ENGR 102 - Lab #10

This week's lab and individual assignments are meant to familiarize you with two of the most commonly used engineering packages in Python, numpy and matplotlib. It will also begin to give you more experience in making function calls and using a module. To do this, as a team you will work some of the basic tutorial/intro material for each one, and ensure that each member of the team knows how it works.

Numpy is a package for performing scientific computing. In particular, it offers functions and types that will allow you to create vectors and matrices, and perform linear algebra operations more easily. The website for numpy is: http://www.numpy.org/

Matplotlib is a library for creating plots, graphs, and charts. It supports a wide range of plots and graphs, and is often used for visualizing data. The website for matplotlib is: https://matplotlib.org/

Activity #0: Check for package installation – to do individually.

Both of these should already be installed on your system, when you downloaded the Anaconda distribution of Python, and you should have checked this in week 1. If you did not get the Anaconda distribution, you may need to download these on your own. You should first check to ensure they are installed on your system.

Create a program with two lines of code:

```
import numpy
import matplotlib
```

and try to run this program. If the program runs without an issue (exit code 0), then you have both packages installed. If you get an error, then at least one is not installed. You may need to ask for help in getting this to run.

Activity #1: Numpy – to be done as a team

The point of this activity is to familiarize your team with the basic matrix and vector structures and operations in numpy. It will also begin to give you more experience in making function calls and using a module. You will do this by going through the numpy tutorial as a team, and making some small modifications along the way.

a) Go to the "Quickstart Tutorial" by following the link to the "NumPy Tutorial" on the numpy home page. The quickstart tutorial shows several commands in the context of an interactive python interpreter. Your team should have one team member open an IDE and try entering and modifying the code shown there, as described below. Each team member may want to have a separate laptop open to read the documentation as you work through the examples.
b) As a team, you are to work through parts of the tutorial (listed below). For each part, your team should do the following:
    a. If there is text, read it, and check to make sure that everyone followed it. For any code in a section, do parts b-d.
    b. Write the code in your own browser and execute it. You will probably need to put in print statements to print out values of variables that are shown automatically in the interactive view.

c. Verify that the output is what you expect – that the arrays are created correctly and the values are computed.
d. Make some modification to the code (e.g. to the entries of a matrix) and rerun to make sure you see how the various functions behave.
e. Ask everyone in the group if they follow what has happened in that section. Do not go on until each person states that they understand.
    i. If some team members do understand, help explain the topics to each other.
    ii. If you are all stuck, or need help, ask the TA/Peer Teacher
f. Then, go on to the next section.

c) The parts of the tutorial to work through (there is a "table of contents" at the right of the page) are:
    a. The Basics – an example
    b. The Basics – array creation
    c. Shape Manipulation – Changing the shape of an array
    d. The Basics – printing arrays
    e. The Basics – basic operations
    f. The Basics – universal functions
    g. Linear Algebra – Simple Array Operations (you do not need to understand eigenvalues/eigenvectors, but can see the command used).
d) At the end, hopefully each person on the team understands how to create arrays of various sizes, how to perform basic linear algebra operations (like get dot product or do a matrix multiply) or call functions for more advanced linear algebra operations.
e) Finally, to demonstrate that your team understands how this process works, your team should create a small program that does the following:
    a. Your program should start with a statement in comments at the top: "As a team, we have gone through all required sections of the tutorial, and each team member understands the material."
    b. Creates 4 matrices, A, B, C, and D, of size 3x4, 4x2, 2x3, and 3x1. You can use randomized or hardcode values for the entries. Output each of these matrices.
    c. Computes the product E = ABC and outputs the resulting matrix.
    d. Prints the transpose of E.
    e. Solves the linear system Ex=D for x (this is the solve command). Print x.

    Turn in this program for Activity 1.

Activity #2: Matplotlib – to be done as a team

The point of this activity is for your team to review some of the capabilities of matplotlib, get practice learning how to find the parameter settings for various graphs, and make a couple of plots each. For your team exercises, you are to, as a team, create 8 different plots. The expectation is that each team member will create 2 such plots, but that you will all work together to ensure everyone knows how to handle different aspects of the process. Here are some details:

- You will need to go to the matplotlib website. Similar to what was done with Activity #1, you should work through the code provided on this page for training.
    - Follow the "Tutorials" link and then select "Pyplot tutorial". Your team should go through the tutorial, as was done for Numpy in Activity #1.
    - You should also check the "Examples" page to see, as a team, a range of examples of plots that can be produced.

    - Do not do more until everyone has followed the tutorial, and agreed that they understand it, and that the group has briefly looked at several Examples
- Your team should create a program that produces 8 plots of data of your choice. You may make up the data to be used in the plots/charts (see details below):
    - Your team should produce 8 different types of charts/plots. You do not want to create more than one of the same type of plot.
    - Each individual should produce 2 plots. You may help each other as much as you wish to make the plots, but each person is expected to fully understand, and have written the code, for the plots that person is "assigned".
    - If you have a smaller team than 4, you only need to create a number of plots that's twice the number of team members.
    - At least one plot per person should come from the Pyplot API (as opposed to an addon). It is OK for all 8 cases to be Pyplot examples.
    - You can generate 8 different programs, or combine your plots in one program (the choice is up to you).
- To see how to create a graph, go to the Pyplot API page (follow the "docs" link on the matplotlib page, then "API Overview", then "The pyplot API". From here, if you follow the link to "matplotlib.pyplot", you will find the details of the Pyplot API).
    - There are a wide variety of functions listed, each with a short description. You will likely need to use more than one function call per plot (e.g. one for the legend, one for the data plot itself, one for labeling the x axis, etc.)
    - Following one of the links for a function, you will see a function description, along with a list and description of parameters (and what is returned).
    - Note that in the "call signature", where the parameters are listed, sometimes you will see an '=' sign after a parameter, along with some value. This indicates that if you don't specify that parameter, it will have the value specified by the = sign.
    - Note that for many functions, there is an example showing how it can be used with the end result. This is sometimes more instructive than the parameter descriptions. **You are encouraged to look at the examples and tutorials!** However, you should not just cut-and-paste the examples and tutorials.
- Each person needs some data set that will be displayed; you may share the same data sets across your team, however you wish. You can generate the data however you wish. This can be random data entered as hardcoded lists/arrays, or it can be something automatically generated (or even something read from a file if you wish). This does not have to be real data in any sense – just something that can be used to display the plot type.
    - Note: Your data should not be just a small modification (tiny change in parameters) of an example/tutorial. However, you do not need to be elaborate in creating data. It should be varying enough that the end result is varied/interesting (e.g. plotting 100 of the same value is not interesting).
    - You should try to make your plot have more than just the minimal data plot. That is, include labels/legends/etc. to make it clear what is being plotted.
    - Remember that the intention here is to work together as a team. Even though each person might generate different plots, you will probably all have similar questions regarding how to set up parameters appropriately.
- After all 8 plots are created, you should, as a team, review each other's plots. Let each person show the code (it is likely just a few lines per plot) they wrote, and briefly explain it to the others on your team. The goal here is for everyone to be at least somewhat familiar with the way that

various plots are created, and have a good sense of the challenges faced in doing so, and in terms of what needs to be specified by a programmer.