha = 'right')

    plt.grid()
    plt.show()
```

4. Plot the average rating per each movie genre, based on the IMDB.csv file.

5. Consider grades data files posted within this week. Plot the combined grades data from the two files, using different colors to display problem scores for each student.
   - Compute the total grade (sum of all four column values) and add it to the plot
   - Display the same data in the decreasing order by the total score (hint: use a DataFrame to store and sort the values).

BARCHART

```
def barPlot ( filename ):
    ''' Simple barchart, saved in a file'''

    plt.figure()  # start a new figure
    itemSeries = pd.Series([34, 56, 12, 67, 89],
                           index = ['one', 'two', 'three', 'four', 'five'])

    plt.title("Sample barchart \n" )
    plt.xlabel('group')
    plt.ylabel('number of members')

    # display the bars
    plt.bar(itemSeries.index,
            itemSeries.values, color = 'purple')

    plt.xticks(np.arange(len(itemSeries)),
               itemSeries.index,
               ha = 'center')

    plt.savefig(filename)

def main():
```



```
    barPlot('plotimg.jpg')
    plt.show()

main()
```

SUBPLOTS

```
import matplotlib.pyplot as plt
import numpy as np

fig2, subplots_fig = plt.subplots(2, 2)  # a figure with a 2x2 grid of Axes
print(fig2, '\n', subplots_fig)

topleft = subplots_fig[0,0]
topleft.plot(range(10), np.random.random(10), 'x', color='blue', label='sample 1')
topleft.plot(range(10), np.random.random(10), 'o', color='orange', label='sample 2')
topleft.grid()
topleft.legend()
topleft.set_title("Dots")

bottomright = subplots_fig[1,1]
x = np.arange(-5, 6)
bottomright.plot (x, 2*x, color = 'red'   )

fig2.suptitle('four subplots demo')

plt.show()
```