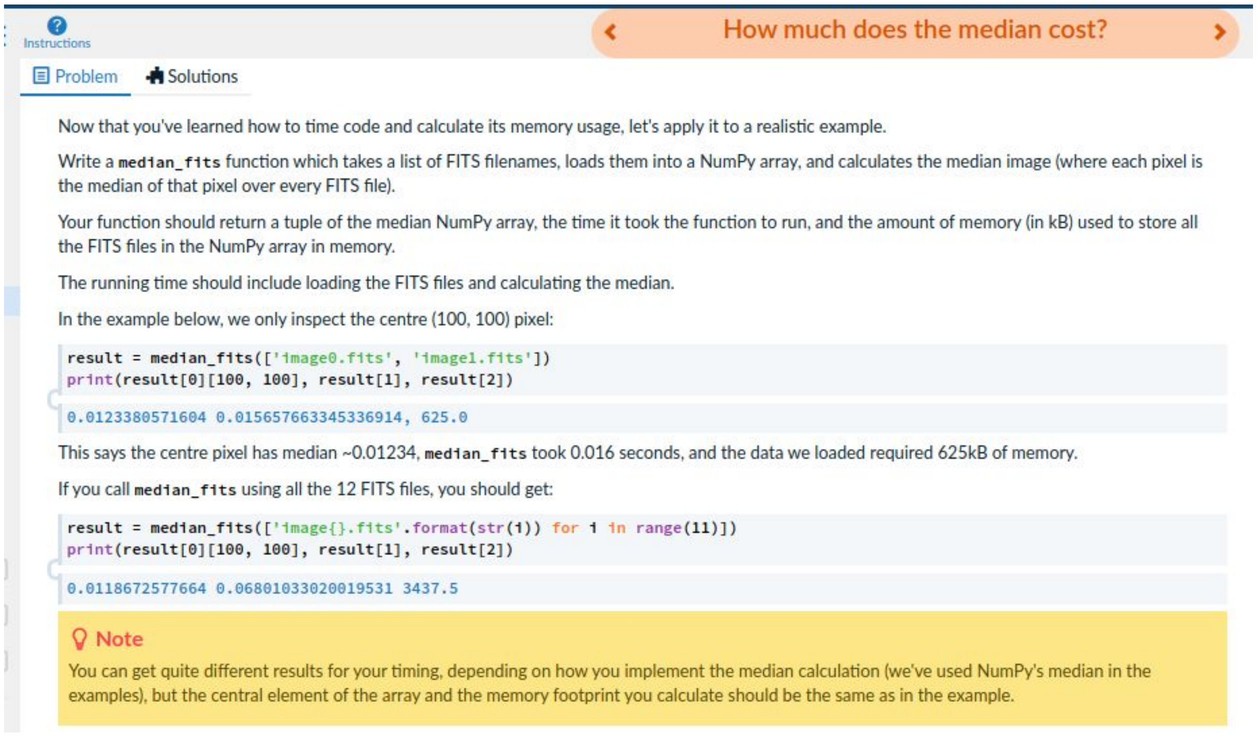

How much does the median cost?

Now that you've learned how to time code and calculate its memory usage, let's apply it to a realistic example.

Write a `median_fits` function which takes a list of FITS filenames, loads them into a NumPy array, and calculates the median image (where each pixel is the median of that pixel over every FITS file).

Your function should return a tuple of the median NumPy array, the time it took the function to run, and the amount of memory (in kB) used to store all the FITS files in the NumPy array in memory.

The running time should include loading the FITS files and calculating the median.

In the example below, we only inspect the centre (100, 100) pixel:

```
result = median_fits(['image0.fits', 'image1.fits'])
print(result[0][100, 100], result[1], result[2])
```

```
0.0123380571604 0.015657663345336914, 625.0
```

This says the centre pixel has median ~0.01234, `median_fits` took 0.016 seconds, and the data we loaded required 625kB of memory.

If you call `median_fits` using all the 12 FITS files, you should get:

```
result = median_fits(['image{}.fits'.format(str(i)) for i in range(11)])
print(result[0][100, 100], result[1], result[2])
```

```
0.0118672577664 0.06801033020019531 3437.5
```

Note

You can get quite different results for your timing, depending on how you implement the median calculation (we've used NumPy's median in the examples), but the central element of the array and the memory footprint you calculate should be the same as in the example.