Introduction to Programming

Week 6, Lab 1: Plotting with matplotlib.

The goal of this lab is to get you acquainted with the basic functionality of NumPy.

Quick Reference

NumPy:

- NumPy is the core tool for performing numerical computations with Python.
- NumPy is optimised for performance.
- It has a very flexible syntax allowing easy operations on entire arrays (called vectors) of data (vectorization).
- To access NumPy use:

import numpy **as** np

- For detailed documentation please see: *https://scipy-lectures.github.io/intro/numpy/index.html*

Note: This lab is slightly different than previous labs. There are no optional parts. All tasks are very short and mandatory.

Task 1: Create a NumPy array (vector) of size 10 filled with zeros.

Task 2: Create a vector of size 10 filled with zeros, except for the fifth element, which should be set to 1.0.

Task 3: Create a vector with values ranging from 10 to 49 (inclusive), with step 1.

Task 4: Create a 3x3 matrix with values ranging from 0 to 8 (inclusive).

Task 5: Find indices of non-zero elements of the vector [1,2,0,0,4,0,1,18,0,0,2,5,0,7]. Print all non-zero elements.

Task 6: Create a 3x3 matrix with random values between 0 and 1.

The aim of this lab is to show you how to use matplotlib to produce high quality graphs.

Quick Reference

matplotlib:

- In order to use matplotlib we need to import it using
  import matplotlib.pyplot **as** plt
- plt.plot(x,y) will plot y as a function of x. Both x and y have to be lists or NumPy arrays of same size. The default plot style is a thin blue line.
- plt.plot(x,y,'r--') plots a red dashed line.
- plt.plot(x,y,'g--', linewidth=3) plots a green dashed line of width 3.
- We can set plot limits using plt.xlim and plt.ylim
- We can set x and y axis labels using plt.xlabel and plt.ylabel
- *CS = plt.contour(X, Y, Z, V)* will produce a contour plot of function $Z=Z(X,Y)$ and draw contours at values specified in array V.
- plt.show() will produce graph.
- For more details please see:
  https://scipy-lectures.github.io/intro/matplotlib/matplotlib.html

Note: All tasks are mandatory!

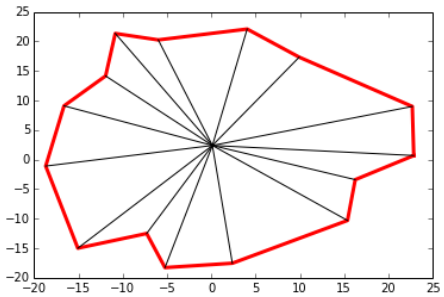
Task 1: Write a program that plots function $y(x)=e^{-x}\sin(x)$ for $x \in [0, 6\pi]$ computed at 100 equidistant points. Plot the graph using red dashed line of width 3. Label axes as "x" and "y". Use font size 16 for the axes labels.

Task 2: Add legend "y(x)" (no quotes) in *Task 1*. Change tick marks on x axis to show as "0", "π", "2π", "3π", "4π", "5π", and "6π" (no quotes). Use font size 16 for the tick marks.

Task 3: Generate 20 random points in the square $[-5,5]\times[-5,5]$ (that is, both x and y coordinates are chosen at random with equal probability from the interval [-5,5]). Plot all those points. Connect them with a green solid line of thickness 3. Label axes as "x" and "y" using font size 20.

Task 4: Read coordinates of a polygon from the file "polygon.dat". Draw this polygon. Make sure that first and last lines are connected. You can find 'polygon.dat' file on MyDundee labs page.

Task 5: Find the geometric centre of the polygon in Task 1. Draw the polygon (in red with line thickness 3). Connect each of its vertices to the centre with a thin black line. The result should look like this:



Week 6, Lab 3: Plotting from the data.

Goal of this lab is to show you how to combine NumPy and matplotlib to produce high quality graphical representation of the data.

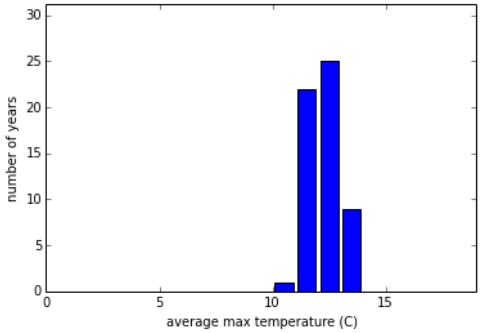
Quick Reference

matplotlib:

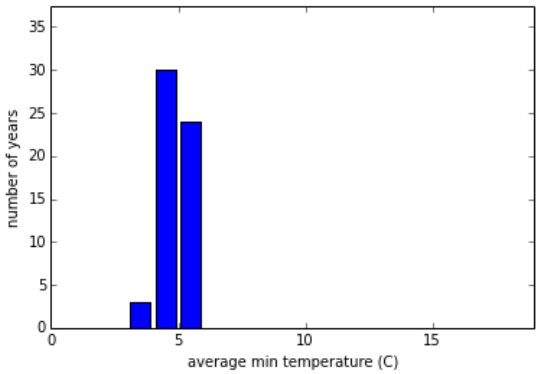
- In order to use matplotlib we need to import it using
  import matplotlib.pyplot **as** plt
- *plt.plot(x,y)* will plot y and a function of x. Both x and y have to be lists or NumPy arrays of same size. Default plot will be thin blue line.
- *plt.plot(x,y,marker='^',ms=8)* will plot y vs x using upper triangle as the symbol and set it to size 8.
- *plt.plot(x,y,marker='s', ms=6, linewidth=3*) will plot y vs x using square as the symbol, set it to size 6 and connect all symbols with a line of width 3.
- We can set plot limits using *plt.xlim()* and *plt.ylim()*
- We can set x and y axis labels using *plt.xlabel* and *plt.ylabel*
- We can place legend using *plt.legend()* function.
- We can create multiple plots using *plt.subplot()* function.
- We can set ticks labels using *plt.xticks()* function
- *plt.show()* will produce graph.
- Please check matplotlib documentation at: http://matplotlib.org/

Task 1: Read weather data from the two files "max_temp.dat" and "min_temp.dat". Both files can be found in the lab section on myDundee. File max_temp.dat contains average maximum temperatures for each month from 1957 until 2013 for the Leuchars weather station. File min_temp.dat contains similar data but for average low temperatures. Both files have 57 lines and 13 columns. Each line is a separate year. Column 1 is the year, columns 2-13 are months.

a) Plot histogram of average maximum temperatures (averaged over all months in a given year). Use bins of width 1 between 0 and 20. The output should look like:

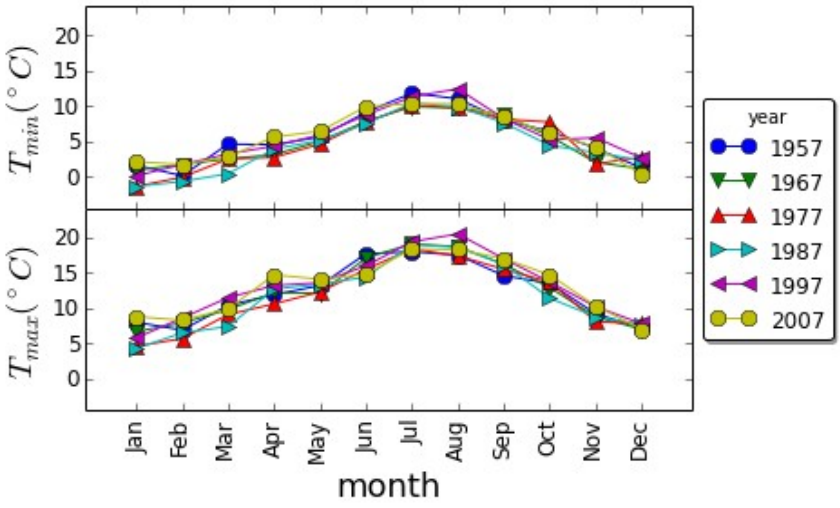


b) Plot histogram of average minimum temperatures (averaged over all months in a given year). Use bins of width 1 between 0 and 20. The output should look



Hint: You may find *np.histogram()* and *plt.bar()* functions very useful. Please check documentation for the details on how to use them.

Task 2: Use the same data files from Task 1 to produce this plot:



Note:

1. You can use following list of markers: markers = ['o','v','^','>','<','8','s','p','*','h','H','D'].
2. Marker size should be 8, line width 1.
3. You should plot every 10th year starting with the first year on the record (1957)
4. You can use *calendar.month_abbr[i]* (i=1,...,12) defined in the *calendar* module to get names of months.
5. ticklabels should use font size 12, axis labels should use font size 18.

Hints:

1. You may find *itertools.cycle()* function useful.
2. *plt.subplots_adjust()* will adjust relative position of two subplots.

3. You may use this code to remove ticks labels from the top subplot:
   frame = plt.gca()
   frame.axes.xaxis.set_ticklabels([])

4. *plt.margins()* function will set margins on your plot so the data does not look “glued” to the y-axis.