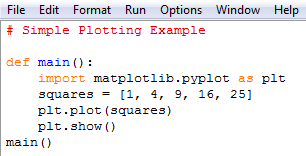
**PYTHON Programming Exercises 6**

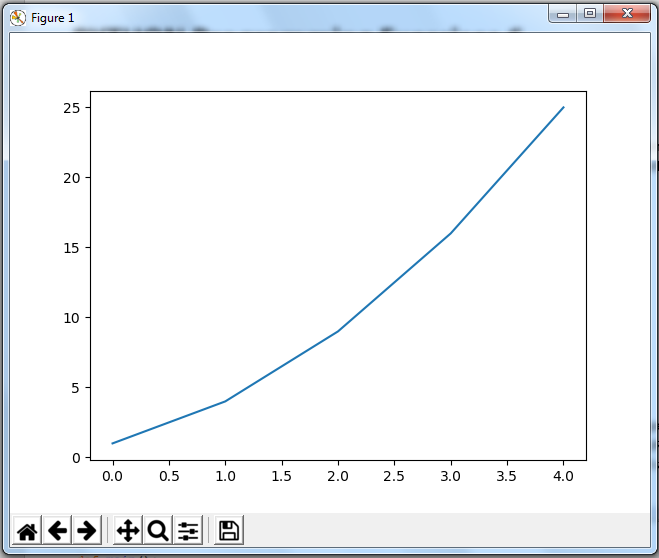
We’re going to take a look at some simple data visualisation using MATPLOTLIB.

**Task 1: Some Basic Plotting Commands**

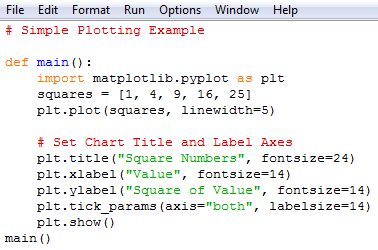
Let’s start with a simple line graph and then customise it to create a more informative visualisation of our data. We’ll use the square number sequence 1, 4, 9, 16, 25 for the graph. Open a new file (plot1.py) with the following and run it:



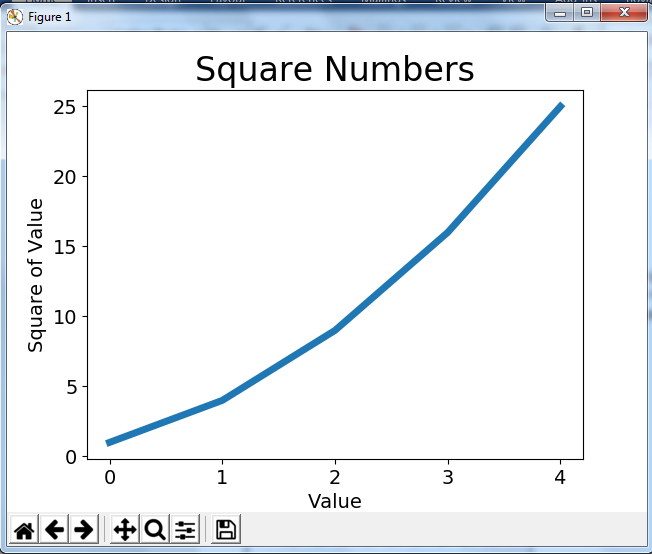
We first import the matplotlib.pyplot module using the alias **plt**, so we don’t have to type the full name repeatedly. (pyplot contains a lot of functions that help us generate charts and plots). We create a list to hold the squares and then pass it to the **plot()** function. **plt.show()** open’s matplotlib’s viewer and displays the plot:



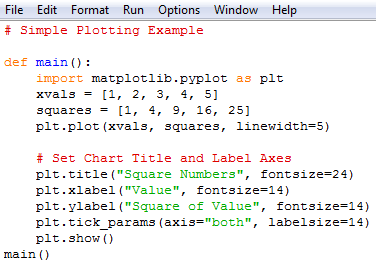
We can use some of the available customisations to improve the readability of this plot:

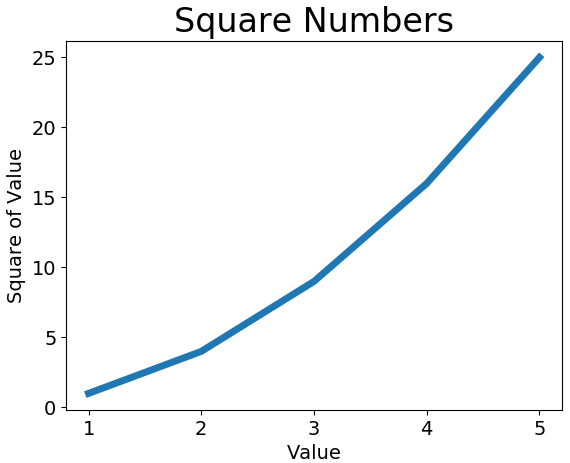


The linewidth parameter controls the thickness of the line; the title() function sets a title; the fontsize parameters control the size of text on the chart; the xlabel() and ylabel() functions allow you to set titles for the axes, and the function tick\_params() styles the tick marks:



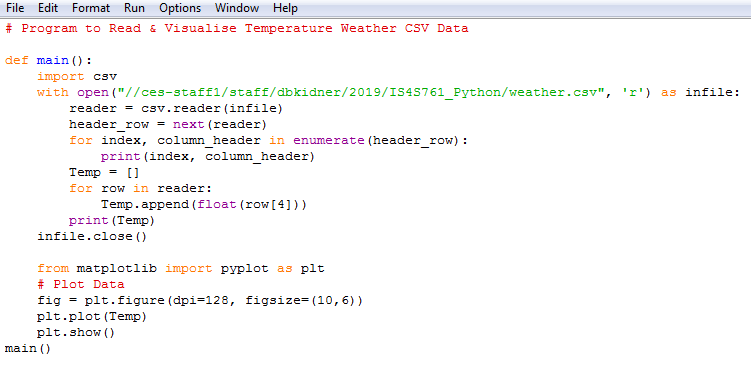
Now that we can see the graph better, can you identify the obvious error in it? Look at the X-value of 4 and its corresponding Y-value of 25. Oooops! When you give plot() a sequence of numbers, it assumes that the first data point corresponds to an x-coordinate value of 0 (but ours is actually 1). Let’s define the x-values and re-plot:



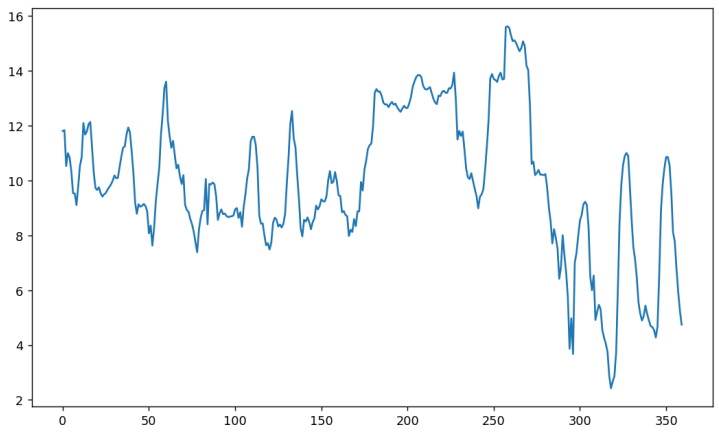


**Task 2: Visualising Our Weather Data**

From before, we saved our program to read the CSV file of our weather data and extract the temperature data for the fortnight to 1st November. Let’s now plot this data. REMEMBER to Edit the pathname to YOUR path of where you saved the data. Add the following 5 lines (starting “from matplotlib …”) to your program and save with an appropriate filename, e.g. weather5.py …

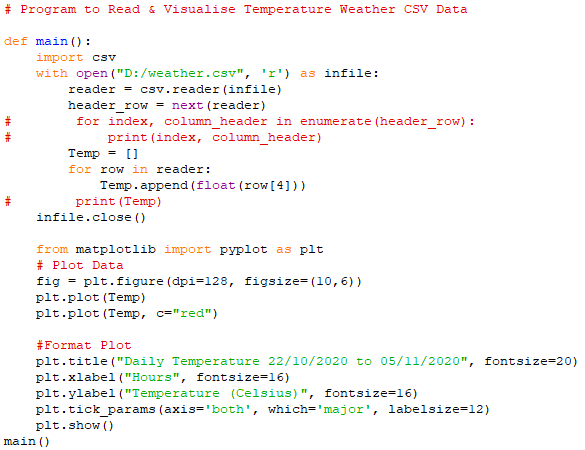


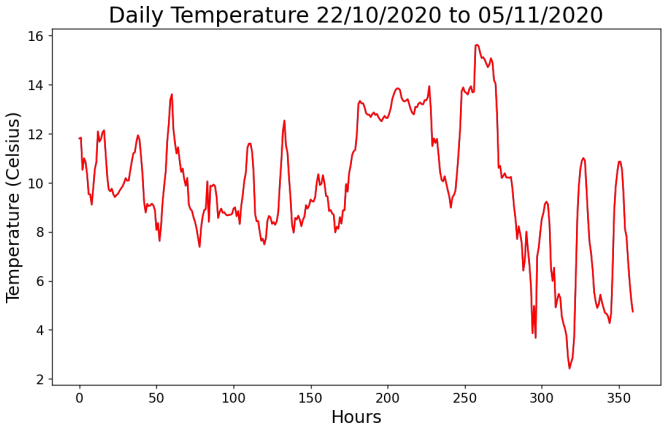
… and run the program. It might take a minute or so to load in the library and plot …



The X-axis corresponds to the hours in the 15-day period and the Y-axis corresponds to the Temperature in Celsius/Centigrade (It was COLD these last few nights!). Hopefully, we can relate the highs and lows to daytime and night-time temperatures respectively. Wasn’t it colder than that?

We can amend the program to add some more info to the chart (and comment out the print statements) and change the line to red (from its default blue):





Feel free to customise the graph as you see fit, or add to it. Check out some of the other options available to you on the MatPlotLib website. When you’re happy with your finished plot, let’s save it to a PDF file. Replace the **plt.show()** command with:



Where the pathname corresponds to **YOUR** workspace path. (We can save to a variety of different output formats. Find out which ones are supported by matplotlib and save the chart in a series of different formats. Look at the file size of each and the quality of the finished output graphic. Which one is best? Make sure that you can load up the PDF output.

**Task 3: The Definitive matplotlib Tutorial**

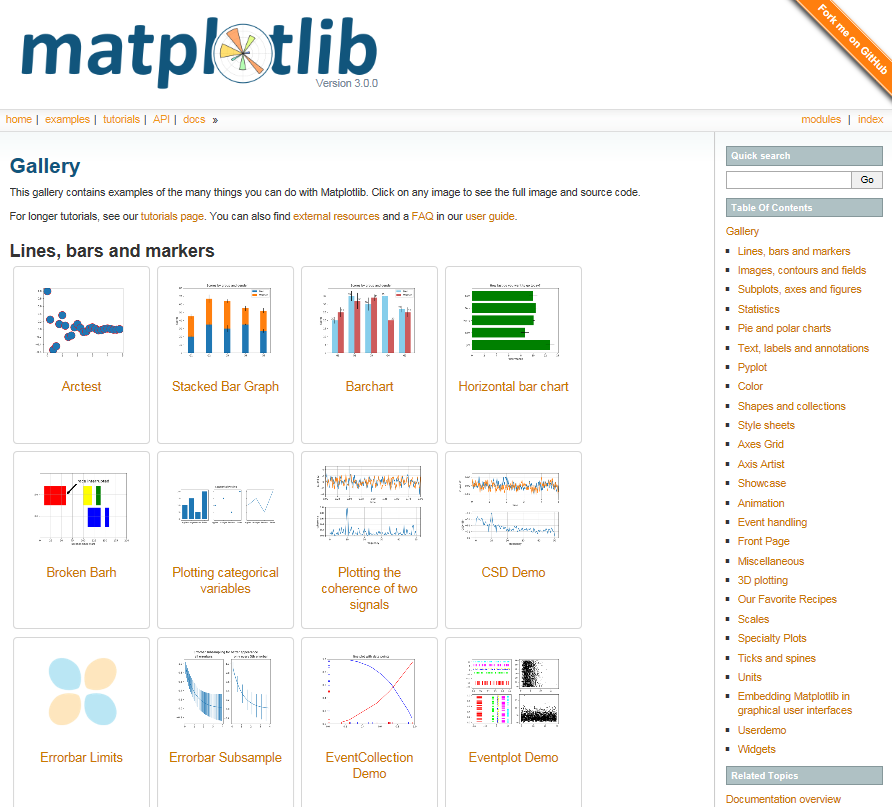
The following exercise is the definitive “learning the basics of matplotlib” tutorial. **Take 20 minutes to work through the exercises.**

[**https://matplotlib.org/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py**](https://matplotlib.org/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py)

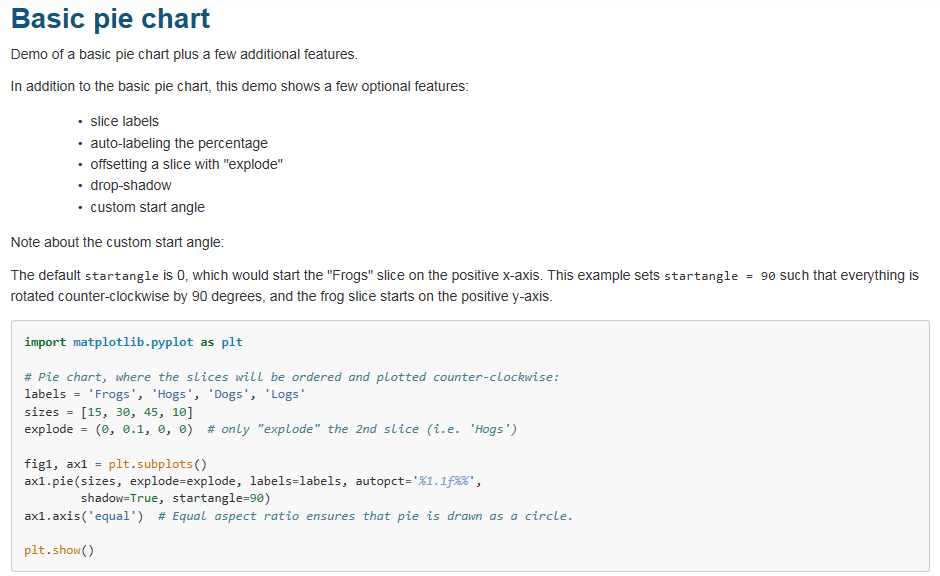
**Task 4: The matplotlib Gallery**

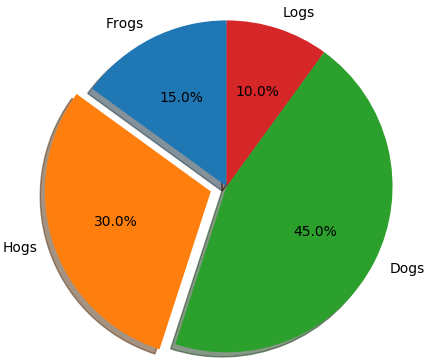
Take 20 minutes or so to check out the matplotlib Gallery (examples with code you can run and test with sample data to generate different types of charts (Bar charts, line graphs, pie charts, and customisation).

**https://matplotlib.org/gallery/index.html**

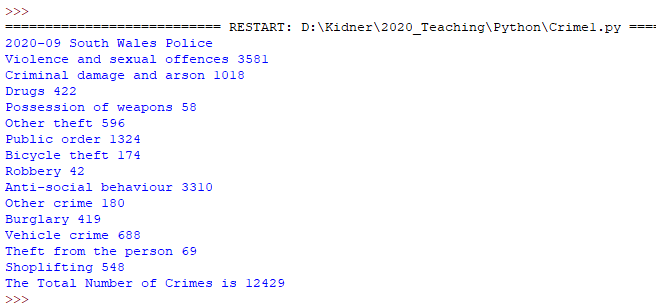


Don’t forget to scroll down the Gallery Examples, e.g. for Pie Charts …

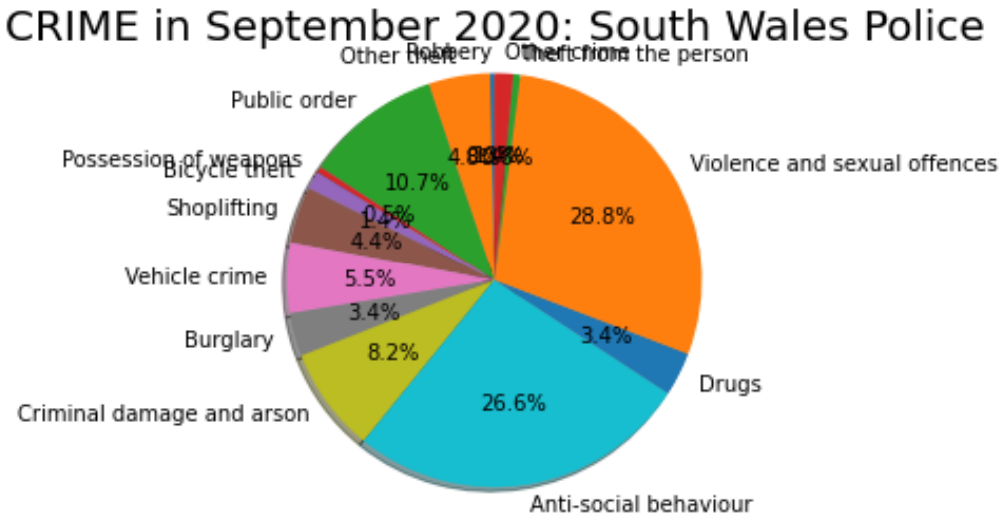




**EXERCISE 1:** In Exercise 2 of PYTHON Programming Exercises 5, you created some output relating to the September 2020 crime data for South Wales Police:

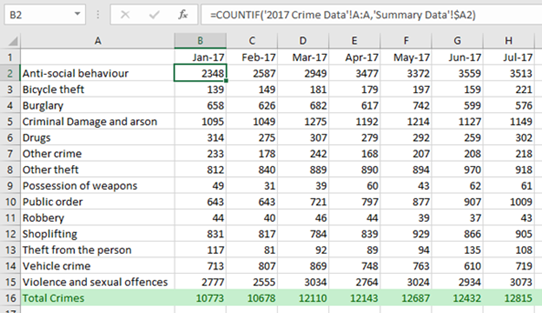


If you didn’t get this, then have a word with your tutor. Using the list you generated for the total crimes, can you generate a Pie Chart of this data?



Hint: You will have to calculate the number of crimes as a percentage to define the pie slices.

**EXERCISE 2:** In the PYTHON CW Tutorial, we considered the possibility of generating crime data for a series of months (in EXCEL using the COUNTIF function).



In PYTHON, can you now extend your code which counts the total number of crimes from just September 2020 to consider the last 12 months, e.g. October 2019 to September 2020 (similar to the output above). As you will need to download and open 12 CSV files in Python, it might be easier to define a function to do the repetitive work. Write out the generated data to a new file (or even a CSV file).

**EXERCISE 3:** Extend your solution from EXERCISE 2 above to:

1. Calculate the percentage (as a proportion of total crime) of each crime type for each month.
2. Visualise this data, e.g. bar charts? Multiple pie charts as sub plots? Explore!