45 Drawing graphs

45.1 About

MatPlotLib is a graphing library for Python. It can plot several different types of graphs such as line charts, bar charts, histograms, scatter plots and others.

45.2 Install MatPlotLib

In a command prompt type

```
pip install matplotlib
```

Make sure the installation is successful with the command:

```
pip list
```

45.3 Plot a line graph

Create a new project called **matplotlibProject** or similar. Don't create a main.py. Instead create a new file called **linegraph1.py**.

Add the line to import the MatPlotLib library

```
from matplotlib import pyplot as plt
```

Set up some data in lists. In this case the data is Malaysia's GDP, in billions of US dollars, from 1960 to 2010.

```
years = [1960, 1970, 1980, 1990, 2000, 2010]
gdpMalaysia = [1.92, 3.86, 24.49, 44.02, 93.79, 255.02]
```

Create a new plot:

```
plt.plot(years, gdpMalaysia, color="blue", marker="x", linestyle="solid")
```

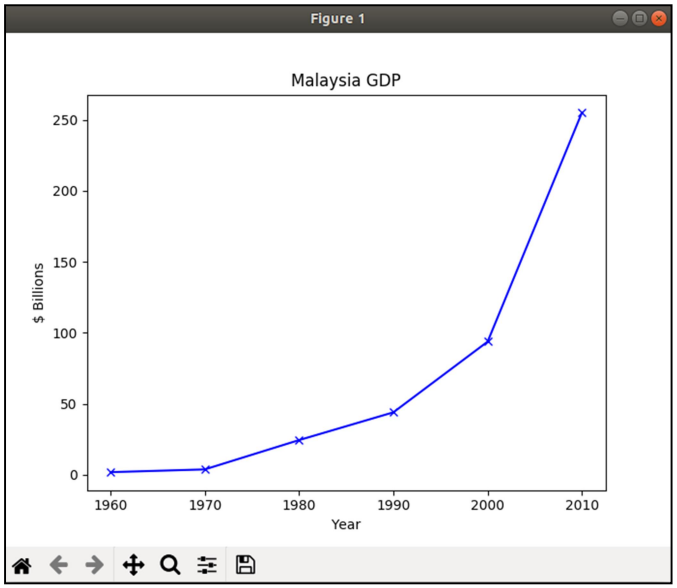
Add some labels to the graph:

```
plt.title("Malaysia GDP")
plt.ylabel("$ Billions")
plt.xlabel("Year")
```

These commands build up the plot in memory but it is not actually shown on screen yet. To show it, add the command:

```
plt.show()
```

Run the program and examine the output:



Note the strip of buttons that are automatically added in the bottom-left of the graph. Move your mouse over them to see their purpose. Try a few of them but note that they won't have much effect yet.

45.4 Add a second dataset

Compare the GDP of Malaysia with Singapore over the same period.
Add a dataset for Singapore in the same place as the others

```
gdpSingapore = [0.70, 1.92, 11.89, 36.15, 95.83, 236.42]
```

Add a plot. MatPlotLib allows several lines to be plotted on one graph. Add this directly under the matching line for Malaysia.

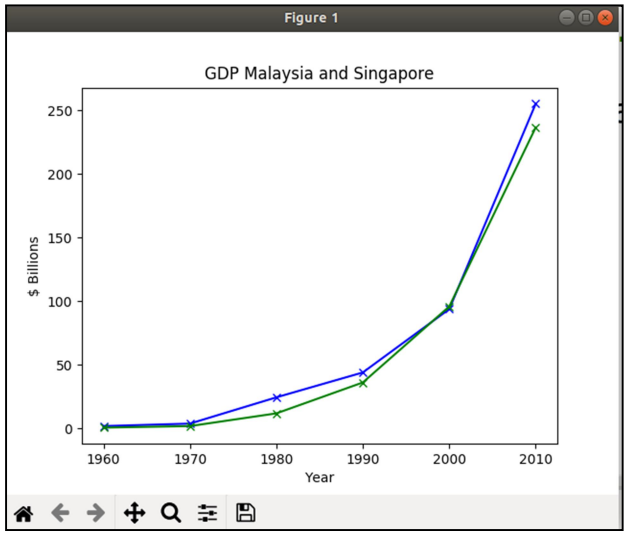
```

```

```
plt.plot(years, gdpSingapore, color="green", marker="x", linestyle="solid")
```

Change the title of the graph- change the existing line as follows:

```
plt.title("GDP Malaysia and Singapore")
```

Run the program and make sure the output is as expected:



45.5 Plot a bar chart

Different types of chart and graph are suitable for different types of data.
In this section you will plot a bar chart which is a good type of graph for visualising quantities.
Create a new file called **barchart1.py**.

Add the import:

```
from matplotlib import pyplot as plt
```

Create sets of data- in this case the populations of the the ten most populous countries in the world, in millions:

```
countries = ["China", "India", "USA", "Indonesia", "Pakistan",
             "Brazil", "Nigeria", "Bangladesh", "Russia", "Mexico"]
populations = [1401.5, 1359.3, 329.4, 265.0, 208.2, 211.2, 188.5, 168.2, 146.9, 126.6]
```

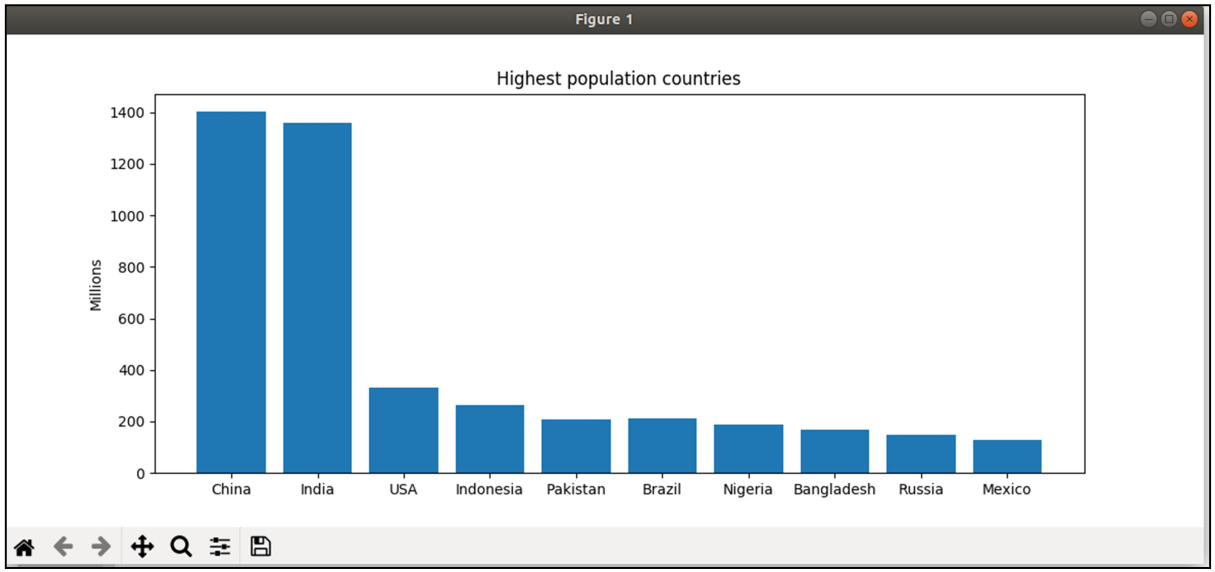
Add the plotting code:

```
plt.bar(range(len(countries)), populations)
plt.xticks(range(len(countries)), countries)
plt.title("Highest population countries")
plt.ylabel("Millions")
```

Show the graph:

```
plt.show()
```

Run the program.
The output might be a bit squashed to begin with- stretch the window to see the data clearly.



45.6 Read a set of data from a file

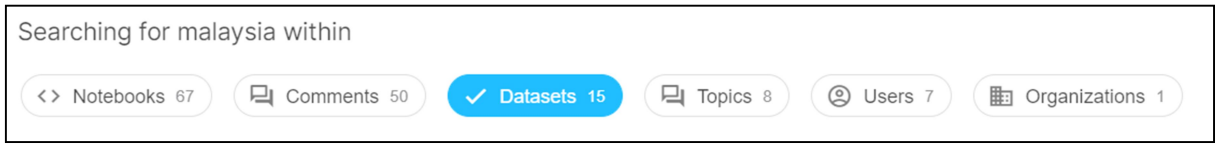
Instead of data directly coded inside our program like the previous examples, it would be a common task to read data from a file. In this example you will take data from a CSV file.

45.6.1 Get a dataset

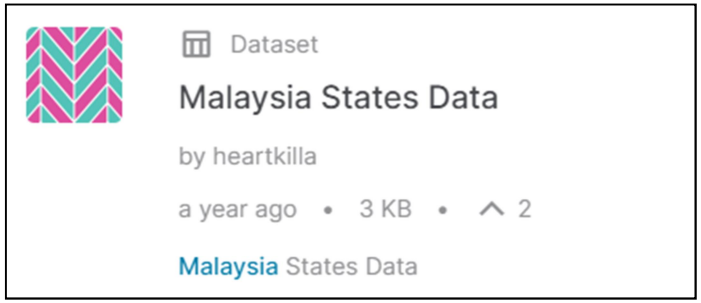
First get the data. Open a browser and visit www.kaggle.com.

Note that you will have to create an account to download any data. Use the **Register** button and follow the process. Continue with the steps below after you have logged in.

Enter **Malaysia** in the search bar. You will get a lot of results so narrow the results by clicking **Datasets**.



Scroll down the list to find **Malaysia States Data**.



Click to open the dataset. Download the files **gdp_info.csv** and **states_info.csv**.
Copy the two files into your project folder.

45.6.2 Tweak the data

Open the **gdp_info.csv** file in the editor and examine the data. Note that:

- The first row contains column headers
- The first column is the state name
- The numerical data is in “quotes”
- The numbers have comma separators (123,456)
- The last row is data for the whole of Malaysia
- The data is in reverse order- the first value is 2016, the last is 2010

```
State or Federal Territory,2016 GDP(RM Million),2015 GDP(RM Million),2014 GDP(RM Million),2013 GDP(RM Million),
Selangor,"280,698","239,968","226,964","212,645","200,906","187,434","177,718"
Kuala Lumpur,"190,075","160,388","152,380","140,534","131,514","122,890","113,095"
Sarawak,"121,414","106,063","102,318","98,089","94,013","92,700","87,131"
Johor,"116,679","98,880","93,665","87,974","84,050","78,946","74,102"
Penang,"81,284","69,844","66,200","61,324","58,353","55,827","52,946"
Sabah,"80,167","70,421","66,376","63,226","61,223","59,339","58,127"
Perak,"67,629","58,033","54,785","52,368","49,756","46,346","43,313"
Pahang,"52,452","45,882","43,946","42,201","40,047","38,148","35,871"
```

We could read the data in Python and remove all the quotes and commas but in this case it will be much easier just to resave the file without them. Double-click the file in a file explorer to open it in Excel.
This would be a good point to delete the last row (Malaysia total) and the first line (headings) as we don't need them.
Select the columns with numbers in them (all the columns except the first one), right-click, pick **Format Cells**, then **Number**. Set decimal places to zero and make sure the thousands separator option is switched off.
Excel now displays the data like this:

	A	B	C	D	E	F	G	H	I
1	State or Fe	2016 GDP(	2015 GDP(	2014 GDP(	2013 GDP(	2012 GDP(	2011 GDP(	2010 GDP(	(RM Million)
2	Selangor	280698	239968	226964	212645	200906	187434	177718	
3	Kuala Lum	190075	160388	152380	140534	131514	122890	113095	
4	Sarawak	121414	106063	102318	98089	94013	92700	87131	
5	Johor	116679	98880	93665	87974	84050	78946	74102	
6	Penang	81284	69844	66200	61324	58353	55827	52946	
7	Sabah	80167	70421	66376	63226	61223	59339	58127	
8	Perak	67629	58033	54785	52368	49756	46346	43313	

Pick **File>Save As** and save the file in CSV format again as **gdp_info_copy.csv** in the same folder.
Check that the resulting file has no commas in the numbers or quotes around the numbers.

```
State or Federal Territory,2016 GDP(RM Million),2015 GDP(RM Million),2014
Selangor,280698,239968,226964,212645,200906,187434,177718
Kuala Lumpur,190075,160388,152380,140534,131514,122890,113095
Sarawak,121414,106063,102318,98089,94013,92700,87131
Johor,116679,98880,93665,87974,84050,78946,74102
Penang,81284,69844,66200,61324,58353,55827,52946
Sabah,80167,70421,66376,63226,61223,59339,58127
Perak,67629,58033,54785,52368,49756,46346,43313
```

45.6.3 Create a script

In the editor create a new file called **csvplot1.py**.

Add the usual import at the top:

```
from matplotlib import pyplot as plt
```

Add some code to read the file. Start with code that reads the file and prints the lines.

```
infile = open("gdp_info_copy.csv")

for line in infile:
    print(line)

infile.close()
```

Save and run and examine the output.

```
State or Federal Territory,2016 GDP(RM Million),2015 GDP(RM Mil
2 GDP(RM Million),2011 GDP(RM Million),2010 GDP(RM Million)

Selangor,280698,239968,226964,212645,200906,187434,177718

Kuala Lumpur,190075,160388,152380,140534,131514,122890,113095

Sarawak,121414,106063,102318,98089,94013,92700,87131

Johor,116679,98880,93665,87974,84050,78946,74102

Penang,81284,69844,66200,61324,58353,55827,52946

Sabah,80167,70421,66376,63226,61223,59339,58127
```

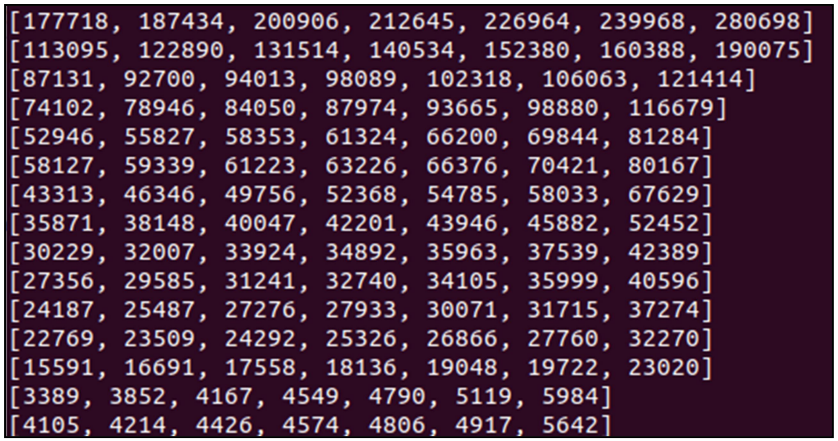
Modify the code to do these tasks:

- Remove the extra line breaks and to split the line into its component parts
- The first element is the state name. We will use that as the line label but we don't need it as a value so we can **pop** it from the list.
- Put the list into ascending order (2010 first, 2016 last)
- Convert the values (which are strings) into integers.

To do these tasks modify the loop to look like this:

```
for line in infile:
    linevalues = line.rstrip().split(",")
    linelbl = linevalues[0]
    linevalues.pop(0)
    linevalues.reverse()
    for i in range(0, len(linevalues)):
        linevalues[i] = int(linevalues[i])
    print(linevalues)
```

This time the output should look like this:



We will still need a range of years- add this to the top of the code:

```
years = [2010, 2011, 2012, 2013, 2014, 2015, 2016]
```

In the loop add the lines to plot the values on a graph:

```
plt.plot(years, linevalues, linestyle="solid", label=linelbl)
plt.legend()
```

Finally after the loop add the line to show the graph:

```
plt.show()
```

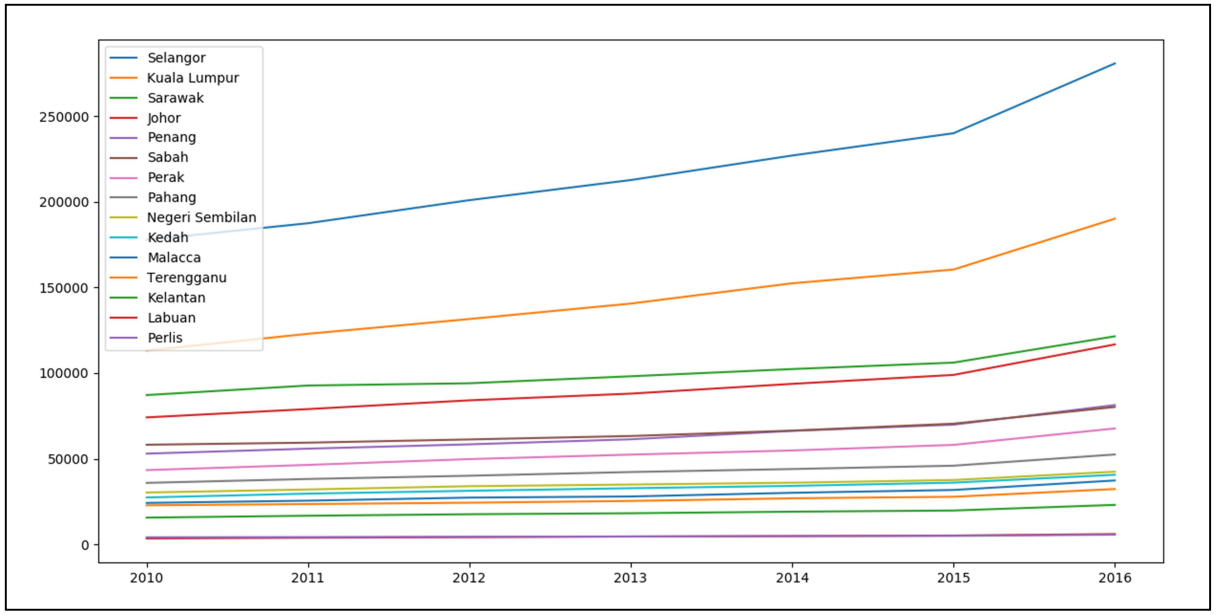
45.7 Checkpoint

At this point your script should look similar to this:

```
from matplotlib import pyplot as plt

years = [2010, 2011, 2012, 2013, 2014, 2015, 2016]
infile = open("gdp_info_copy.csv")

for line in infile:
    linevalues = line.rstrip().split(",")
    linelbl = linevalues[0]
    linevalues.pop(0)
    linevalues.reverse()
    for i in range(0, len(linevalues)):
```

```
            linevalues[i] = int(linevalues[i])
        print(linevalues)

        plt.plot(years, linevalues, linestyle="solid", label=linelbl)
        plt.legend()

plt.show()
infile.close()
```

Run the script and examine the output:



Note that the output is slightly skewed by the entry for Selangor which is in a much higher range of values than the rest of the data. Try removing that line from the file and plotting the data again to see the differences more clearly.

45.8 Scatter plots

Scatter plots are useful for visualising the relationship between two sets of data. Scatterplots require two data sets of equal size and optionally a third data set containing labels.

Create a new file called **scatterplot1.py**. Add the following code.

```
from matplotlib import pyplot as plt

widgets = [5, 7, 9, 13, 86, 103]
gromits = [23, 38, 77, 102, 165, 198]
labels = ["Alice", "Bob", "Carol", "Derek", "Ernie", "Fiona"]

plt.scatter(widgets, gromits)

for label, widget, gromit in zip(labels, widgets, gromits):
    plt.annotate(label, xy=(widget, gromit), xytext=(5,-5), textcoords="offset points")

plt.xlabel("Widgets")
plt.ylabel("Gromits")
plt.title("Widgets and Gromits")
plt.show()
```

45.9 Challenge

One of the reasons behind visualising data is to ask questions of the data. One question we might ask is : does the population of a state directly correlate with its area?

Examine the **states_info.csv** file. It contains the name of the state, the population and the area, along with some other information such as telephone area code.

Clean the data so that it contains only these three values.

Draw a scatterplot that uses population on the y axis, area on the x axis. Annotate the points with the state names.

46 Challenge : Anscombe's quartet

46.1 About

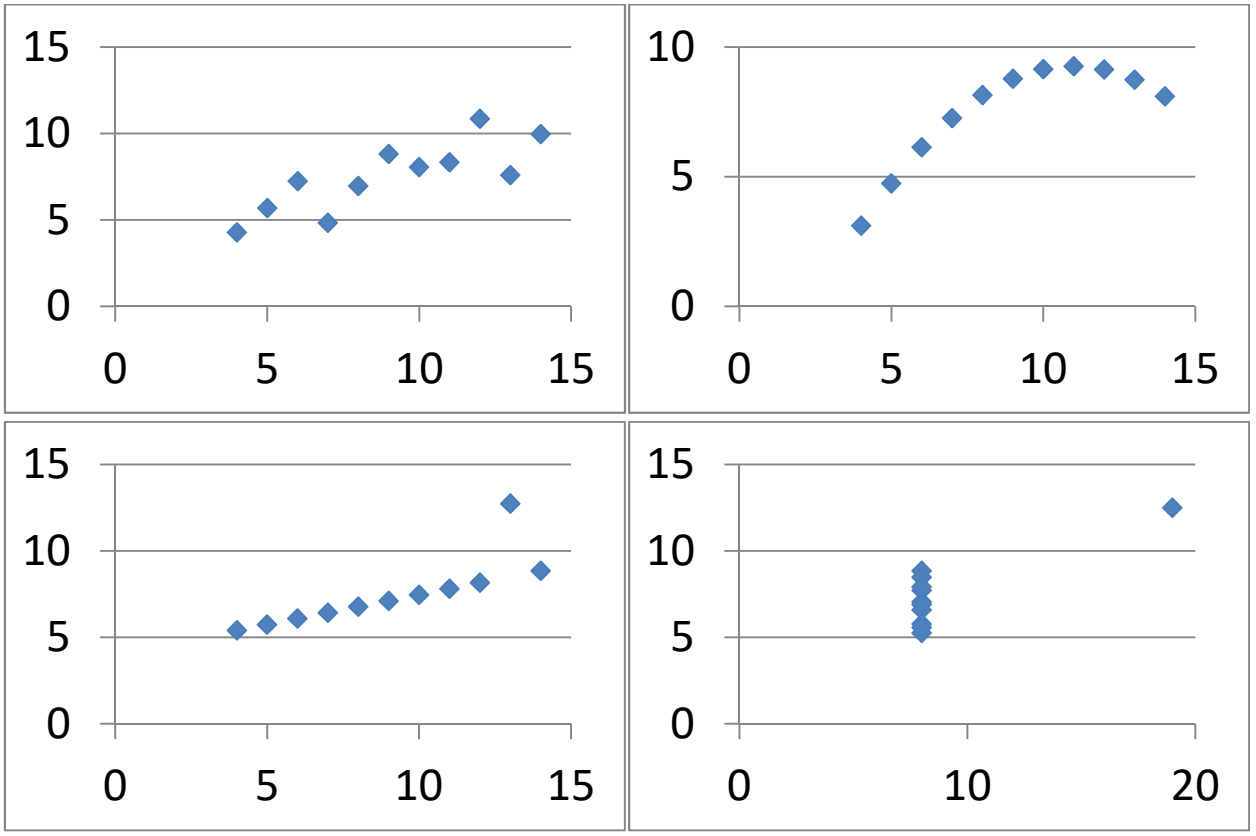
IN 1973, British statistician Frank Anscombe published an article in a journal. He wanted to show how useful it is to plot graphs to visualise data instead of relying on the numbers alone. Anscombe was also an early advocate for statistical computing and stated that computers should also be able to plot graphs as well as do calculations. As part of the article he published four sets of data that have become known as Anscombe's quartet.

46.2 The data

1		2		3		4	
x	y	x	y	x	y	x	y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

46.3 The challenge

Use MatPlotLib to plot graphs for each of the datasets in Anscombe's quartet. The outputs for the four datasets should look similar to this:



All the datasets have the same statistical properties: mean, variance, correlation coefficient, line of best fit

You would expect that they would look similar when visualised but curvature and outliers have thrown off the summary statistics.

Anscombe's point was: always plot the data- never rely on summary statistics alone.