Creating plots with MatPlotLib

MatPlotLib is a library for creating plots. There are a variety of plots available within `matplotlib.pyplot`, including:

- Line plot (e.g. plotting a function): `plot`;
- Histogram: `hist`;
- Scatter plot: `scatter`;
- Bar chart: `bar`;
- Pie chart: `pie` (though note that when representing data, a pie chart is almost never the right chart);
- 3D plot: `plot_surface`.

Scatterplot

As a basic example, let's make a scatterplot. I sampled my class for their height in cm and the time it took them to travel to university in minutes. The following code reads the data into tuples `height` and `journey` and then plots these on a scatter plot, labelling the axes.

We start by loading pyplot from matplotlib and calling it `plt`. We do this so we can access pyplot commands by putting `plt.` at the front.

```
import matplotlib.pyplot as plt
```

Now we load the data as lists `height` for height and `journey` for journey time.

```
height = [163, 162, 155, 161, 161, 172, 170, 157, 158, 167, 167, 160, 165, 160
journey = [5, 2, 3, 4, 4, 3, 60, 2, 2, 40, 3, 5, 3, 3, 2, 2, 30, 40, 30, 20, 5
```

The command to draw a scatter plot is `plt.scatter`.

```
plt.scatter(journey,height)
```

We label the axes using `plt.xlabel` for the x-axis and `plt.ylabel` for the y-axis.

```
plt.xlabel("Journey to university (mins)")
plt.ylabel("Height (cm)")
```

We can add a title to the plot

```
plt.title("Commute time vs. height")
```

Finally, if we would like to see the plot, we run

```
plt.show()
```

Customise the plot

There are lots of settings for plots, for example let's plot the amount of exercise per week against height for different eye colours. This involves loading in three sets of data for each axis, and setting the colour (`c`), `marker` and `label` for each series.

First we load pyplot and input the data - `height` for height data and `exercise` for amount of exercise per week, each for three eye colours `br` (brown), `bl` (blue) and `gr` (green).

```
import matplotlib.pyplot as plt

height_br = [163, 162, 170, 157, 158, 167, 167, 160, 174, 163, 172.5, 179.1, 1
height_bl = [155, 172, 160, 161, 161, 188, 165, 186, 160, 186, 180, 167, 175,
height_gr = [161, 161, 165, 163, 168.5, 169]

exercise_br = [1, 2, 3, 2, 1, 0, 1, 2, 3.5, 1, 0.5, 4, 3, 6, 5, 4, 9]
exercise_bl = [2, 5, 2, 1, 0, 1, 1, 5, 2, 4, 4, 3, 4, 2, 5, 4]
exercise_gr = [2, 2, 0, 1, 0, 3]
```

Now we plot data for each eye colour separately. We use `c` to control the colour, `marker` to control the shape of the point, and `label` to give the marker a name.

```
plt.scatter(height_br,exercise_br,c="brown",marker="o",label="Brown")
plt.scatter(height_bl,exercise_bl,c="blue",marker="*",label="Blue")
plt.scatter(height_gr,exercise_gr,c="green",marker="d",label="Green/hazel")
```

We label the axes and give the plot a title, as before.

```
plt.xlabel("Exercise per week (hours)")
plt.ylabel("Height (cm)")
plt.title("Exercise time vs height for different eye colours")
```

In order to see a legend (showing what the different markes mean), we run

```
plt.legend()
```

Finally, to see the plot we run

```
plt.show()
```

Multiple plots on same output

You can also create multiple plots on the same output using `subplot`. Here is a more complicated plotting example which uses `subplot` and shows off various other features.

The command `subplot` uses three parameters: the number of rows, the number of columns and the index of the plot to draw. So `plt.subplot(3,1,2)` would plot a 3 by 1 arrangement of plots and make this the 2nd one.

For example, here we draw a two by two grid of plots. There are various other features demonstrated here, including different types of plots.

Note that we can put LaTeX code in the labels. To do this, we put `r` before the string and put `$...$` around the LaTeX code, for example r"x^2" would produce x^2.

```
import matplotlib.pyplot as plt
import numpy as np

# Histogram
height = [163, 162, 155, 161, 161, 172, 170, 157, 158, 167, 167, 160, 165, 160
plt.subplot(2,2,1) # plot 1 of 4
plt.hist(height)
plt.xlabel("Height (cm)")

# Pie chart
eyes = [16, 17, 5, 1]
eyes_labels = ('Blue', 'Brown', 'Green', 'Hazel')
plt.subplot(2,2,2) # plot 2 of 4
plt.pie(eyes, labels=eyes_labels, autopct='%.1f%%')
plt.title("Eye colours")

# Bar chart
alcohol_labels = [0, 2, 4, 5, 7, 8, 10, 12, 15, 20]
alcohol_data = [19, 2, 1, 3, 1, 2, 6, 2, 2, 1]
plt.subplot(2,2,3) # plot 3 of 4
plt.bar(alcohol_labels,alcohol_data)
plt.xlabel("Units of alcohol per week")

# Simple plot
x = np.arange(0,2*np.pi,0.01) # like range(), but allows decimals
f = np.sin(3*x/2)
plt.subplot(2,2,4) # plot 4 of 4
plt.plot(x,f)
plt.xlabel(r"$x$") # labels and title using Mathtext, which is similar to LaTe
plt.ylabel(r"$f(x)$") # Mathtext labels use raw strings e.g. r"string", which
plt.title(r"Plot of $f(x)=\sin\left(\frac{3x}{2}\right)$")

plt.tight_layout() # sorts out overlapping charts, labels, etc.
plt.show() # Actually show the plots
```

Exercise

1. Plot the following data showing my computer's CPU temperature over an afternoon on a suitable chart with appropriate labelling.

Time	Temperature (C)

Time	Temperature (C)
12:00:01	17.0
12:15:01	13.1
12:30:01	0.0
12:45:01	0.0
13:00:01	0.2
13:15:01	70.4
13:30:02	43.1
13:45:01	6.9
14:00:01	28.8
14:15:01	0.0
14:30:01	1.0
14:45:01	10.4
15:00:01	19.6
15:15:01	32.4
15:30:01	12.6
15:45:01	1.0
16:00:01	25.5
16:15:01	28.6
16:30:01	18.4
16:45:01	30.9
17:00:01	12.4
17:15:01	32.9