1 Reading and writing CSV files using NumPy and Pandas

Here we will load a CSV called iris.csv. This is stored in the same directory as the Python code.

As a general rule, using the Pandas import method is a little more 'forgiving', so if you have trouble reading directly into a NumPy array, try loading in a Pandas dataframe and then converting to a NumPy array.

1.1 Reading a csv file into a NumPy array

NumPy's *loadtxt* method reads delimited text. We specify the separator as a comma. The data we are loading also has a text header, so we use *skiprows=1* to skip the header row, which would cause problems for NumPy.

```
import numpy as np

my_array = np.loadtxt('iris_numbers.csv',delimiter=",", skiprows=1)

print (my_array[0:5,:]) # first 5 rows

OUT:

[[5.1 3.5 1.4 0.2 1. ]
 [4.9 3.  1.4 0.2 1. ]
 [4.7 3.2 1.3 0.2 1. ]
 [4.6 3.1 1.5 0.2 1. ]
 [5.  3.6 1.4 0.2 1. ]]
```

1.2 Saving a NumPy array as a csv file

We use the *savetxt* method to save to a csv.

```
np.savetxt("saved_numpy_data.csv", my_array, delimiter=",")
```

1.3 Reading a csv file into a Pandas dataframe

The *read_csv* will read a CSV into Pandas. This import assumes that there is a header row. If there is no header row, then the argument *header = None* should be used as part of the command. Notice that a new index column is created.

```
import pandas as pd

df = pd.read_csv('iris.csv')

print (df.head(5)) #  First 5 rows

OUT:

   sepal.length  sepal.width  petal.length  petal.width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa
```

1.4 Saving a Pandas dataframe to a CSV file

The *to_csv* will save a dataframe to a CSV. By default column names are saved as a header, and the index column is saved. If you wish not to save either of those use *header=True* and/or *index=True* in the command. For example, in the command below we save the dataframe with headers, but not with the index column.

```
df.to_csv('my_pandas_dataframe.csv', index=False)
```