EMAT10007 – Introduction to Computer Programming

Exercises – Week 8. Matplotlib

Exercises this week will explore the use of Matplotlib, a python module designed for data visualisation.
Note: The standard was to import the matplotlib module is by adding the following line at the start of your code: `import matplotlib.pyplot as plt`. Any function belonging to the matplotlib module can then be accessed by writing, for example, `plt.plot()`.

Part 1. Intro to MatPlotLib

Exercise 1 - Line graph

1. Create two lists of integers named `x` and `f` with the following values:
   `x = [0,2,4,5,8,10]`
   `y = [1,3,3,3,4,5,6]`
   Now try to plot these values using the `plt.plot()` function. What error occurs? Can you guess why?
2. Try again with the following lists:
   `x = [0,2,4,5,8,10]`
   `y = [1,3,3,4,5,6]`
   Hint: Remember to use `plt.show()` to display the graph.
3. Now alter your graph so it has the following:
   - x axis label: `x`
   - y axis label: `y`
   - title: `Plot of y vs x`
4. Finally, modify your code so the resulting graph looks like the graph in the following figure:

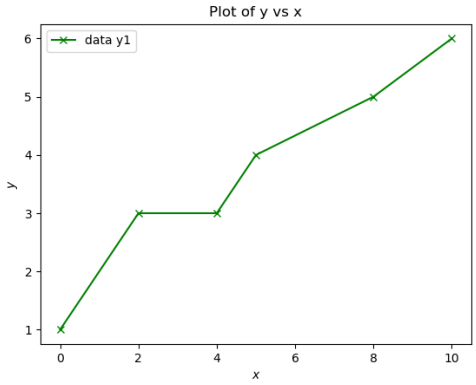

Figure 1: Final plot for Exercise 1.

Hint: Use the optional format string and `plt.legend()` function as described in the lecture notes.
You can look up detailed information on the `plt.plot()` function here: *https://matplotlib.org/3.1.1/api/_as_gen/m*
including details of the marker styles and colours that can be displayed in the optional format string.

Part 2. Saving plots and importing

Exercise 2 - Histograms and csv

1. Create a function named `diceRolls` that prints out the results of a number `n` of dice rolls. Make `n` a variable that can be passed to the function.
   Hint: Use the numpy `np.random.randint()` function.
2. Create a .csv file named `diceRolls.csv` in the same folder as your code (you can do this using notepad, excel...). Modify the `diceRolls` function so it saves the results in `diceRolls.csv`.
   Note: Remember to add `import csv` at the start of your code.
   Hint: Replace the `<?>` in the following code with the result of the diceRolls function:
   ```
   with open('diceRolls.csv',mode='w') as f:
       writer = csv.writer(f, delimiter = ',')
       writer.writerow(<?>)
   ```
   Run your code with `n = 100` and check your `diceRolls.csv` file to make sure data has been saved.
3. Now import the dice rolls from your diceRolls.csv file and save it as a numpy array named `data` as described in the lecture slides. Print out the values of data to screen.
4. Let's try to visualise that data. First, try using `plt.plot()`. This shows the results of each dice roll, but is slightly confusing to look at.
5. Now, let's try to represent the distribution of dice rolls using a histogram. Use `plt.hist()` as described in the lectures and `plt.show()` to visualise the histogram.
6. Add a title and x and y axis labels to your histogram. Try running your code with n = 10, 100, 1,000 and 10,000 to see how the distribution of dice rolls changes.

Part 3. Curve fitting

Exercise 3 - Polynomials

1. Generate two random numpy arrays of 20 floats ranging from 1 to 10, named `x` and `y`. Sort the lists.
   Hint: Use `np.random.uniform()` and `np.sort()`.
2. Display the data from the sorted `x` and `y` functions using the `plt.scatter()` function.
3. Use polyfit to determine the coefficients of a second degree polynomial (`deg = 2`) fit to your `x` and `y` data. Then use `poly1d` to generate a new array of fitted data, `yfit`.
4. Show the original data using `plt.scatter` and the fitted data using `plt.plot()` in the same graph. Add a title, legend and x and y axis labels to the graph.
5. Modify your code so the polynomial fit is now of the 5th degree (set the 3rd parameter of `polyfit` to 5). How doese this affect the fit to your data?