A Quickstart Guide to Using Numpy and Matplotlib

- The capabilities of Python can be extended with many different packages and modules
- Numpy is a great mathematical package, which contains lots of definitions for many different mathematical functions. It also introduces its own array type, which behaves like normal Python lists, but is also much more powerful
- Matplotlib is a library for creating visualisations such as graphs and plots and much more
- Both these libraries can seem quite daunting to use when first starting out, so the aim of this guide is to give a quick overview of basic usage that will ease you in. For more details see the Numpy tutorial and the Matplotlib tutorials

Getting started with Numpy and Matplotlib

- To use Numpy and Matplotlib, they first need to be installed
- This can be done using the `pip` tool from the command line, i.e. on Windows cmd:

```
python -m pip install --user numpy matplotlib
```

- To then use Numpy and Matplotlib within a Python script, they need to be imported:

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

- The `as` keyword used above means that in the remainder of the script, the Numpy module can simply be referred to as `np`, and the pyplot module (from the Matplotlib library) as `plt`
  - e.g. instead of using `numpy.pi`, to get the number π, `np.pi` can be used

Creating and using Numpy arrays

- Numpy defines a new type, the `ndarray`
- There are a number of ways to create these arrays, but the most useful way for our purposes uses the `linspace` function provided by Numpy:

```
1    import numpy as np
2    t = np.linspace(0,5,51)
3    x = 2 * t
```

- The above program simply creates a 1-dimensional array (stored in variable `t`), of size 51, of linearly spaced numbers from 0 to 5,
  - i.e. `t = [0, 0.1, 0.2, ... , 4.8, 4.9, 5]`
- The third line performs an operation on `t`, and stores the result in the variable `x`
- Operations on `ndarrays` are performed elemntwise,
  - i.e. `x = [0, 0.2, 0.4, ... , 9.6, 9.8, 10]`

Using arrays to plot graphs

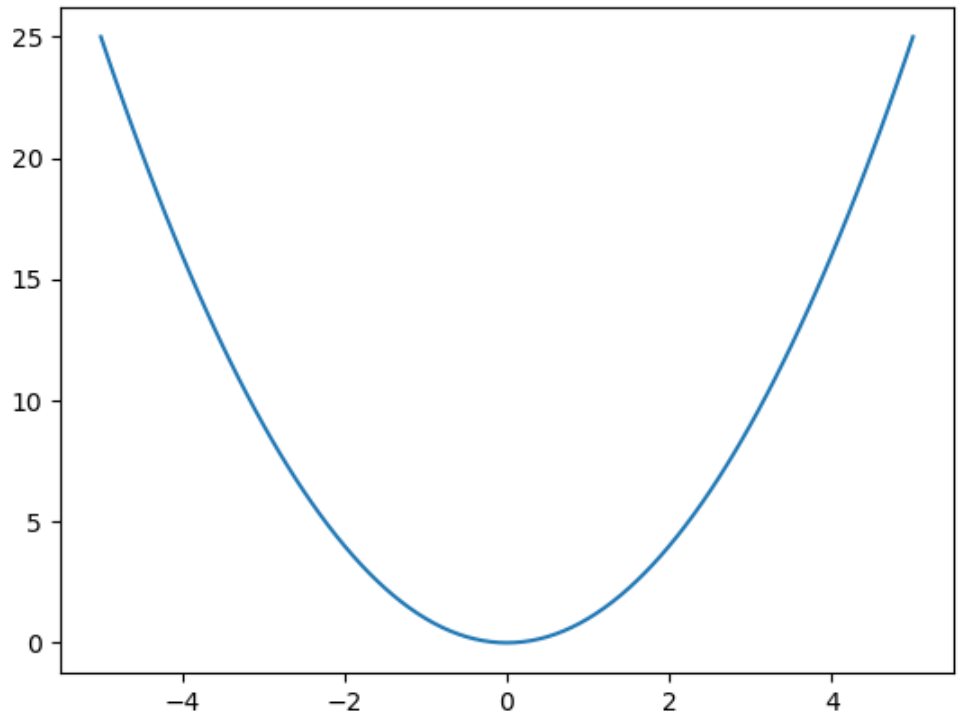
- To plot a graph, say of $y = x^2$:

```
1    import numpy as np
2    import matplotlib.pyplot as plt
3
4    x = np.linspace(-5,5,101)
5    y = x**2
6
7    plt.plot(x, y)
8    plt.show()
```

- Import numpy and pyplot as before
- Create an array, `x`, of linearly spaced values from -5 to 5, i.e. the 'domain of the function'
  - 101 points have been chosen, which is more than enough for this example. Care must always be taken when choosing the number of data points to ensure a smooth plot
- Calculate y
- To plot the data, the function `plt.plot` is used
- The graph is then displayed to the user using `plt.show`

Using arrays to plot graphs cont.

- The program on the previous slide yields the following graph:



- Before entering `plt.show()` the graph can be further customised, e.g. adding axis labels, legends etc. See the Matplotlib documentation for more details (see also the Numpy documentation for more details on what Numpy can do)