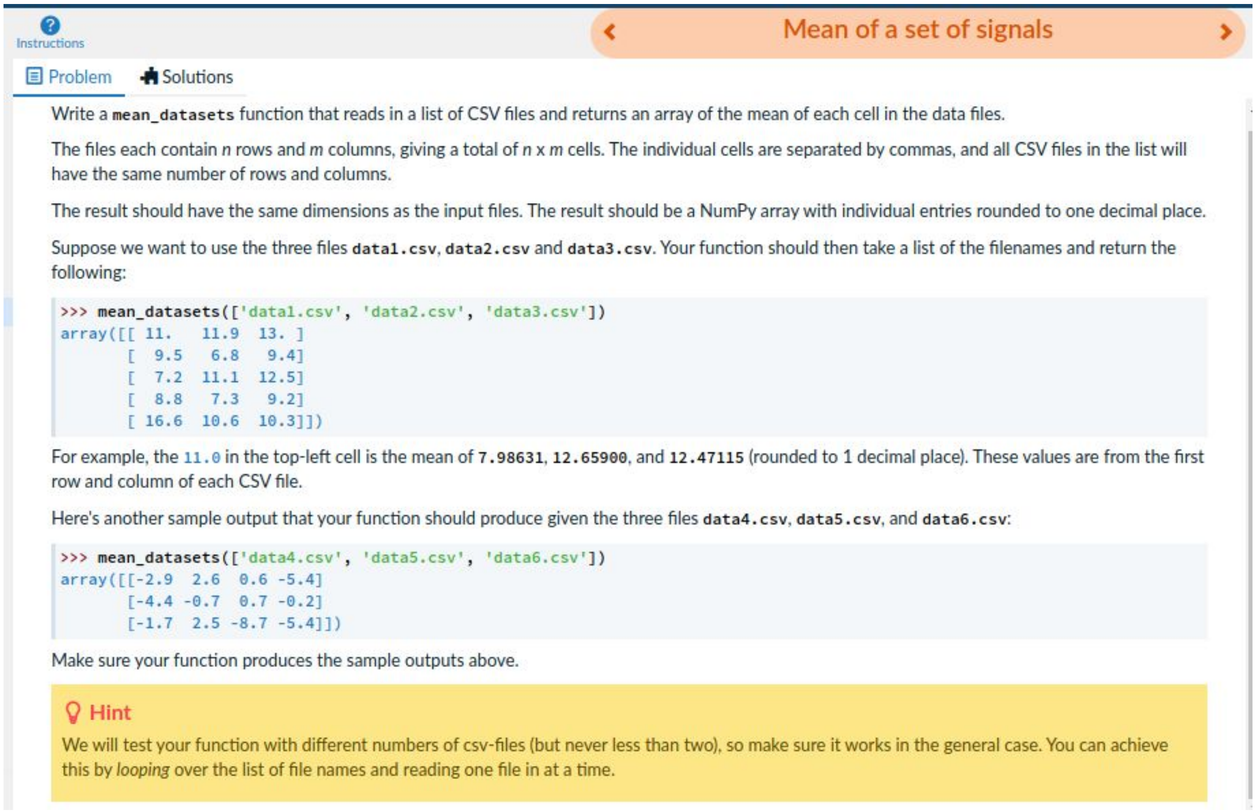

Mean of a set of signals

Write a `mean_datasets` function that reads in a list of CSV files and returns an array of the mean of each cell in the data files.

The files each contain *n* rows and *m* columns, giving a total of *n* x *m* cells. The individual cells are separated by commas, and all CSV files in the list will have the same number of rows and columns.

The result should have the same dimensions as the input files. The result should be a NumPy array with individual entries rounded to one decimal place.

Suppose we want to use the three files `data1.csv`, `data2.csv` and `data3.csv`. Your function should then take a list of the filenames and return the following:

```
>>> mean_datasets(['data1.csv', 'data2.csv', 'data3.csv'])
array([[ 11.   11.9  13. ]
       [  9.5   6.8   9.4]
       [  7.2  11.1  12.5]
       [  8.8   7.3   9.2]
       [ 16.6  10.6  10.3]])
```

For example, the `11.0` in the top-left cell is the mean of `7.98631`, `12.65900`, and `12.47115` (rounded to 1 decimal place). These values are from the first row and column of each CSV file.

Here's another sample output that your function should produce given the three files `data4.csv`, `data5.csv`, and `data6.csv`:

```
>>> mean_datasets(['data4.csv', 'data5.csv', 'data6.csv'])
array([[-2.9  2.6  0.6 -5.4]
       [-4.4 -0.7  0.7 -0.2]
       [-1.7  2.5 -8.7 -5.4]])
```

Make sure your function produces the sample outputs above.

Hint

We will test your function with different numbers of csv-files (but never less than two), so make sure it works in the general case. You can achieve this by *looping* over the list of file names and reading one file in at a time.