FITS

The files which are processed from observatories

Opening a FITS file in Astropy returns a HDU (*Header/Data Unit*) list. Each HDU stores headers and (optionally) image data.

The header contains metadata about the HDU object, e.g. its dimensions and data type. Every HDU can contain image data. The first HDU is called the *primary HDU*.

If we want to access individual HDUs, we can index the HDU list object returned by `fits.open`. The image data can be accessed using the `data` attribute:

```
from astropy.io import fits

hdulist = fits.open('image0.fits')
data = hdulist[0].data

print(data.shape)
```

The image data is conveniently stored in a NumPy array, so we can operate on it directly. This example prints the dimensions of the image in the primary HDU.

You often want to visualise the image data stored in FITS files. We can do this using the plotting library matplotlib.

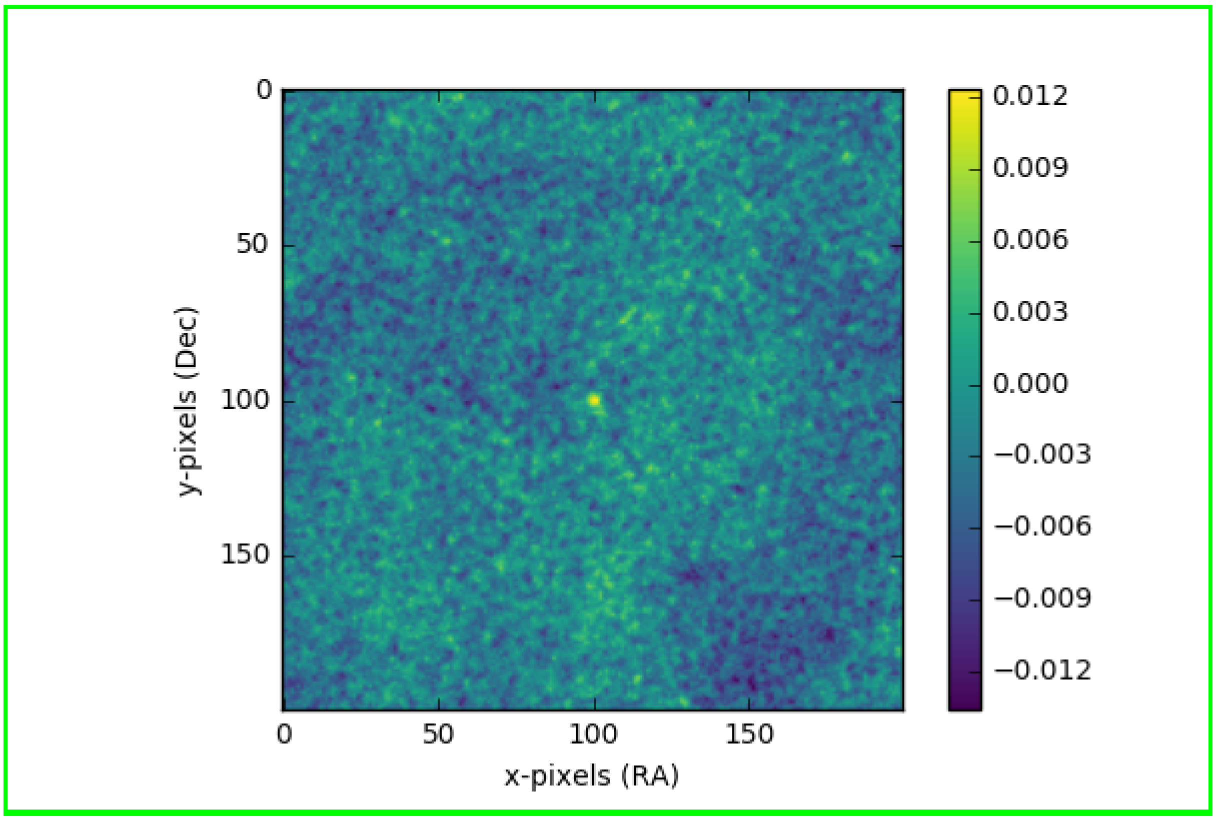
This example creates a 2D plot from the previous FITS image:

```
from astropy.io import fits
import matplotlib.pyplot as plt

hdulist = fits.open('image0.fits')
data = hdulist[0].data

# Plot the 2D array
plt.imshow(data, cmap=plt.cm.viridis)
plt.xlabel('x-pixels (RA)')
plt.ylabel('y-pixels (Dec)')
plt.colorbar()
plt.show()
```

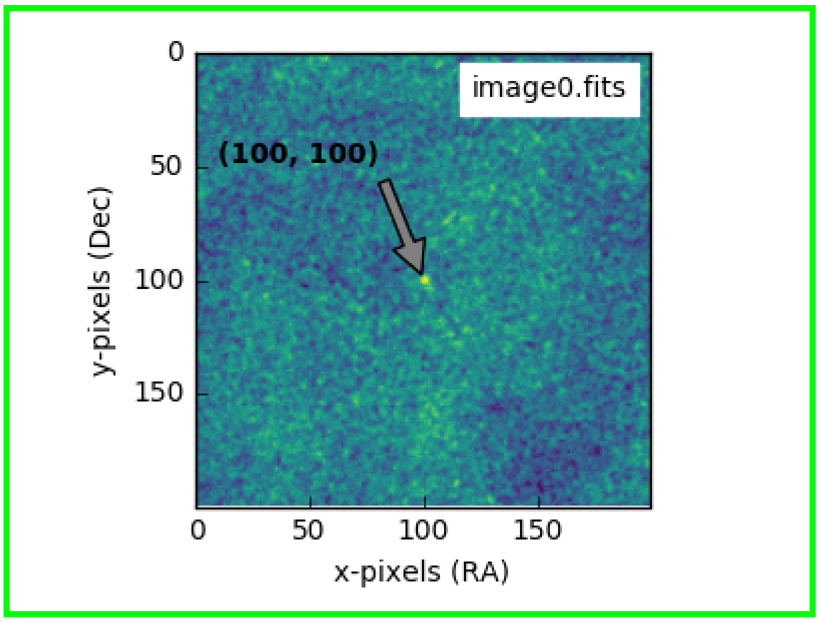
The code above produces the following image:



Write a **`load_fits`** function that loads in a FITS file and finds the position of the brightest pixel (i.e. the maximum value) in its image data. To make this function work for arbitrary files, pass the name of the FITS file as an argument to the function. Using the file **`image0.fits`** from the previous examples, your program should work like this:

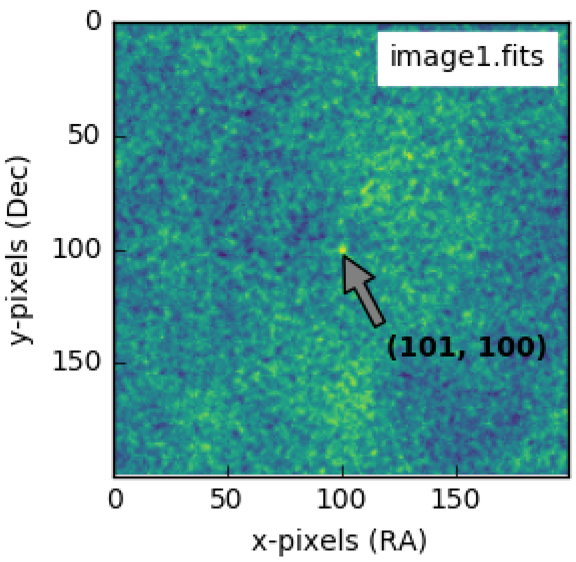
```
>>> load_fits('image0.fits')
(100, 100)
```

The brightest pixel in this image is exactly in the centre of the array, as you can check visually by plotting it:



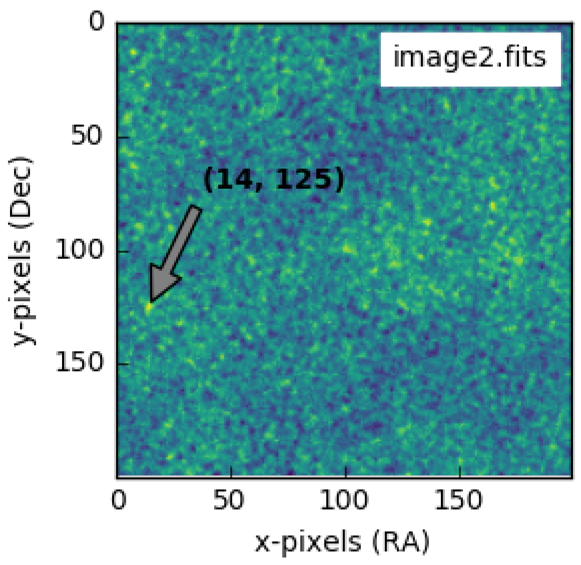
For the file **`image1.fits`**, the maximum is shifted by 1 pixel to the left and your function should produce the following output:

```
>>> load_fits('image1.fits')
(101, 100)
```



The FITS files for this example all contain image data of the same pulsar but that does not necessarily mean that it is visible in all of these images. When you take a look at the file **`image2.fits`** for instance, your function should be able to tell you that the brightest pixel is somewhere completely different:

```
>>> load_fits('image2.fits')
(14, 125)
```



If you want to plot the arrays out for yourself, keep in mind that any additional code in your script needs to go into a `if __name__ == '__main__'` statement to be safely ignored by the automarker.

To solve this problem, you will need to use the Astropy module to read in the FITS files and to extract the image data, as we've seen on the two previous slides.

Then, to find the brightest pixel of the image we are looking for the largest value in the 2D array. The **`argmax`** function from numpy provides precisely this functionality: it searches for the largest value in the array and returns its position. However, if you've printed out the result of **`argmax`** on its own you would have found that it does not return a tuple of x and y coordinates but just a single index. Why is that? This function works on a flattened (or ravelled) array, i.e. the array gets converted to a 1D array internally before the maximum is found. To revert this, or to "unravel" the result, we can call the function **`unravel_index`** and pass it the dimension of the initial data array as second argument.

Sample solution

```
from astropy.io import fits
import numpy as np

def load_fits(filename):
  hdulist = fits.open(filename)
  data = hdulist[0].data

  arg_max = np.argmax(data)
  max_pos = np.unravel_index(arg_max, data.shape)

  return max_pos
```