Explore More!

Points awarded: _____

Name: _____

Net ID: _____

Module 004 (previously 10A): Plotting with Python

Laboratory Outline

In lab we have used MATLAB to store, manipulate, and plot our measurement data. MATLAB stored our data in arrays, and made it easy to generate professional looking, highly customizable plots. Over time MATLAB has grown to include a wide array of toolboxes and functionality that make it an endlessly useful tool in scientific and engineering fields. However, many alternatives exist, some of which are free to use and open-source. Python, while not by itself a MATLAB replacement, has a number of scientific libraries available that make it a fine substitute for MATLAB. Plus, being open-source you can install it free of charge, and contribute to larger coding projects with it online. In this module we are going install and explore some basic tasks with Python.

Prerequisites

- A Python installation, including the matplotlib package and the Spyder IDE.

Parts Needed

- None. This is a programming exercise.

At Home: This exercise can be completed on any computer with the appropriate Python packages installed on it.

Installing Python via Anaconda

Before we get to plotting, we first want to install Python along with some scientific libraries and an Integrated Development Environment (IDE) to work in. Instead of installing each of these individually, common practice is to install a Python distribution that includes a package manager program and helps keep your installation tidy. For this module we are going to install Anaconda (this module is clearly not for ophidiophobes).

Visit https://www.anaconda.com/download/ to download an installer for Windows, Mac, or Linux. We will be using the Python 3.6 version. The installer should be pretty straight forward, but docs.anaconda.com has additional documentation if you need it. Once installed you should have a new folder called Anaconda3 in your programs listing. All of them are worth tinkering with, but for this module we are going to be making use of the program Spyder. Spyder is an IDE, meaning it is where we will type

Using EWS lab computers?

You can install Anaconda on your lab computer or any EWS machine. However, during the installation make certain that you install it just on your account (other options require admin privilege)

Notes:

our code, work with our variable workspace, and have access to a Python command prompt. Functionally and design-wise, it will feel very similar to MATLAB.

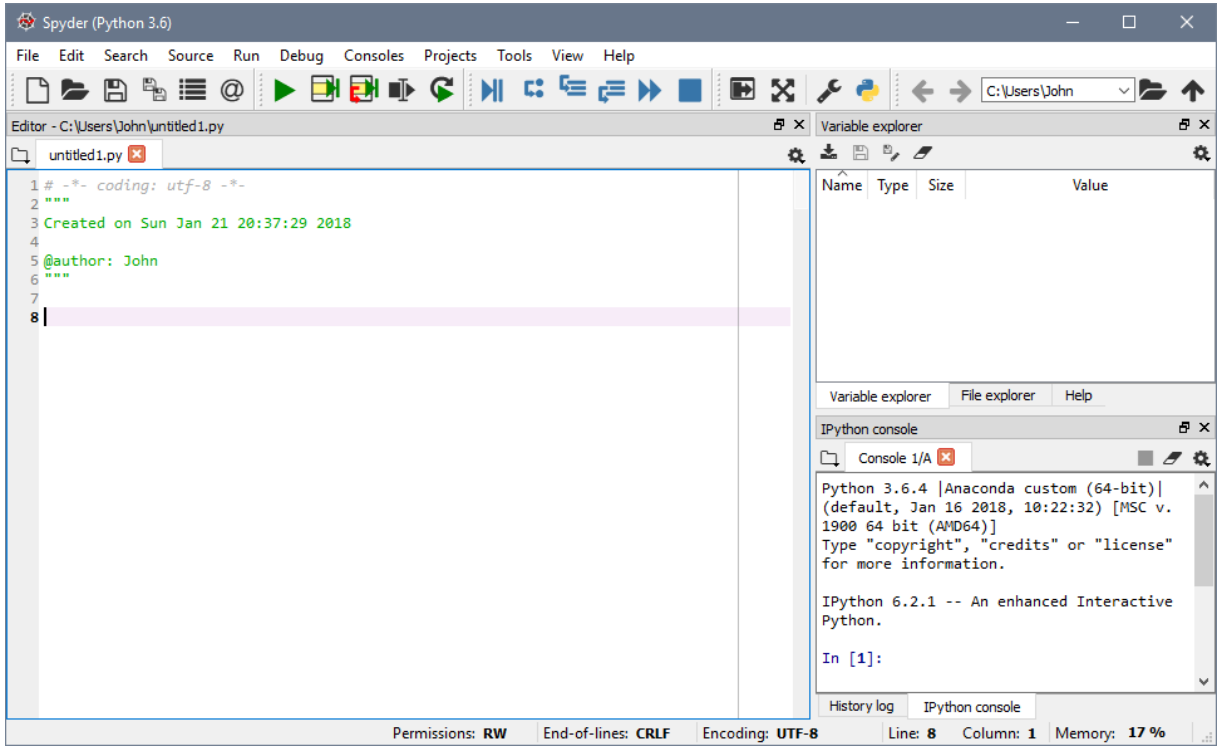

Figure 1: A new instance of Spyder

Importing necessary packages

Python is a general-purpose high-level programming language, and a very popular one too! However one of its greatest strengths is the number of packages available online that can expand its functionality and make it better suited for specific tasks. When we wish to use Python as a MATLAB substitute, two packages are of particular interest. The packages are described as:

- NumPy: the fundamental package for scientific computing with Python. It contains among other things:
  - a powerful N-dimensional array object
  - sophisticated (broadcasting) functions
  - tools for integrating C/C++ and Fortran code

Notes:

    - useful linear algebra, Fourier transform, and random number capabilities
- Matplotlib: a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

We must first import these packages before we can use any of the commands they contain. Instead of working directly from the command line, which is labeled as the "IPython console" in the lower right corner of the Spyder interface, we will write a Python program, so that we can execute all of our commands at once. A blank program should already be open in the main editor window. Now we will add some code that imports our packages, creates two arrays, and plots them. Enter the code and execute the program by pressing the green "play" triangle icon on the top bar. We'll explain what the code does as we progress.

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

The import commands import two sets of commands from the numpy and matplotlib packages. We also assign two keywords, np and plt. We will need to preface any commands from these packages with the appropriate keyword.

```
voltage = np.array([0,1,2,3,4,5,6,7,8,9,10])
current = np.array([0,1,4,9,16,25,36,49,64,81,100])

plt.figure()
plt.plot(voltage,current)
```

Above we defined two numpy arrays, and then used matplotlib to create a new figure and to plot our data. Again, the basic approach is almost identical to what we have done in MATLAB, just with certain syntactical changes. Note: these are just placeholder values we used. For this module***, instead of the given voltage and current values above***, use values from a past lab or prelab. Include what assignment your data is from in the plot title, as well as your netID.

```
plt.title('My first Python plot: netID')
plt.xlabel('Voltage (V)')
plt.ylabel('Current (A)')
plt.grid(axis='both')
```

The above commands apply a title, labels, and grid markings.

At this point, with all of the above code added to our program, execute the script.

Note*:* by default Spyder will display figures inline in the IPython console, instead of opening them in new windows. Depending on the sort of work you are doing, sometimes this is a more convenient way to quickly take a look at some data. You can change the behavior by going to Tools> Preferences> IPython console> Graphics tab> and change Backend from "Inline" to "Automatic."

Notes:

So at this point we should have a plot that simply displays some data from a past lab or prelab. The plot should look similar to that shown in Figure 2, but with your data displayed.

Task: At this point, print out your plot making certain that the title contains the lab number your data is from and your netID.

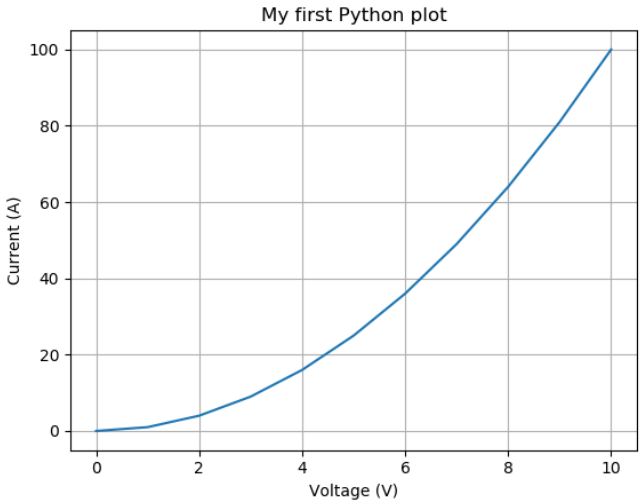

Figure 2: A simple Python plot

A different look at data

Formatting and presenting data is not something an engineer should treat as an afterthought. Intuitive ways of showing data can help make your presented information more accessible, and even reveal certain trends of properties that are not otherwise obvious.

https://matplotlib.org/tutorials/introductory/sample_plots.html shows a number of sample plots that one can generate using matplotlib. Using any of the varied plot types on the site, generate a well labeled and titled plot of some data of your choice. Get creative, you can use any data you want. You can try using bar graphs, pie charts, pcolor contour plots, or 3D graphs. Maybe a pie chart of how you spend your time on an average school day, or 3D plot a particularly interesting mathematical function. Label your axes appropriately.

Task: At this point, print out your second plot making certain that the title and axes properly describe the data you are presenting. Also make certain your netID is in the title.

Notes:

Learning Objectives

- Learn an alternative to MATLAB for plotting data via the matplotlib library for the Python programming language.
- Figure out creative ways to present data beyond basic 2D line plots.

Explore Even More!

When you installed Anaconda, a program called Jupyter Notebook was installed as well. Jupyter is a great way of making interactive Python applets that can be easily shared and displayed in a web browser. Try making your own notebook, using an example notebook from the matplotlib site as a starter template. All example pages on matplotlib.org should have a link to a notebook at the bottom of the page.

Figure 3: Download links from the matplotlib introductory tutorials page