MATPLOTLIB TUTORIAL

Go through the tutorial and complete all exercises.
Make a single python file with ALL of the solution.

For each Exercise, add comment line as follows
<A Blank Line>
# Code for Exercise 1
(Followed by the code)

In the output make sure the printout contains

<A Blank Line>
 Exercise 1
(Followed by the results of Exercise 1)

Turn in one file called <lastnameMatplotLibTutorial.py> and <lastnameMatplotLibResults>.txt

Any code that says "Run the following code and observe the printed results" need NOT be turned in.

Introduction

This tutorial will provide an introduction to matplotlib and its uses. It will discuss the following topics:

1) Install matplotlib
2) Create a plot
3) Add/modify parts of a figure

We will be teaching this course using the Spyder through Anaconda.

[Note for experts: If you are already familiar with data science, you may prefer a different IDE or platform such as Jupyter Notebooks.]

More documentation on NumPy can be found from the matplotlib website

Notes on these notes

In these notes this font indicates something that appears on a computer screen or which should be typed in to the computer *this font* indicates where you should substitute something. For example:

```
edit filename
```

means type edit followed by the name of a file. Comments in square brackets are asides or explanatory notes often these are intended for experts who might want to know more about the subject.

What to do if you are stuck in an exercise

Try rereading the previous bits of the worksheet or reading ahead a little to see if this helps. If not, then there are model answers for each exercise available – these are all on the website in the same directory as sourcecode for this worksheet. You will see how to access this later in the worksheet.

Installing matplotlib

Matplotlib should be installed when Anaconda and its packages are installed. Otherwise, it can be installed in the command line with the following:

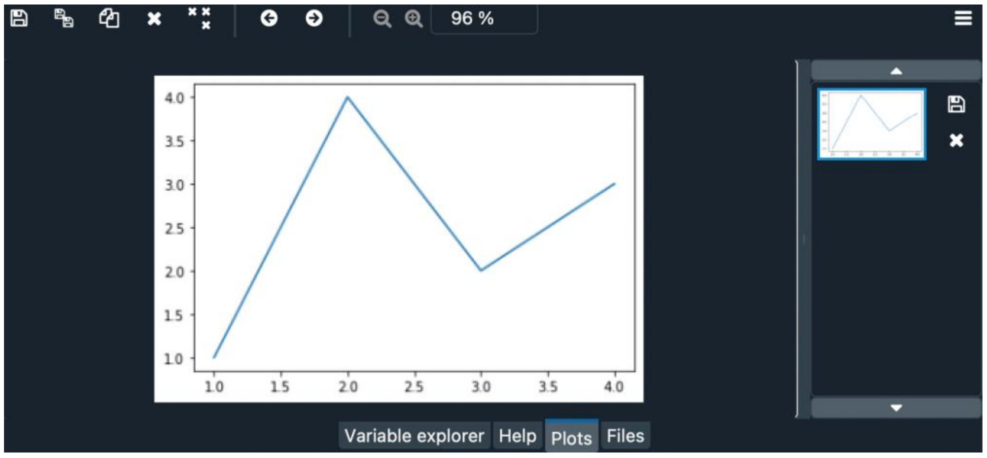
```
pip install matplotlib
```

To check what version of matplotlib you have, type the following into the command line:

```
pip show matplotlib
```

Viewing Plots in Spyder

In the Spyder IDE, plots are not printed to the console, but rather the "Plots" pane:



The format in which plots are displayed may vary for different IDEs.

Introduction to Plotting

Matplotlib graphs your data on Figures (i.e., windows, Jupyter widgets, etc.), each of which can contain one or more Axes (i.e., an area where points can be specified in terms of x-y coordinates (or theta-r in a polar plot, or x-y-z in a 3D plot, etc.).

Creating a Plot

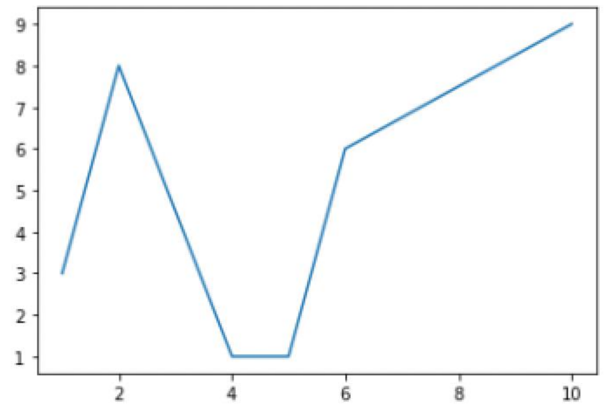
The most simple way of creating a figure with an axes is using pyplot.subplots. We can then use Axes.plot to draw some data on the axes.

Run the following code and observe the printed results:

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

fig, ax = plt.subplots()  # Create a figure containing a single axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])  # Plot some data on the axes.
```

Exercise 1:
Create a plot that resembles the following table of graph. Name the figure **fig1**.



Parts of a Figure

Figure

The **whole** figure. The figure keeps track of all the child Axes, a smattering of 'special' artists (titles, figure legends, etc), and the **canvas**. (Don't worry too much about the canvas, it is crucial as it is the object that actually does the drawing to get you your plot, but as the user it is more-or-less invisible to you). A figure can contain any number of Axes, but will typically have at least one.
The easiest way to create a new figure is with pyplot.

Run the following code and observe the printed results:

```
fig = plt.figure()  # an empty figure with no Axes
fig, ax = plt.subplots()  # a figure with a single Axes
fig, axs = plt.subplots(2, 2)  # a figure with a 2x2 grid of Axes
```

Axes

This is what you think of as 'a plot', it is the region of the image with the data space. A given figure can contain many Axes, but a given Axes object can only be in one Figure. The Axes contains two (or three in the case of 3D)Axis objects (be aware of the difference between **Axes** and **Axis**) which take care of the data limits (the data limits can also be controlled via the axes.Axes.set_xlim() and axes.Axes.set_ylim() methods). Each Axes has a title (set via set_title()), an x-label (set via set_xlabel()), and a y-label set via set_ylabel()).
The Axes class and its member functions are the primary entry point to working with the OO interface.

Axis

These are the number-line-like objects. They take care of setting the graph limits and generating the ticks (the marks on the axis) and ticklabels (strings labeling the ticks). The location of the ticks is determined by a Locator object and the ticklabel strings are formatted by a Formatter. The combination of the correct Locator and Formatter gives very fine control over the tick locations and labels.

Artist

Basically everything you can see on the figure is an artist (even the Figure, Axes, and Axis objects). This includesText objects, Line2D objects, collections objects, Patch objects ... (you get the idea). When the figure is rendered, all of the artists are drawn to the **canvas**. Most Artists are tied to an Axes; such an Artist cannot be shared by multiple Axes, or moved from one to another.

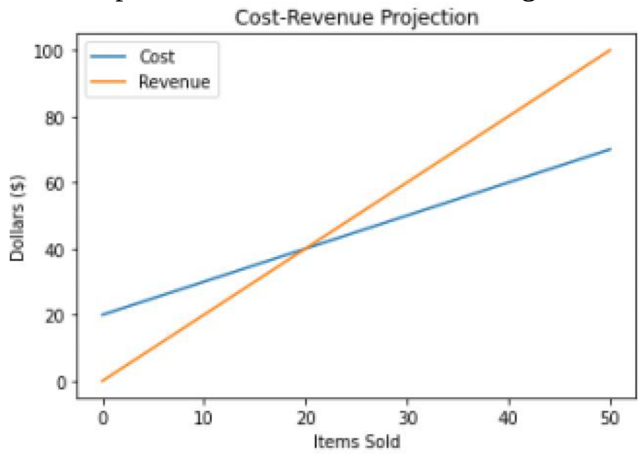
The following code example uses NumPy and matplotlib to create a plot. Take note of the code that produces different features of the plot, including titles.

Run the following code and observe the printed results:

```
x = np.linspace(0, 2, 100)

fig, ax = plt.subplots()  # Create a figure and an axes.
ax.plot(x, x, label='linear')  # Plot some data on the axes.
ax.plot(x, x**2, label='quadratic')  # Plot more data on the axes...
ax.plot(x, x**3, label='cubic')  # ... and some more.
ax.set_xlabel('x label')  # Add an x-label to the axes.
ax.set_ylabel('y label')  # Add a y-label to the axes.
ax.set_title("Simple Plot")  # Add a title to the axes.
ax.legend()  # Add a legend.
```

Exercise 2:
Create a plot that resembles the following table of graph. Name the figure **fig2**.


Cost-Revenue Projection
Cost
Revenue
Dollars ($)
Items Sold