Week 4
Maths

Python Anywhere

If you are unable to install NumPy and Matplotlib, you can use the cloud Python service Python Anywhere to run your code. To use it, sign up for a free account at `http://www.pythonanywhere.com`.

`numpy`

The `numpy` module allows you to make and use vectors within Python. It is common to rename `numpy` as `np` when you inport it.

```
import numpy as np
```

The command `np.array` will create a vector.

```
import numpy as np
a = np.array([1,0,0])
print(a)
```

```
[1 0 0]
```

You can add and subtract vectors, multiply them by scalars, and take their dot and cross products:

```
import numpy as np
a = np.array([1,0,0])
b = np.array([0,1,0])
print(a+b)
print(a-b)
print(3*a)
print(np.dot(a,b))
print(a.dot(b))
print(np.cross(a,b))
```

```
[1 1 0]
[ 1 -1  0]
[3 0 0]
0
0
[0 0 1]
```

You can also create matrices by making an array from a list of lists:

```
import numpy as np
a = np.array([[1,0],[1,1]])
b = np.array([[4,0],[0,2]])
v = np.array([5,1])
print(a+b)
print(a.dot(b))
print(a.dot(v))
print(a.T)
```

```
[[5 0]
 [1 3]]
[[4 0]
 [4 2]]
[5 6]
[[1 1]
 [0 1]]
```

WARNING

For matrices and vectors, the * operator will perform component-wise multiplication, not matrix-matrix or matrix-vector multiplication.

You can use **numpy** to find the inverse of a matrix:

```
import numpy as np
a = np.array([[1,0],[1,1]])
print(np.linalg.inv(a))
```

```
[[ 1.  0.]
 [-1.  1.]]
```

You can use **numpy** to find the determinant of a matrix:

```
import numpy as np
a = np.array([[1,2],[1,1]])
print(np.linalg.det(a))
```

```
-1.0
```

You can use **numpy** to find the eigenvalues and eigenvectors of a matrix:

```
import numpy as np
a = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(np.linalg.eig(a))
```

```
(array([  1.61168440e+01,  -1.11684397e+00,
         -1.30367773e-15]),
array([[-0.23197069, -0.78583024,  0.40824829],
       [-0.52532209, -0.08675134, -0.81649658],
       [-0.8186735 ,  0.61232756,  0.40824829]]))
```

numpy contains some helpful functions for making vectors and matrices. **np.zeros** will make a vector or matrix full of zeros. **np.ones** will make a vector or matrix full of ones. **np.eye** will make an identity matrix. **np.random.rand** will make a random vector or matrix.

```
import numpy as np
print(np.zeros(3))
print(np.zeros([3,3]))

print(np.ones(3))
print(np.ones([3,3]))

print(np.eye(4))

print(np.random.rand(5))
print(np.random.rand(4,2))
```

```
[ 0.   0.   0.]
[[ 0.   0.   0.]
 [ 0.   0.   0.]
 [ 0.   0.   0.]]
[ 1.   1.   1.]
[[ 1.   1.   1.]
 [ 1.   1.   1.]
 [ 1.   1.   1.]]
[[ 1.   0.   0.   0.]
 [ 0.   1.   0.   0.]
 [ 0.   0.   1.   0.]
 [ 0.   0.   0.   1.]]
[ 0.11255399  0.7742422   0.6236629   0.73659403
  0.49337274]
[[ 0.10133038  0.17647796]
 [ 0.28261989  0.10458666]
 [ 0.25989634  0.13737324]
 [ 0.61591857  0.27501992]]
```

1. Use `numpy` to find the inverse of the matrix

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 2 & 8 \\ 1 & 0 & 1 \end{pmatrix}$$

2. Use `numpy` to find $\mathbf{x} \in \mathbb{R}^3$ such that

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 2 & 8 \\ 1 & 0 & 1 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 2 \\ 4 \\ 5 \end{pmatrix}$$

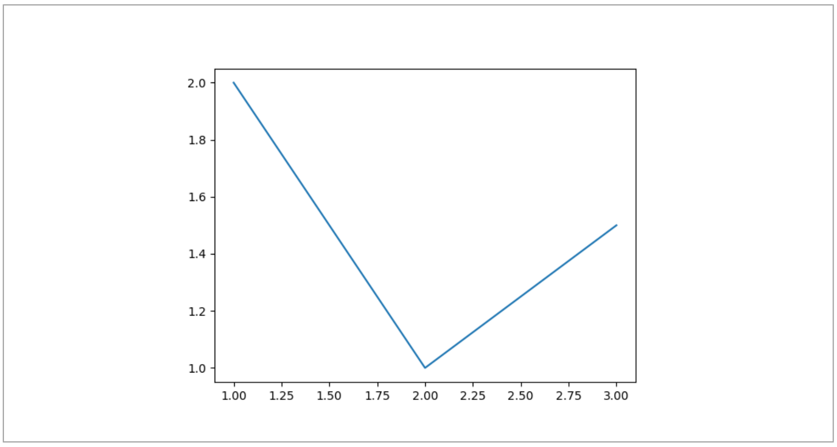
matplotlib

The module `matplotlib` can plot graphs. It is common to import the submodule `matplotlib.pylab` and rename it as `plt`.

```
import matplotlib.pylab as plt
```

You can then plot a graph using `plt.plot` and show it using `plt.show`. The first input of `plt.plot` is a list of the x-coordinate of each point; the second input is the y-coordinate of each point.

```
import matplotlib.pylab as plt
plt.plot([1,2,3],[2,1,1.5])
plt.show()
```



The command `plt.savefig` can be used to save a plot to a file:

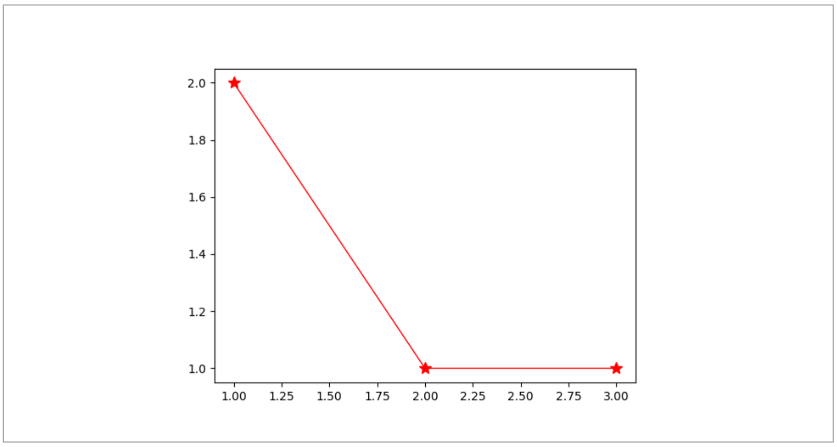
```
import matplotlib.pylab as plt
plt.plot([1,2,3],[2,1,1.5])
plt.savefig("plot.png")
```

WARNING

If you are running your code on Python Anywhere, `plt.show()` will not do anything. Instead, you should use `plt.savefig("filename.png")` to save the plot, then open the file.

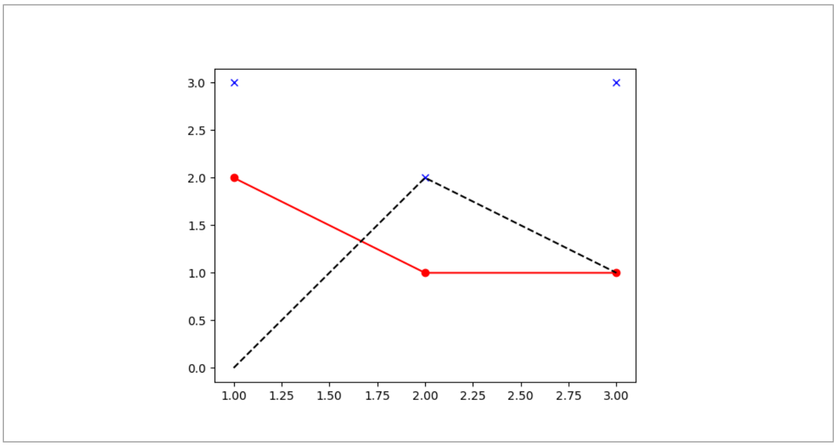
You can pass extra parameters into `plt.plot` to style the data you are plotting:

```
import matplotlib.pylab as plt
plt.plot([1,2,3],[2,1,1],linewidth=1,color="red",marker="*",markersize=10)
plt.show()
```



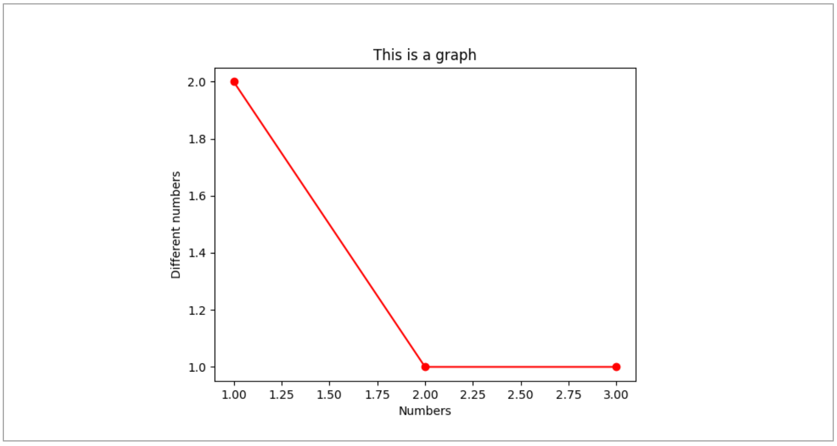
You can control many of the styling options with less typing by using a formatting string. For example, the string `"ro-"` will plot a red (`r`) line (`-`) with circles (`o`) marking each point.

```
import matplotlib.pylab as plt
plt.plot([1,2,3],[2,1,1],"ro-")
plt.plot([1,2,3],[3,2,3],"bx")
plt.plot([1,2,3],[0,2,1],"k--")
plt.show()
```



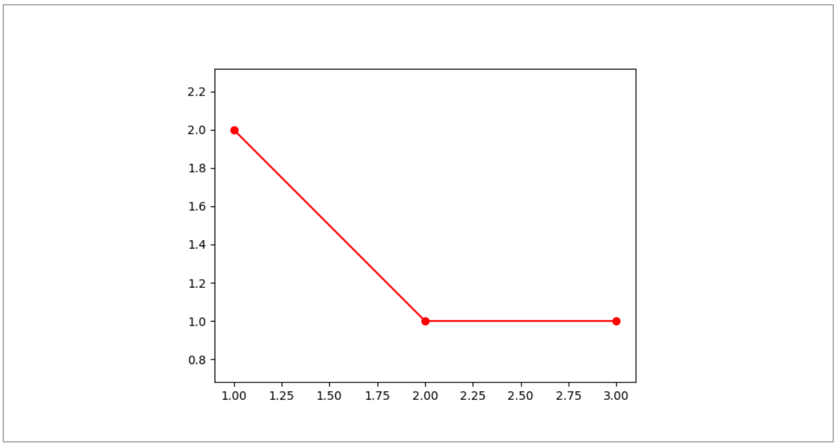
There are many more commands in `matplotlib` that you can use to style plots. The meaning of the following should be obvious.

```
import matplotlib.pylab as plt
plt.plot([1,2,3],[2,1,1],"ro-")
plt.xlabel("Numbers")
plt.ylabel("Different numbers")
plt.title("This is a graph")
plt.show()
```



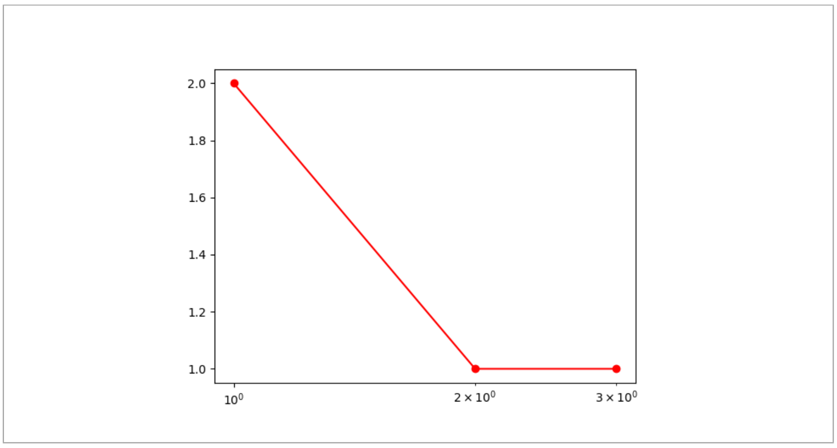
There are also some less obvious commands. `plt.axis("equal")` will make the axes equal aspect.

```
import matplotlib.pylab as plt
plt.plot([1,2,3],[2,1,1],"ro-")
plt.axis("equal")
plt.show()
```



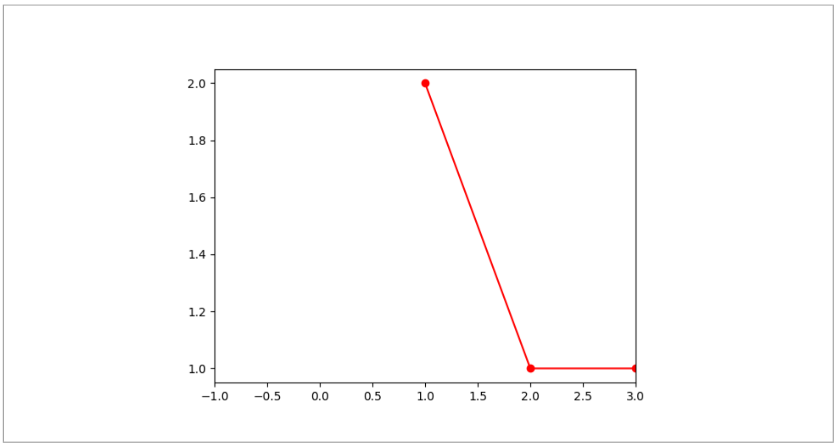
`plt.xscale("log")` and `plt.yscale("log")` will make the axes logarithmically scaled.

```
import matplotlib.pylab as plt
plt.plot([1,2,3],[2,1,1],"ro-")
plt.xscale("log")
plt.show()
```



`plt.xlim` and `plt.ylim` will set the limits of the axes.

```
import matplotlib.pylab as plt
plt.plot([1,2,3],[2,1,1],"ro-")
plt.xlim([-1,3])
plt.show()
```



If you want to plot multiple graphs in the same file, you should use `plt.clf()` to clear the current plot before starting the next plot.

`matplotlib` can do a lot more than we have discussed here, including plotting bar graphs (`plt.bar`), making contour plots (`plt.contour` and `plt.contourf`), and 3d plotting. I suggest Googling for example of how to do these as and when you need them.

3. Use `matplotlib` and `random` to make a plot of 100 random numbers against 100 different random numbers.

4. Use `matplotlib` to draw a plot of $y = x^2$ for x between -5 and 5.

Putting it all together

Using all the Python commands you have learned in the four weeks so far, do the following questions.

5. Use `matplotlib` to draw a plot of $y = \sin x$ for x between -5 and 5.

6. Use `numpy` to calculate the determinants of the following matrices:

$$\begin{pmatrix}4 & 0\\0 & 2\end{pmatrix} \qquad \begin{pmatrix}1 & 0\\1 & 1\end{pmatrix} \qquad \begin{pmatrix}1 & 1\\1 & 1\end{pmatrix} \qquad \begin{pmatrix}1 & 2 & 1\\-1 & 2 & 0\\0 & 4 & 1\end{pmatrix} \qquad \begin{pmatrix}1 & 0 & 1 & 0\\0 & 1 & 1 & 1\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{pmatrix}$$

Invert the non-singular matrices.

7. Use `matplotlib` to draw a plot of $y = \frac{1}{x}$ for x between -5 and 5.

8. Use `numpy` to find the Eigenvectors of the following matrices.

$$\begin{pmatrix}4 & 0\\0 & 2\end{pmatrix} \qquad \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix} \qquad \begin{pmatrix}0 & 1\\-1 & 0\end{pmatrix} \qquad \begin{pmatrix}1 & 0 & 1 & 0\\0 & 1 & 1 & 1\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{pmatrix}$$

9. Given two vectors, $\mathbf{x}$ and $\mathbf{y}$, you can find another vector $\mathbf{z}$ using the formula

$$\mathbf{z} = \mathbf{y} - \left(\frac{\mathbf{y}\cdot\mathbf{x}}{\mathbf{x}\cdot\mathbf{x}}\right)\mathbf{x}.$$

The vectors $\mathbf{z}$ and $\mathbf{x}$ are perpendicular.

Write a Python script that generates two random vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$ then calculates $\mathbf{z}$ using the above formula. Show that $\mathbf{x}\cdot\mathbf{z} = 0$ to confirm that $\mathbf{x}$ and $\mathbf{z}$ are perpendicular.

10. Write a function that takes a function as an input, and makes a plot of this function for x between -4 and 4.

Use your function to plot $y = \sin x$, $y = \cos x$ and $y = \tan x$.

11. For $x = -2, -1.9, -1.8, -1.7, ..., 2$, use `numpy` calculate the eigenvalues of the matrix

$$\begin{pmatrix} 1 & x \\ x & 1 \end{pmatrix}.$$

Use `matplotlib` to make a plot showing x (on the x-axis) against the largest eigenvalue of this matrix (on the y-axis). On the same axes, but in a different colour, plot x against the smallest eigenvalue of this matrix.

Assessed question

When you have completed this question, you should show your code to one of the helpers in your class.

12. For $x = -2, -1.9, -1.8, -1.7, ..., 2$, use `numpy` calculate the determinant of the matrix

$$\begin{pmatrix} x & 1 & 0 \\ 0 & x & 1 \\ 1 & 0 & x \end{pmatrix}.$$

Use `matplotlib` to make a plot showing x (on the x-axis) against the determinant of this matrix (on the y-axis).