**Exercise xx**

*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 2.7.14
* Jupyter notebooks

*Ensuring we have the pre-requisite code installed.*

1. Before we can use *pandas* we need to make sure we have it installed. From the VM’s command-line (terminal window), enter:  
     
   sudo pip install numpy  
   sudo pip install pandas  
     
   After each one, you should see software being downloaded and installed on your VM. **pip** is the Python package manager that installs Python packages from a central source of packages on the Internet. (**P**ip **I**nstalls **P**ackages)

*Downloading our sample data*

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

mkdir ~/hyg  
cd ~/hyg

1. 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

1. Either way that you downloaded it, you now need to uncompress it:   
     
   gunzip hygdata\_v3.csv.gz
2. Check it’s the right size:  
   ls -l ~/hyg  
     
   You should see:  
   -rw-rw-r-- 1 big big 33449663 Apr 21 2015 hygdata\_v3.csv
3. The VM has a “notebook” system called Jupyter configured by default. The result is that instead of starting a command line repl[[1]](#footnote-1), there is a web based editor/evaluator launched instead.
4. To start this, type (from the same command line that is in the hyg directory):  
     
   jupyter notebook
5. 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

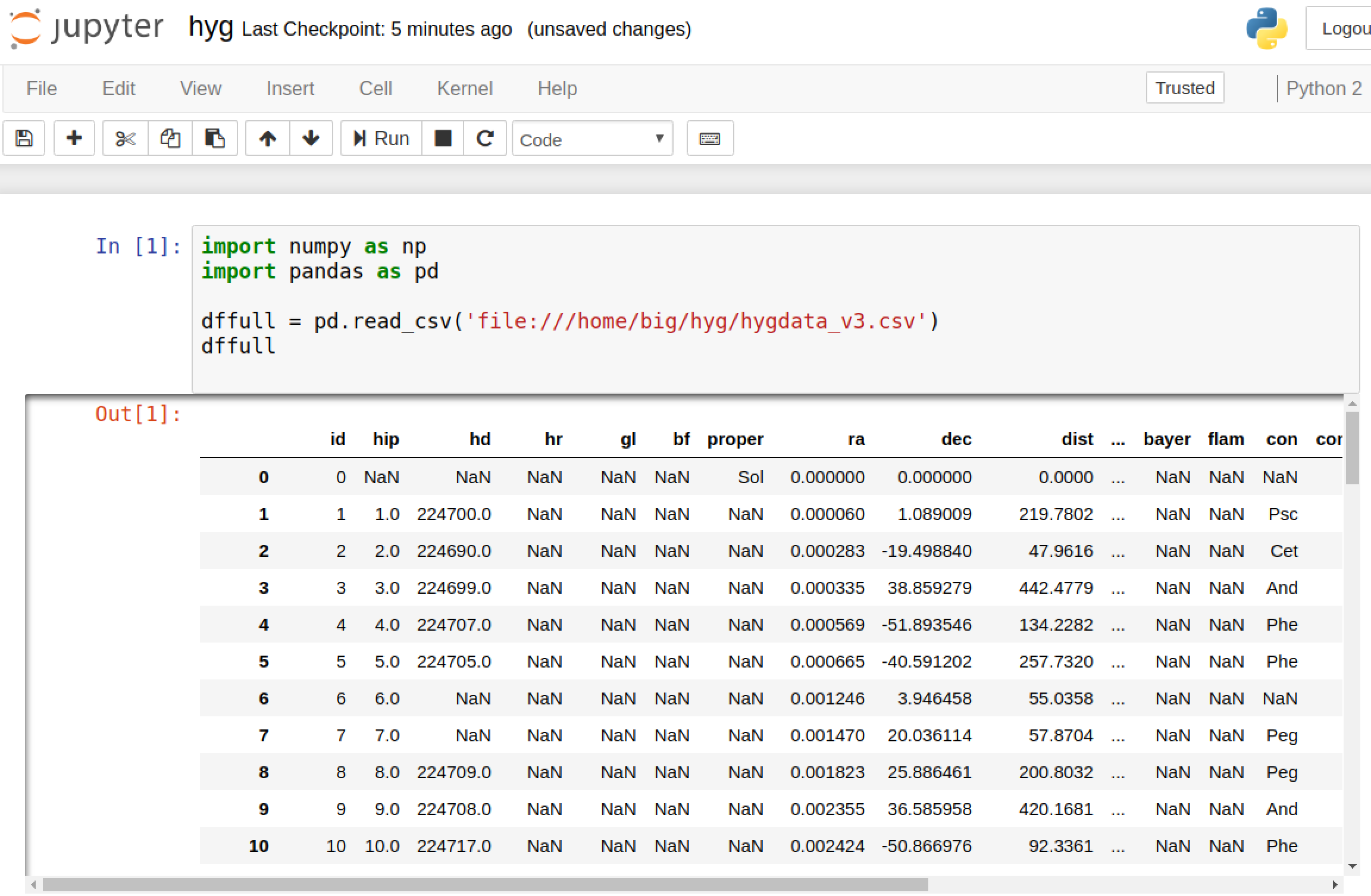
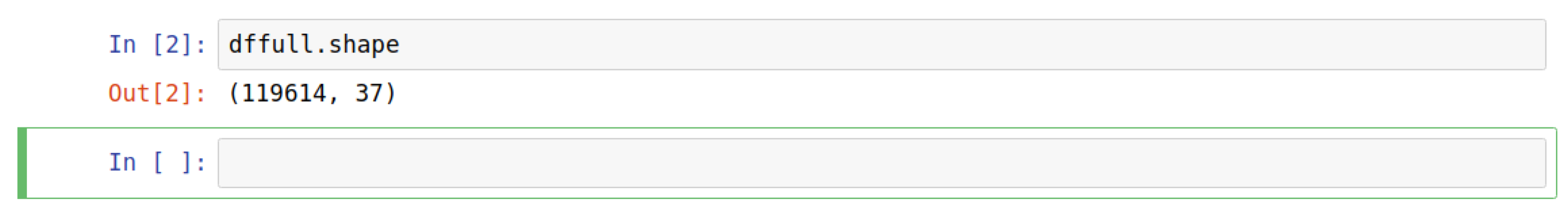
[I 13:53:23.866 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

[C 13:53:23.868 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,

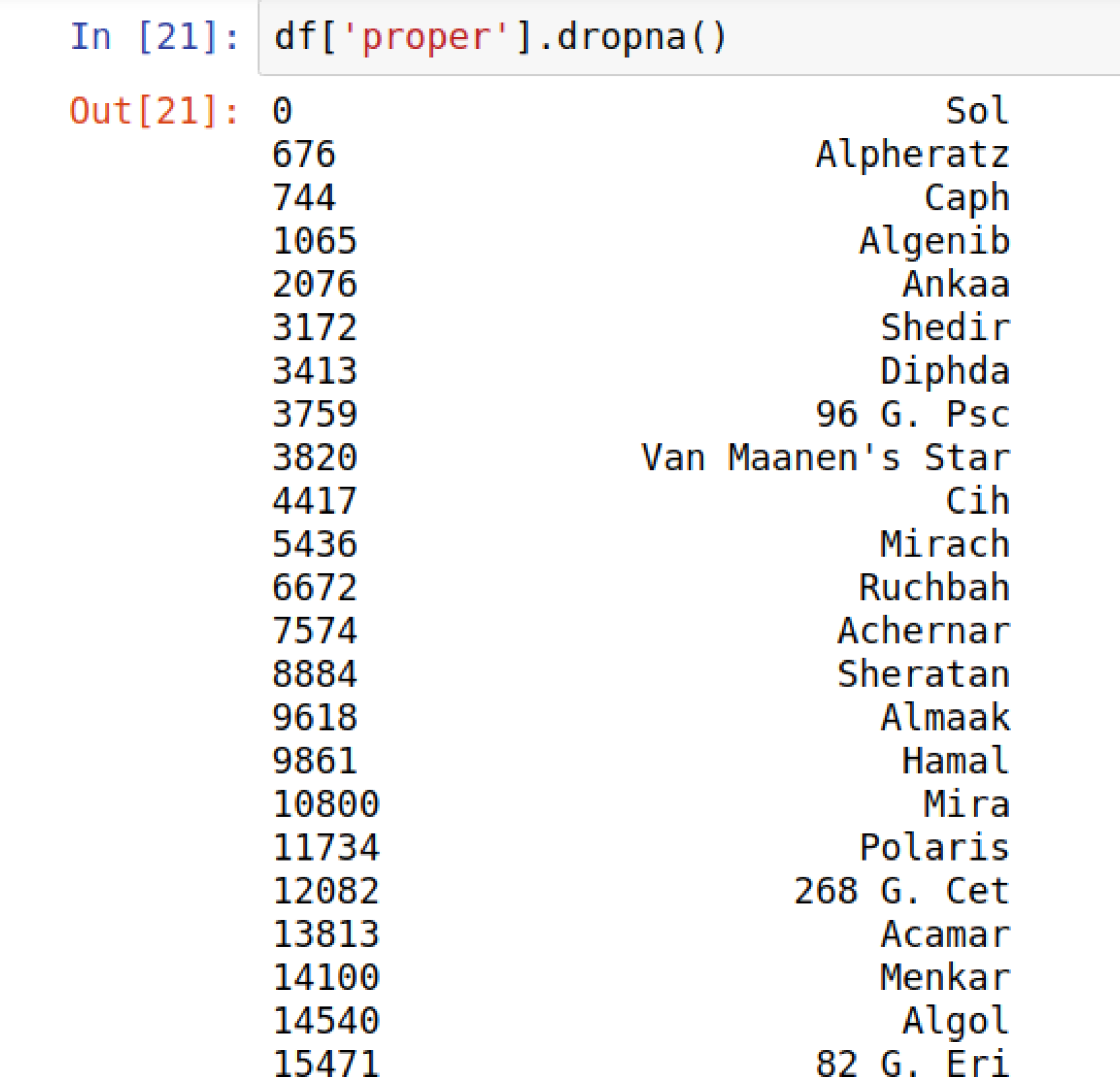
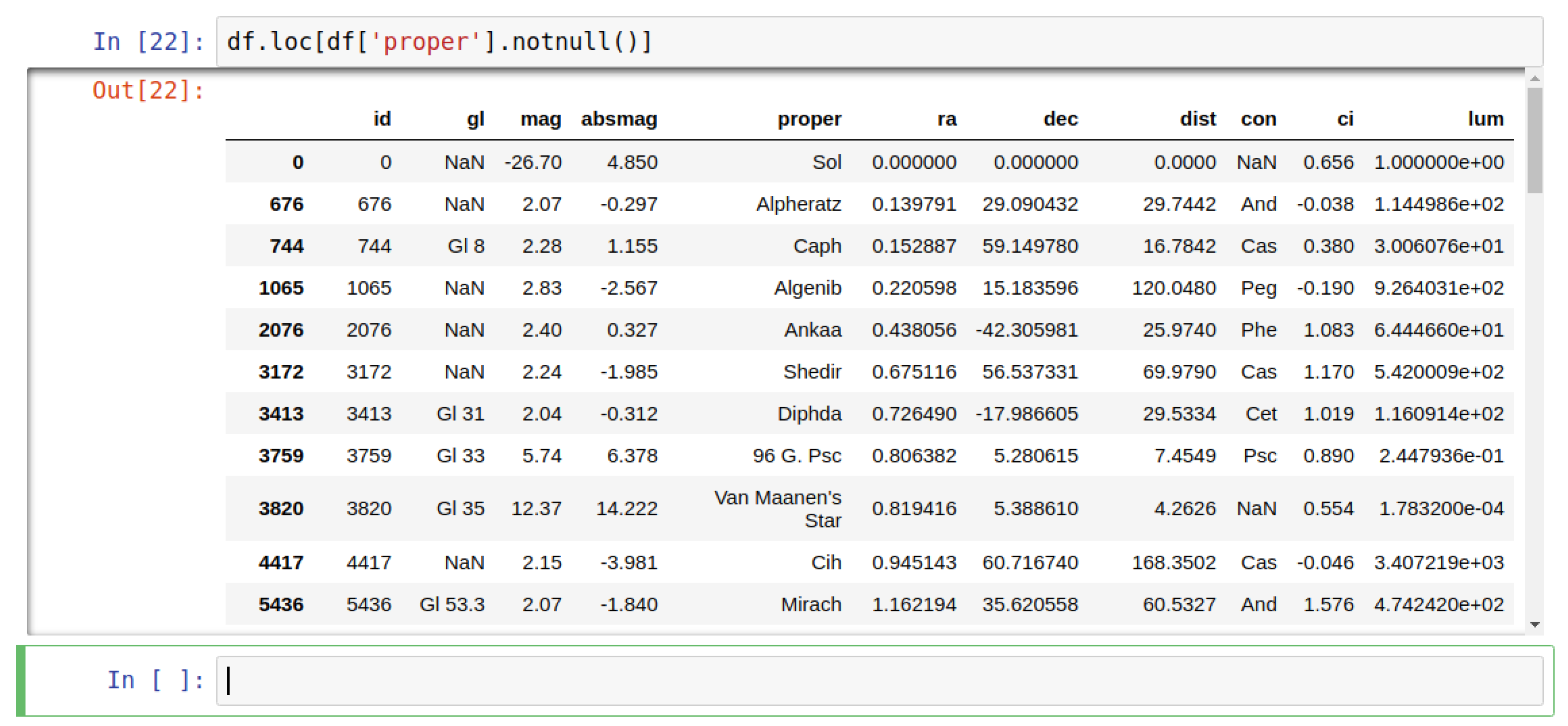
to login with a token:

http://localhost:8888/?token=fd655aab32ed4840ceb47b8b7392b1243a27f56350888a91

1. And then a browser window will pop up.  
   
2. Use the **New** button to create a new Python2 notebook:  
   
3. Click on the name of the notebook (currently “Untitled”) and rename it to **hyg**
4. 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/big/hyg/hygdata\_v3.csv')
     
   dffull # show the   
     
   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.
5. Now click on the Run icon 
6. You should see:  
   
7. 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>
8. You can also just get that information (number of rows and columns) by using the dataframe shape. In the next cell type:  
     
   dffull.shape  
     
   and 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

1. Paste or type that into a new cell and execute it.
2. Now, lets identify the stars that have a ‘proper’ name.
3. 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()
4. You should see something like:  
   
5. 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).
6. 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()]
7. You should see:  
   
8. This has selected every row which meets the criteria (i.e. that the column *proper* is not null).
9. 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)
10. If you just want to see the first 10 rows of a DataFrame you can use:  
      
    df.head(n=10)
11. Use those to identify the five furthest “proper named” stars. What do you think of the data?
12. You can select on multiple criteria at once, e.g.:  
    df.loc[(df['proper'].notnull()) & (df['dist']<100000)].sort\_values(

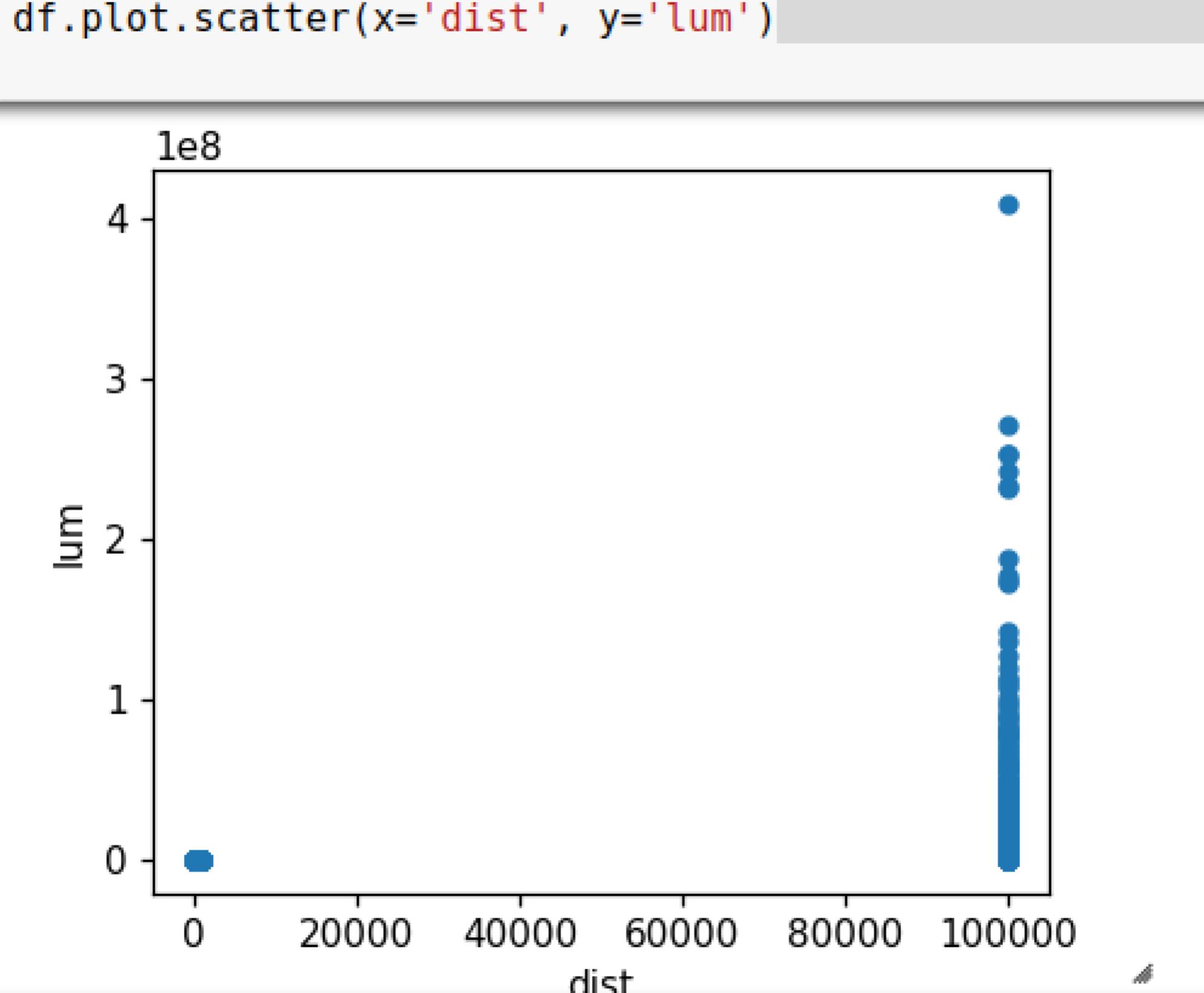
'dist', ascending=False)

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

**Visualisation**

1. 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
2. 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.
3. Any pandas dataframe or series is automatically plottable by matplotlib (although you may not get anything useful!).
4. Try it:  
   df.plot()
5. For something more useful, let’s plot a scatter graph of luminosity vs distance:

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

1. You should see:  
   
2. Once again, it looks like the data is incorrect and therefore not useful (see the comment in the documentation under the distance attribute).
3. Redo the graph this time filtering out any distance >= 100,000.
4. 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?
5. Extension:  
     
   Explore the data further using the matplotlib to identify any interesting correlations between the data.
6. Before finishing, close the Jupyter browser windows and then stop the Jupyter server by using Ctrl-C on the window, and then **y**.

1. Read Eval Print Loop [↑](#footnote-ref-1)