Exercise 5

Simple data processing with Python and Pandas

Prior Knowledge
Unix Command Line Shell
Simple Python

Learning Objectives
First steps with Pandas
Understand the Jupyter Notebook model

Software Requirements
(see separate document for installation of these)

- Python 3.8.x
- Jupyter notebooks

Downloading our sample data

1. Let's make a directory to store our code.

```
mkdir ~/hyg
cd ~/hyg
```

2. Now let's download some star data.
This data is found at:
http://www.astronexus.com/hyg

You can either download the data by going to that website and finding HYG3.0 and downloading into the newly created directory, or you can use a command line and type:

```
wget
http://www.astronexus.com/files/downloads/hygdata_v3.csv.gz
```

3. Either way that you downloaded it, you now need to uncompress it:

```
gunzip hygdata_v3.csv.gz
```

4. Check it's the right size:
```
ls -l ~/hyg
```

You should see:
```
-rw-rw-r-- 1 oxclo oxclo 33449663 Apr 21  2015 hygdata_v3.csv
```

5. The VM has a “notebook” system called Jupyter configured by default. The result is that instead of starting a command line repl[1], there is a web based editor/evaluator launched instead.

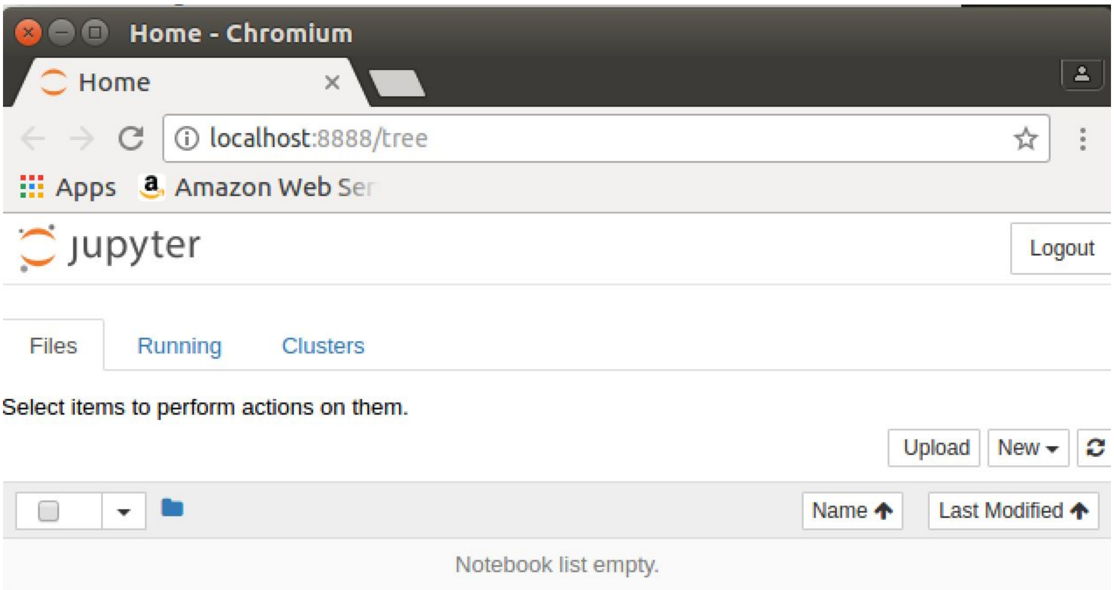
6. To start this, type (from the same command line that is in the hyg directory):

   `jupyter notebook`

7. In the command-line you will see

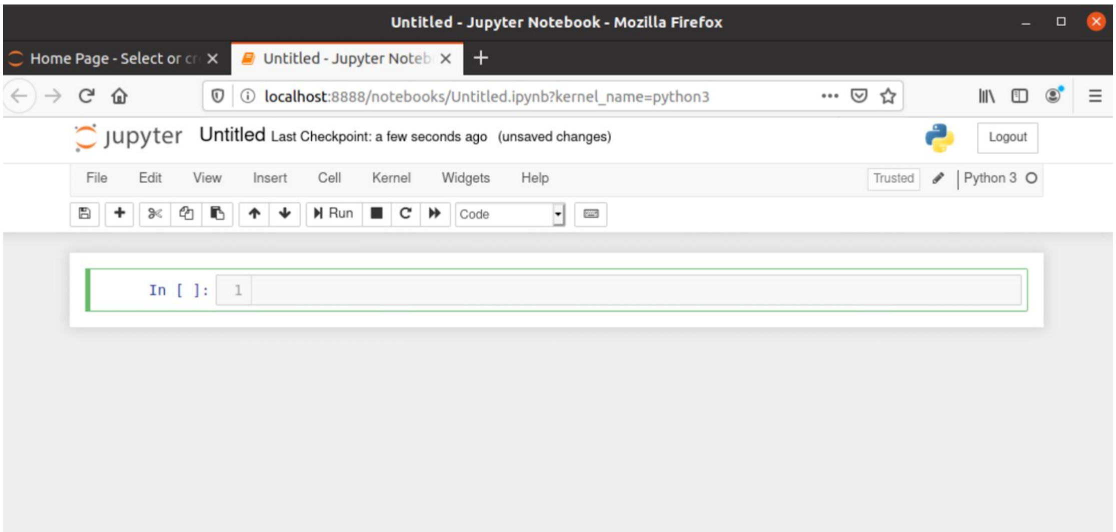
```
[I 13:53:23.865 NotebookApp] Serving notebooks from local directory: /home/oxclo/pse
[I 13:53:23.866 NotebookApp] 0 active kernels
[I 13:53:23.866 NotebookApp] The Jupyter Notebook is running at:
http://localhost:8888/?token=fd655aab32ed4840ceb47b8b7392b1243a27f56350888a91
[I 13:53:23.866 NotebookApp] Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).
[C 13:53:23.868 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=fd655aab32ed4840ceb47b8b7392b1243a27f56350888a91
```

8. And then a browser window will pop up.



[1] Read Eval Print Loop

9. Use the **New** button to create a new Python3 notebook:



10. Click on the name of the notebook (currently "Untitled") and rename it to **hyg**

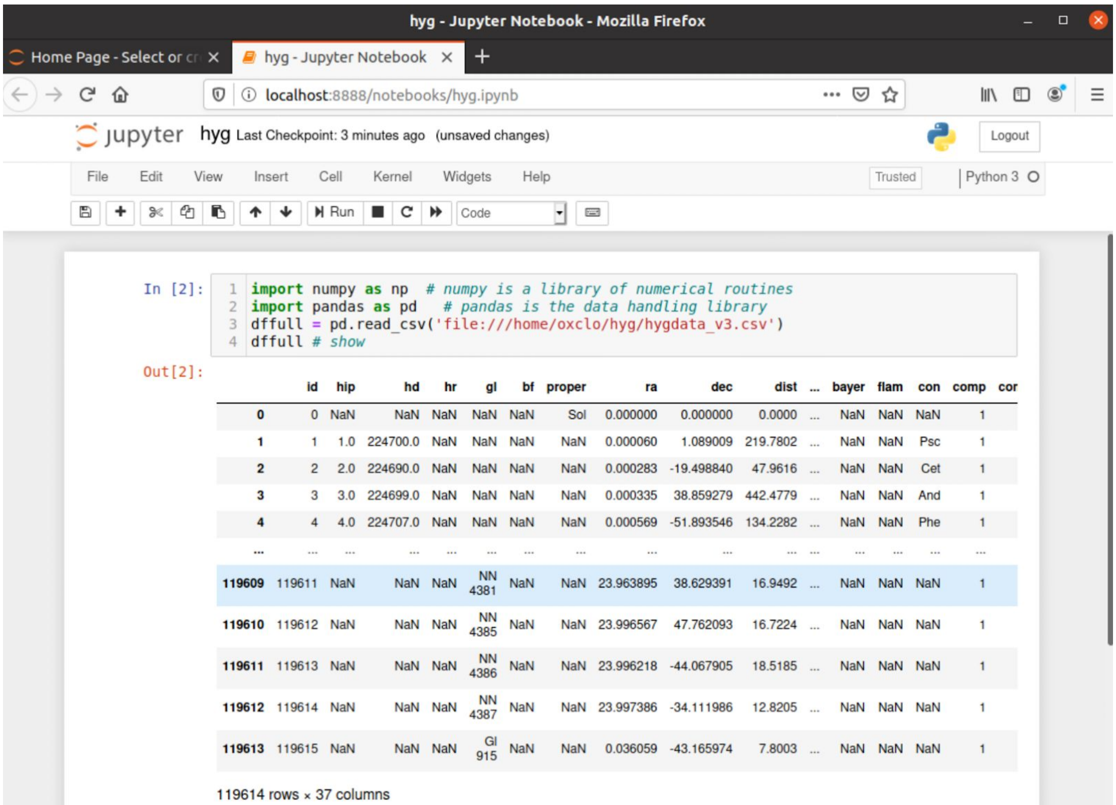
11. Now type the following into the **Cell** (next to the words **In [ ]:**)
You don't need to type in the comments!

```
import numpy as np  # numpy is a library of numerical routines
import pandas as pd   # pandas is the data handling library
dffull = pd.read_csv('file:///home/oxclo/hyg/hygdata_v3.csv')
dffull # show
```

This is creating a DataFrame. This is an object offered by the pandas library that helps deal with tabular data. It is very good at dealing with data that naturally falls into rows and columns and also that has missing elements.

12. Now click on the Run icon 

13. You should see:



14. Scroll down to the bottom of the table and you should see how many rows (stars) are in the catalogue. Note how the notebook automatically knows how to display pandas dataframes in an intelligent manner. Also note that you are not seeing all the rows or columns because there is too much data to display.

    You can see the description of the columns here:
    https://github.com/astronexus/HYG-Database/blob/master/README.md

15. Before we do any more data processing, let's configure Jupyter to do nice *tab completion*. In a new cell enter:

    ```
    %config IPCompleter.greedy=True
    ```

    Anything starting with % is a hint that this is for Jupyter not for Python. Make sure you **Run** this cell.

16. Another handy hint is to paste this into a new cell and run it:

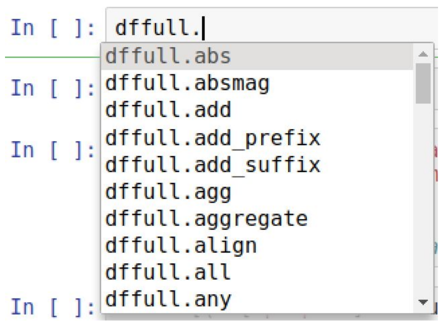
    ```
    from IPython.core.display import display, HTML
    display(HTML("<style>.container { width:100% !important; }</style>"))
    ```

    This expands the cells to take the full width available.

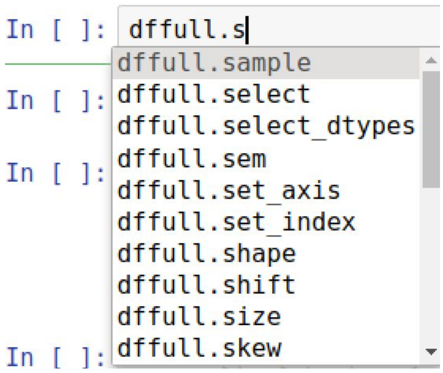
17. You can also just get that information (number of rows and columns) by using the dataframe `df.shape`. In the next cell type:

    dffull.

    Before typing anything else, hit the Tab key. You should see all possible options for syntax now appear in a little box like this:



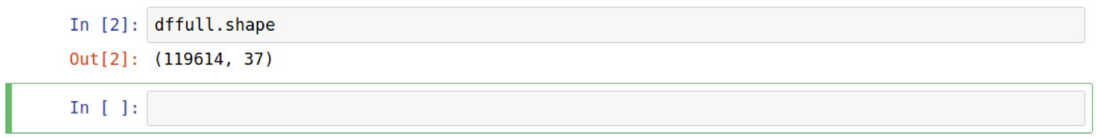
Now type ‘s’, and you should see just the operations starting with ‘s’ appear:



Now move down using the down arrow and select ‘shape’ by hitting Enter.

Now hit **Ctrl-Enter** (same as Run icon)

You should see:



Not all the columns are of interest to us. One simple approach is to create a new dataframe that only uses some of the columns from the old dataframe.

To do that, we can use the following syntax:

```
columns = ['id', 'gl', 'mag', 'absmag', 'proper', 'ra', 'dec',
   'dist','con', 'ci','lum']
df = pd.DataFrame(dffull, columns=columns)
df # show the resulting dataframe
```

18. Paste or type that into a new cell and execute it.

19. Now, let's identify the stars that have a 'proper' name.

20. There are a couple of ways we could do this. The first, simple one, is to just select that column, and then drop all NaN entries:

    Execute:

    ```
    df['proper'].dropna()
    ```

21. You should see something like:

```
In [6]:  1 df['proper'].dropna()
         2

Out[6]: 0                   Sol
        676           Alpheratz
        744                Caph
        1065            Algenib
        2076              Ankaa
                       ...
        113008        Fomalhaut
        113521           Scheat
        113603           Markab
        113687    Lacaille 9352
        118084        p Eridani
        Name: proper, Length: 146, dtype: object
```

22. Notice that this no longer looks quite the same. This is because this has created a Series object instead of a DataFrame (Each column is effectively a Series, and we've extracted one column).

23. Suppose we want the whole DataFrame (all the columns), but only those with a 'proper' name. We can use a selection function to *locate* the right rows:

    ```
    df.loc[df['proper'].notnull()]
    ```

24. You should see:

```
In [22]: df.loc[df['proper'].notnull()]
Out[22]:
```

	id	gl	mag	absmag	proper	ra	dec	dist	con	ci	lum
0	0	NaN	-26.70	4.850	Sol	0.000000	0.000000	0.0000	NaN	0.656	1.000000e+00
676	676	NaN	2.07	-0.297	Alpheratz	0.139791	29.090432	29.7442	And	-0.038	1.144986e+02
744	744	Gl 8	2.28	1.155	Caph	0.152887	59.149780	16.7842	Cas	0.380	3.006076e+01
1065	1065	NaN	2.83	-2.567	Algenib	0.220598	15.183596	120.0480	Peg	-0.190	9.264031e+02
2076	2076	NaN	2.40	0.327	Ankaa	0.438056	-42.305981	25.9740	Phe	1.083	6.444660e+01
3172	3172	NaN	2.24	-1.985	Shedir	0.675116	56.537331	69.9790	Cas	1.170	5.420009e+02
3413	3413	Gl 31	2.04	-0.312	Diphda	0.726490	-17.986605	29.5334	Cet	1.019	1.160914e+02
3759	3759	Gl 33	5.74	6.378	96 G. Psc	0.806382	5.280615	7.4549	Psc	0.890	2.447936e-01
3820	3820	Gl 35	12.37	14.222	Van Maanen's Star	0.819416	5.388610	4.2626	NaN	0.554	1.783200e-04
4417	4417	NaN	2.15	-3.981	Cih	0.945143	60.716740	168.3502	Cas	-0.046	3.407219e+03
5436	5436	Gl 53.3	2.07	-1.840	Mirach	1.162194	35.620558	60.5327	And	1.576	4.742420e+02

```
In [ ]:
```

25. This has selected every row which meets the criteria (i.e. that the column *proper* is not null).

26. You can sort the data based on a column using the following syntax, e.g. to identify the stars by distance.

```
df.sort_values('dist', ascending=False)
```

27. If you just want to see the first 10 rows of a DataFrame you can use:

```
df.head(n=10)
```

28. Use those to identify the five furthest "proper named" stars. What do you think of the data?

29. You can select on multiple criteria at once, e.g.:

```
df.loc[(df['proper'].notnull()) & (df['dist']<100000)].sort_values(
  'dist', ascending=False)
```

(extension exercise: Looking at the data, why did I choose to select data with distance < 100000 ?)

30. Identify the Gliese catalog identifier of the three least luminescent stars.

Visualisation

31. We can do some simple graphing of the data in Jupyter very easily.

    In a new cell, we can set this up with the following commands:

```
%matplotlib notebook
import matplotlib.pyplot as plt
```

32. MatPlotlib is a simple graphing package for Python. The second line imports it for use in your code. The first line tells Jupyter to automatically plot diagrams made by matplotlib.

33. Any pandas dataframe or series is automatically plottable by matplotlib (although you may not get anything useful!).

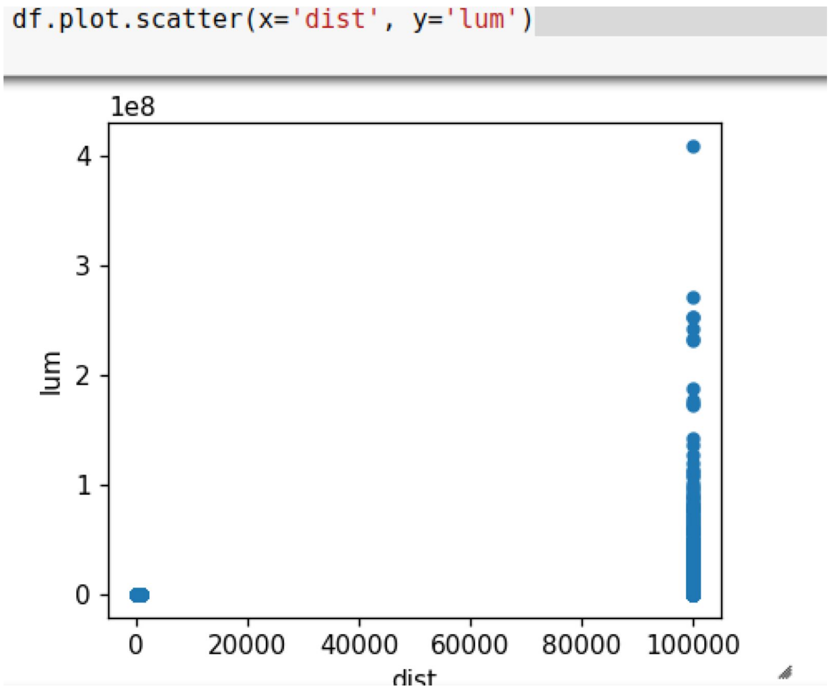
34. Try it:

```
df.plot()
```

35. For something more useful, let’s plot a scatter graph of luminosity vs distance:

```
df.plot.scatter(x='dist', y='lum')
```

36. You should see:



37. Once again, it looks like the data is incorrect and therefore not useful (see the comment in the documentation under the distance attribute).

38. Redo the graph this time filtering out any distance >= 100,000.

39. This still isn't much use. Now try making the scales logarithmic by adding the parameters `logx=True, logy=True` to the plot. Is there anything meaningful about the resulting graph?

40. **Extension:**

    Explore the data further using the matplotlib to identify any interesting correlations between the data.

41. Before finishing, close the Jupyter browser windows and then stop the Jupyter server by using Ctrl-C on the window, and then **y**.