Exercise 1: Getting started with Matplotlib

Aim: Introduce the Matplotlib interactive plotting tool

Issues covered:

- Importing Matplotlib
- Using the interactive plotting tool
- Generating some simple line graphs
- Saving a figure

If you get bored of having to close the interactive plotting window use:

```
plt.pause(1)
```

instead of:

```
plt.show()
```

1. Let's import Matplotlib and create our first plot.

a. Type the "`import matplotlib.pyplot as plt`" in the Python prompt.
b. Plot the line defined by `range(10)`.
c. Display the plot using "`plt.show()`".
d. Click the zoom button and then highlight a rectangle in the centre of the plot.
e. Click the pan button and then move around the plot (whilst zoomed in).
f. Click the back and forward buttons to move through a history of the plots you have generated.
g. Click the save button and save your plot as a PNG file.
h. Finally, close the plot using the "X" button in the top right corner.

2. Let's create a pretty plot of save chemistry data.

a. Our data set is:

```
Time (decade): 0, 1, 2, 3, 4, 5, 6.
CO2 concentration (ppm): 250, 265, 272, 260, 300, 320, 389
```

b. Create a line graph of CO_2 versus time. View the plot.
c. Re-draw the graph with a blue dashed line.
d. Add a title and axis titles to the plot.

3. Let's add a second line to our example.

a. Continuing with the above data plot, add some additional data:

```
Temp (°c):  14.1, 15.5, 16.3, 18.1, 17.3, 19.1, 20.2
```

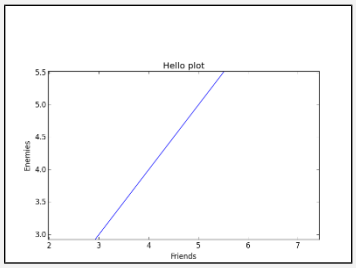
b. Save the output (using Python code) to a PDF file.

Solution 1: Getting started with Matplotlib

1.

```
>>> import matplotlib.pyplot as plt
>>> plt.plot(range(10))
>>> plt.show()
```

E.g.



2.

```
>>> times = range(7)
>>> co2 = [250, 265, 272, 260, 300, 320, 389]
>>> plt.plot(times, co2)
>>> plt.plot(times, co2, 'b--')
>>> plt.title("Concentration of CO2 versus time")
>>> plt.ylabel("[CO2]")
>>> plt.xlabel("Time (decade)")
>>> plt.show()
```

3.

```
>>> times = range(7)
>>> co2 = [250, 265, 272, 260, 300, 320, 389]
>>> temp = [14.1, 15.5, 16.3, 18.1, 17.3, 19.1, 20.2]
>>> plt.plot(times, co2, 'b--', times, temp, 'r*-')
>>> plt.savefig("co2_temp.pdf")
>>> plt.show()
```

Exercise 2: Multiple axes and multiple graphs

Aim: Introduce plotting with multiple axes and multiple graphs on the page

Issues covered:

- Plotting lines with different axes
- Using the subplot function to create multiple graphs on a single page

If you get bored of having to close the interactive plotting window use:

```
plt.pause(1)
```

instead of:

```
plt.show()
```

1. Let's re-use our previous example with different axes.

a. Import pyplot as "plt" (as before).
b. Run the line: `fig, ax1 = plt.subplots()`
c. You can now create your first plot using "ax1" instead of "plt".
d. Our data set is:

```
Time (decade): 0, 1, 2, 3, 4, 5, 6.
CO2 concentration (ppm): 250, 265, 272, 260, 300, 320, 389
```

e. Create a line graph of CO_2 versus time. Do not view the plot yet.
f. Set the y-axis label to "[CO2]" using the "ax1.set_ylabel" method.
g. Get a second axis object using: `ax2 = ax1.twinx()`
h. Plot the following temperature values to this second axis:

```
Temp (°c):  14.1, 15.5, 16.3, 18.1, 17.3, 19.1, 20.2
```

i. Set the second y-axis label to "Temp (degC)" using the "ax2.set_ylabel" method.
j. Display the plot using "plt.show()".

2. Let's draw three graphs side by side on a single page.

a. Use the "subplot" function to select the first of three plots (side-by-side).
b. Plot a line of values: `range(0, 10, 1).`
c. Select the second plot with "subplot".
d. Plot a line of values: `range(10, 0, -1).`
e. Select the third plot with "subplot".
f. Plot a line of values: `[4] * 10`
g. Display the plot using "plt.show()".

Solution 2: Multiple axes and multiple graphs

1.

```
>>> import matplotlib.pyplot as plt
>>> fig, ax1 = plt.subplots()
>>> times = range(7)
>>> co2 = [250, 265, 272, 260, 300, 320, 389]
>>> ax1.plot(times, co2, "b--")
>>> ax1.set_ylabel("[CO2]")
>>> ax2 = ax1.twinx()
>>> temp = [14.1, 15.5, 16.3, 18.1, 17.3, 19.1, 20.2]
>>> ax2.plot(times, temp, "r*-")
>>> ax2.set_ylabel("Temp (degC)")
>>> plt.show()
```

2.

```
>>> plt.subplot(1, 3, 1)
>>> x = range(0, 10, 1)
>>> plt.plot(x)
>>> plt.subplot(1, 3, 2)
>>> y = range(10, 0, -1)
>>> plt.plot(y)
>>> plt.subplot(1, 3, 3)
>>> z = [4] * 10
>>> plt.plot(z)
>>> plt.show()
```

Exercise 3: Plotting gridded data on a map

Aim: Introduce plotting gridded data using Cartopy

Issues covered:

- Importing Cartopy
- Using Cartopy for geospatial plotting
- Integration with Matplotlib

1. Let's grab some data from a NetCDF file and quickly plot it.

a. The file "`example_data/tas.nc`" contains surface air temperature differences. We can extract the data and prepare it by importing the "`example_code/map_data.py`" module.
Import everything to the local scope: `from example_code.map_data import *`

b. The following variables now exist in the local scope: `tas` (temperature), `lons` (longitudes for all grid boxes), `lats` (latitudes for all grid boxes).

c. Import cartopy and pyplot with:

```
import cartopy.crs as ccrs
import matplotlib.pyplot as plt
```

d. Create a new figure: `fig = plt.figure()`

e. Set up a projection instance with a regular lat/lon coordinate reference system:

```
proj=ccrs.PlateCarree(central_longitude=0)
```

```
f. Make a plot using this projection
mymap = fig.add_subplot(1,1,1, projection=proj)
```

g. Create a "Jet" colour map to plot the data:

```
im1 = mymap.pcolormesh(lons, lats, tas, shading='flat',
 cmap=plt.cm.jet)
```

h. Add coastlines: mymap.coastlines(resolution='110m')

i. Save the plot as "`tas1.png`".

j. Display the plot.

2. Let's jazz up the plot by creating a wrapping longitude and adding some features.

a. Follow the instructions above but this time we'll add in some features.

b. wrap the data and longitudes to remove the missing data line down the middle of the plot.

```
import cartopy.util as cartopy_util
tas, lons = cartopy_util.add_cyclic_point(tas, lons)
```

c. Add a title: "Change in Surface Air Temperature from MOHC HadGEM2-ES"
Hint use: mymap.set_title("title")

d. Add some vertical and horizontal grid lines:
gridlines = mymap.gridlines(crs=ccrs.PlateCarree()

e. Add a colour bar after generating the colour map "im1", with:
cb = fig.colorbar(im1, orientation="horizontal")

f. Save the plot as "tas2.png". Compare the plot with that produced above.

g. Display the plot.

Solution 3: Plotting gridded data on a map

1.

```
from example_code.map_data import *
import cartopy.crs as ccrs
import matplotlib.pyplot as plt

fig = plt.figure()
proj=ccrs.PlateCarree(central_longitude=0)
mymap = fig.add_subplot(1,1,1, projection=proj)

im1 = mymap.pcolormesh(lons, lats, tas, shading='flat', cmap=plt.cm.jet)

mymap.coastlines(resolution='110m')

plt.savefig("tas1.png")
plt.show()
```

2.

```
from example_code.map_data import *
import cartopy.crs as ccrs
import matplotlib.pyplot as plt
import cartopy.util as cartopy_util
fig = plt.figure()
proj=ccrs.PlateCarree(central_longitude=0)
mymap = fig.add_subplot(1,1,1, projection=proj)

tas, lons = cartopy_util.add_cyclic_point(tas, lons)
im1 = mymap.pcolormesh(lons, lats, tas, shading='flat', cmap=plt.cm.jet)

mymap.coastlines(resolution='110m')
mymap.set_title('Change in Surface Air Temperature from MOHC HadGEM2-ES')

gridlines = mymap.gridlines(crs=ccrs.PlateCarree())

cb = fig.colorbar(im1, orientation="horizontal")

plt.savefig("tas2.png")
plt.show()
```

The plots should look like this: