SXPS288-19J Proposed Python Integration

Python Integration into SXPS288-19J – start to end of Astronomy Topic.

# Version and Notes

Version: 1.0

Date: 25/06/2019 10:07

Notes:

* First suggested version.
* Includes suggested text for insertion/substitution in existing text.

# Introductory Sections

OU NOTE: Possibly move the ‘Python – Everything you wanted to know’ link to the ‘Before the module begins: Preparing for SXPS288’ section as we might hope this is mostly revision.

OU NOTE: We should provide a ‘Python’ section in the Resources tab for notebooks etc.

# Week 1

OU NOTE: We should provide a simple notebook in the ‘Python’ Resources section (see above) for the students to download and run in section 3.6 of the ‘Python – getting started’ section.

ADD the following activity/text at the end of section 3.6:

“Activity 0.1 Running a notebook: You will find a simple notebook entitled ‘Blah Blah Blah’ in the Resources section’ Download this, save it somewhere where you can find it and make sure you can run it.”

ADD a new section somewhere in Week 1

“Python – Revision and New Concepts”

ADD the following text:

“

Being able to program in Python is an essential skill for any scientist involved with collecting, analysing or modelling data. You will already have picked up some of these skills but may be a little rusty. In this section you will look at and interact with number of Python Jupyter notebooks (available under the Python Resources tab) which will consolidate some of the coding knowledge needed and introduce some new concepts and ways of working. You will find that many of the notebooks contain exercises for you to try out. Don’t skip these as they are designed to help ensure you will be able to do the necessary Python activities that you will encounter later in this module. You will be given simple solutions, but it is advisable to try the exercise before you look at them. You will also find that later on you will be able to use some of this code in completing the data processing activities so time spent now will pay dividends later. The ‘Python – Everything you wanted to know’ section should help you here.

You can do these notebooks in any order but the following is a suggested sequence.

1. FileIO: This introduces you to getting data into and out of a Python program using basic Python commands. In the next notebook you will be introduced to a Python package called ‘Pandas’ which, in many ways, simplifies this process. However, there will be times when plain old Python will be the only way.
2. Pandas: Pandas is a package that extends, or simplifies, Python’s data processing abilities. You can easily read in data formed in a spreadsheet like manner and then process it much like you might try to do in a spreadsheet package. It is however, more powerful than a spreadsheet and can handle much bigger data sets.
3. Matplotlib: This revises the matplotlib plotting package and introduces some new concepts.
4. Astropy: This is a Python package written and used pretty much exclusively by astronomers. It provides a number of useful functions that allow you to easily switch between different astronomical coordinate systems.
5. Regex: This covers the idea of ‘Regular Expressions’ which allow you to search for specific strings within data. Initially, this subject can seem pretty daunting, but once mastered provides a very powerful tool. This notebook is optional as you can do most of what is required later on without it. However, it will simplify many of the tasks you do come across.
6. Bokeh: This package provides display functions in a similar way to Matplotlib which you already know. Where it differs from Matplotlib is in its extensive ‘interactive’ abilities. As with Regex, this is optional but will make your life easier later on.

“

# PIRATE

T.B.D.

# ARROW

OU NOTE: I’ve suggested some rearrangement of the order of the presentation and activities within ARROW. This is to allow the Python Activities to look like they are more than just Excel substitutes. It also avoids a lot of output/re-input of results – either to file or via pen/paper. All of which should help convince the students that Python is just a waste of effort. I don’t think this should involve much extra effort – just text rearrangement.

## WEEK 1

OU Note: I suggest we provide full answers to week 1 activities by providing the notebook called “Early ARROW Python Exercises”.

Section 1.3 21cm radio emissions

Turn ‘Reveal Answer’ question about wavelength to frequency below Figure 1.1 into a Python Activity. Use the following text:

“

1. Write a short Python program to ask the user to input a wavelength in cm and output the frequency. Define any physical constants as variables.
2. BONUS - Display the output using scientific notation to 4 decimal places. (See section 3.4 of Python - Everything
3. BONUS - Extend this so that it allows the user to repeat the conversions until no longer required.

NOTE: Answers to these are provided in the Notebook “Early ARROW Python exercises.”

“

### Section 1.4 The Doppler effect.

Add an Activity to this with the following text:

“

1. Write a function that takes a frequency shift (or numPy Array or Pandas Series of frequencies) and a rest frequency and returns these converted to radial velocity, using the Doppler effect.
2. Optionally, using information from the 'astropy' notebook, use constant values defined within the astropy.constants module for the speed of light. Also, try and use the astropy.units to return the result in km/s
3. Write a short program to test this.

HINT: You’ll need to consult the ‘Astropy’ notebook here and make use of numPy arrays (Python – Everything you wanted to know, section 5.3).

NOTE: Answers to this is provided in the Notebook “Early ARROW Python exercises”

“

## WEEK 2

### Section 2.2 Beam width and resolution.

Add an activity to this with the following text:

“

Write a short Python program to ask for the diameter of a radio telescope dish in metres and the observed wavelength in cm. Print out the beam width in radians and also in degrees.

NOTE: Answers to these are provided in the Notebook “Early ARROW Python exercises”

“

### Section 2.7 , Activity 2.4 - Target Selection

Provide an alternative, Python solution for Activity 2.4. Add the following:

“

ALTERNATIVELY, this activity can be completed as a Python program. This will allow you to further develop your coding skills and, once complete, will allow you to very quickly re-calculate an observing plan should you need to change your observing time. This programming activity requires you to:

Write a program to take an observing time, an observing location, and a target (In this case, the location is ARROW’s, the time should be entered by the user and the target should be a Galactic longitude located on the Galactic equator) and work out if this target is visible from the location at the observing time.

HINT: Pretty much all of the code hints you need can be found in the Astropy notebook.

HINT: The target will be visible if it is within ARROW’s Altitude and Azimuth limits that can be found in the Stellarium activity above.

HINT: As this is the first major, astronomy related bit of code you’ve attempted, here is a suggested way to do this:

1. Import appropriate packages/modules (you only need stuff from Astropy)
2. Input date and time of the proposed observation (ideally prompt for them in suitable format - 'yyyy-mm-dd' and 'hh:mm:ss')
3. Create an ARROW EarthLocation (the 'Location' example in 'Astropy' notebook shows you how to do this.)
4. Create an 'observation' object (the last exercise in the 'Astropy' notebook will help here.)
5. Create a 'SkyCoord' for your observation target in the 'galactic' frame ('l' from above, 'b' will be 0. Once again, 'Astropy' notebook will help.)
6. Now convert from a 'galactic' frame to and 'altaz' frame. ('Astropy' notebook - Frames section).
7. Get the Altitude and Azimuth (in degrees is best) and check if the valuesnmean that the target is visible.

OPTIONAL: You could read, or calculate, a list of Galactic longitudes and use these in a loop to repeat steps 5-7 and print out a set of targets indicating if they are visible for your observation time or not.

“

## WEEK 4:

OU NOTE: It makes sense to do the background subtraction in the same program as the radial velocity conversion and combine these to a single Activity. In this way you don't have to write out then re-read a redundant data file.

Therefore I suggest a modified Activity 4.1 moves into section 4.1 and making it a 'Display and Explore your data' exercise.

The rest of 4.1 should then be merged into 4.2 to make a first stage data reduction Activity with the optional baseline removal integrated into this single Activity.

### 4.1 Data analysis

ADD the following after the current two paragraphs in section "4.1 Data analysis" (perhaps change name of section to “Data inspection”?).

"

Often, the first task undertaken with collected scientific observations is to visualise or display the raw data. This allows the scientist to ensure that the data 'looks right' - is in a general form that might be expected and essentially free from gross distortions. It also allows a quick opportunity to make some notes on any key features that might stand out (maximum and minimum values, rough positions of any standout feature, the 'noisiness' of the data etc.) and this might well prove be useful in planning the future processing of the data.

"

And then add the Activity:

"

Activity 4.1 Displaying your own data

Allow approximately 1 hour

Relevant learning outcomes: %%^^%^

In this activity you will write a Python program to display a raw ARROW spectrum. When you've done this, display each of your ARROW spectra and make rough notes covering the key features you observe.

In overview this will involve the following:

1. Input the name of a file to display
2. Read in the data.
3. Display the spectra in a clear format

HINT: The best display package to use is 'Bokeh'. Review the Bokeh notebook you met in week 1.

HINT: The best package to read in the data is 'Pandas'. Review the Pandas notebook you met in week1. The spectrum file contains 12 header lines. For this activity these are not important (they will be for later activities).

HINT: There is a notebook in the Python Resources called "ARROW\_4.1\_Hints" that contains hints, tips and code snippets that you might find useful.

OPTIONAL: You might like to read in several of your files at once and display them in a 'grid' of sub-plots. Later on this will be very useful in producing figures for inclusion in your TMA report.

"

### 4.2 data processing

Remove the “Step 2 Plot the spectrum” step from the current Activity 4.1 and combine the rest of it into the Activity 4.2.

“

*Activity 4.2 - Processing the data in your observed spectra.*

*Allow approximately 4 hours*

*Relevant learning outcomes: %%^^%^*

*Write a Python program that will read in a spectrum file, optionally remove the baseline distortion, convert the frequency values to radial velocity values and write a new comma delimited file (CSV file) containing this converted data.*

*In overview you will be writing a Python program to:*

1. *Read in a raw spectrum*
2. *Optionally, read in, average and subtract baseline spectra*
3. *Convert the measured frequencies, using the Doppler formula, to radial velocities*
4. *Optionally display the converted spectrum - use either 'Matplotlib' or, preferably, 'Bokeh'.*
5. *Write the spectrum back to another file.*

*HINT: You'll need to read the spectrum header lines (there's 12 of them - all starting with '#') in using ordinary Python File IO. Later you'll use ordinary FileIO to write these to a new file then APPEND the modified pandas Dataframe to this as comma delimited data. (use pandas .to\_csv() with "mode='a'). You'll want this information preserved for future use in the Topic.*

*HINT: Read in, process and write out the main spectral data using Pandas.*

*HINT: You can display the spectrum using either Matplotlib or Bokeh. Matplotlib is superficially easier but Bokeh will be more useful later on.*

*HINT: Review the 'Pandas' and 'Bokeh' notebooks which you first used in Week 1*

*HINT: There is a notebook in the Python Resources called "ARROW\_4.2\_Hints" that contains hints, tips and code snippets that you might find useful.*

**Step 1a: Read in the header data**.

The first few lines of the data file contain information about the observation.  Here is an example of the top few lines of a data file taken toward Galactic longitude 80o.

#instrument=ARROW  
#ra=20.6067  
#dec=40.5706  
#alt=28.6425  
#az=59.6833  
#tracking=true  
#startTime=1477310580  
#startTimeUTC=2016-10-24T12:03:00.566Z  
#user=is4555  
#fullScan=true  
#completeTime=1477310680  
#completeTimeUTC=2016-10-24T12:04:40.937Z  
frequency intensity  
-800000 2.49  
-795000 2.463  
-790000 2.468  
-785000 2.48

The first twelve lines (starting with #) show the details of the observation including the date and time that the observation was made, and the telescope right ascension and declination.

These 12 lines should be read using standard Python File IO and stored for later. You’ll need to add them back when you write the modified file back to disk.

**Step1b. Read the main spectrum data**

The remaining lines contain information from each of the 401 frequency channels of the radio spectrometer, starting with a heading row, labelled ‘frequency’ and ‘intensity’. For each row, you can see:

* the frequency shift Δf  from the frequency of neutral hydrogen, in Hz (e.g. ‘-78500’ in the last row shown above. The negative value represents motion away from us)
* the intensity measured by the telescope (in arbitrary units) in each frequency channel (e.g. ‘2.48’).

Use Pandas to read this data into a DataFrame with columns indexed/named with the column header ‘frequency’ and ‘intensity’

**Optional Step 2a. Process it – remove the base line**

This optional part of this activity uses the spectra you might have collected 'off-source'.

INCLUDE TEXT/FIGURE FROM Activity 4.2, INCLUDING Figure 4.2 AND FOLLOWING PARAGRAPH – UP TO BUT NOT INCLUDING “Step 1 Read in …”

You'll need to read these in, find and average of the 'intensity' column (as the frequency column will be the same as the main spectra you only need to concern yourself with the 'intensity' column)

The steps will be something like this:

1. Read in the separate background files using Pandas. This will be as above, but you can ignore the header lines completely.
2. Average the 'intensity' columns. Take advantage of the fact that, as with a numPy 1D Array, you can sum a number of Pandas Series (or DataFrame columns) by just using '+'. And you can divide a whole column by a number by just using '/'. See Section 5.3 'NumPy Arrays, in the "Python Everything You Wanted To Know" resource.
3. Subtract this from the main spectrum 'intensity' column.

**Step 2. Process it – Convert from frequency to radial velocity.**

The calculation is done using the Doppler shift equation (Eq 1 in Section 1.4) rearranged to make v (the speed of motion of the source of radiation along the line of sight, i.e. the radial velocity) the subject:

where  is Δf is the frequency shift in the data file, c is the speed of light and  f= Δf+f0. In this expression, f0 , the rest frequency of hydrogen is 1420.4 MHz. The radial velocity is greater than zero if the object is moving away from the observer, or less than zero if it is approaching the observer.

Take care with units and express the radial velocity in the spreadsheet in .

The first channel of the spectrometer is listed as a frequency shift of -800000Hz . By the analysis outlined above, you should have found that this corresponds with a speed of about 169 kms-1 . Repeat the calculation, but this time instead of using f= Δf+f0  in the denominator, use f . Does this make a significant difference to your result ?

[Reveal answer](https://learn2.open.ac.uk/mod/oucontent/view.php?id=1478970&section=4.2)

The difference in frequency shift between adjacent channels of the spectrometer is about 5000 Hz. What difference in radial velocity does this correspond to at a frequency of 1420.4 MHz?

[Reveal answer](https://learn2.open.ac.uk/mod/oucontent/view.php?id=1478970&section=4.2)

Look in the ‘Hints’ notebook for help here but remember you can operate on the whole column at once because it is a Pandas object. You’ll also only have to add a new column - ‘velocity.

**Step 3. Display it**

Use Bokeh, or Matplotlib to display it.

**Step 4. Write it out to disk.**

Don’t forget you’ll need to write the header information you read in and save earlier before you write out the new data.

Use the pandas “to\_csv()” function to write out he main spectrum data easily. You might like to include the “index=False’ parameter.

Don’t forget to choose a suitable file name.

“

### 4.4 Measurements from your data.

Modify Activity 4.3:

Change the first line from:

“Inspect each spectrum, and measure the speed corresponding to the centre of each peak in each spectrum.”

To

“Using the program you wrote for activity 4.1 (slightly modified – you need to display ‘velocity’ now), inspect each spectrum you saved from Activity 4.2 and measure the speed corresponding to the centre of each peak in each spectrum.”

## WEEK 5

OU NOTE: Activity 5.1 – we could ask the students to do this with a Python program (I’ve written a function which could be provided), but this probably isn’t the best use of their time. In the future, this kind of function would be best applied at the point after changing frequency to velocity (Activity 4.2) as part of the data reduction, producing LSR corrected spectra which could then be used for the peak measurement directly. However, this is probably too much of a re-write of the material. Perhaps next session?

### Activity 5.2 Plotting and measuring speeds from archive spectra

"

***Activity 5.2 Plotting and measuring speeds from archive spectra***

Allow approximately 1 hour

Relevant learning outcomes: %%^^%^

In this activity you will write a Python program to read in an array of archived, high resolution ARROW spectra. When you've done this, display each of these ARROW spectra and *determine the maximum positive speed at which hydrogen emission is detected. Estimate an appropriate uncertainty for each measurement you make. Note that these archive spectra have already been corrected to the Local Standard of Rest, so no further adjustment to the observed speeds is necessary. Record all your measurements in your log book.*

In overview this will involve Reading in the archived data and then looping through, and displaying, these spectra in order to measure Vobs\_max.

In a little more detail:

1. Estimate Vobs\_max with an estimated uncertainty and record this in your lab notebook..

HINT: This is basically similar to Activity 4.3

HINT: The best display package to use is 'Bokeh'. Review the Bokeh notebook you met in week 1. This notebook contains a simple, but useful example of looping through columns of data (Section 5 brings this together).

HINT: The best package to read in the data is 'Pandas'. Review the Pandas notebook you met in week1. The spectrum file contains 12 header lines. For this activity these are not important and can be discarded.

**Step 1. Get the name of the file containing the archived spectra.**

**Step 2. Read the data into a pandas DataFrame**.

**Step 3. Loop through the spectra.**

In turn, loop through the separate data columns, display them using Bokeh and measure Vobs\_max (with uncertainty) for each longitude. NOTE. The Bokeh notebook contains useful examples of how to do this.

“

### Activity 5.3 Plotting the rotation curve

“

First, you need to produce a comma delimited file that contains the information about V\_obs\_max you’ve just measured in Activity 5.2. This file should have the following format:

A first line that reads longitude,v\_obs\_max,v\_error with subsequent lines that contain the appropriate three numerical values separated by a ‘,’. So, a typical file might look like:

longitude,v\_obs\_max,v\_error  
0,61.2,5  
10,197,5  
20,149.7,3  
30,137.2,10  
40,112.8,7  
50,103.2,6  
60,72.8,7  
70,52.3,5  
80,47,5  
90,39.1,7

OU NOTE: I guess we should alter these values to nonsense to avoid someone cutting and pasting it and thus avoiding Activity 5.2. We could make a suitable file available for students unable to complete 5.2 for legitimate reasons.

NOTE: You could use a spreadsheet to do this or a good text editor. If you use a text editor, don’t use something like Word or WordPad that might place ‘invisible’ formatting codes into the file. On Windows, Notepad++ is a suitable, and free, text/code editor that can, incidentally, also be very useful for writing Python code.

Save this file with an appropriate name such as ‘vobs\_max.csv’.

Now produce a Python program to read in this data, transform this to R (radius from centre) and V (rotation speed) values and plot a rotation curve.

In more detail:

1. Set up any package/module imports, functions you might want to define and any standard variable values you might need. You will, at least, use the adopted values for the distance of the Sun from the Galactic centre (R0 = 8.5 kpc)  and the rotational speed of the Sun around the Galactic centre (V0 = 220 kms-1) in the calculations.
2. Read the data from the file you’ve just created into a Pandas DataFrame (or a numPy array – we’ll use a DataFrame from here on as an example).
3. Add a new column (indexed as ‘R’) to the DataFrame containing the calculated values using Equations 5.5 from earlier. (R=R0 sin(γ)).
4. Add a new column (indexed as ‘V’) to the DataFrame containing the calculated values using Equations 5.8 from earlier. (V = Vobs,max + V0 sin(γ)).
5. Add a new column (indexed as ‘V\_err’) to the DataFrame containing the estimated error values for V. HINT these will be the same (in this case) as v\_error.
6. Plot out V vs R to get the Galactic Rotation Curve. Don’t ‘join the dots’ with a line and do try and include error bars.

HINT: because you’re using a DataFrame column/Series or a numPy array, you can operate on the whole object at once – no need to iterate through all the values.

HINT: Use numPy sin() and be careful to convert degrees to radians if appropriate.

HINT: There is a notebook in the Python Resources called "ARROW\_5.3\_Hints" that contains hints, tips and code snippets that you might find useful.

OPTIONAL (Very): In the next Topic we’ll be introducing you to a new Python skill – fitting optimised curve equations to data. If you’re feeling very Python confident, look at a new package called ‘SciPy’ and one of its functions scipy.optimise.curve\_fit. See if you can work out of to fit a curve of the form V = C0 – C1e-c2R to your data. This has no real Physics basis but will produce a curve of the right form.

The rotation curve plot should show very little variation in speed as a function of distance from the centre of the Galaxy, at least beyond about 4 kpc from the Galactic centre. Remember that the uncertainties you recorded in each value of  will give an associated uncertainty in the plotted values of  on your rotation curve.

“

### Activity 5.8 Mapping out the spiral arms of the Galaxy

“

This is very similar to Activity 5.3. First, you need to produce a comma delimited file that contains the information about the cloud speeds you measured in activity 4.3 and corrected to the LSR frame of reference in Activity 5.1. *Now add in a selection of data from other observers, as posted to the project forum, corresponding to other Galactic longitudes (i.e. other lines of sight). Feel free to add as many or as few extra data as you wish.* This file should have the following format:

A first line that reads longitude,vobs with subsequent lines that contain the appropriate two numerical values separated by a ‘,’. So, a typical file might look like:

longitude,vobs  
30,123  
30,-1  
40,68  
40,-5  
40,-23

Save this file with an appropriate name such as ‘clouds\_v.csv’.

OU NOTE: I guess we should alter these values to nonsense to avoid someone cutting and pasting it and thus avoiding Activity 5.2. We could make a suitable file available for students unable to complete 5.2 for legitimate reasons.

NOTE: You could use a spreadsheet to do this or a good text editor. If you use a text editor, don’t use something like Word or WordPad that might place ‘invisible’ formatting codes into the file. On Windows, Notepad++ is a suitable, and free, text/code editor that can, incidentally, also be very useful for writing Python code.

Save this file with an appropriate name such as ‘clouds.csv’.

Now produce a Python program to read in this data, transform this to R (radius from centre) and V (rotation speed) values and plot a rotation curve.

In more detail:

1. Set up any package/module imports, functions you might want to define and any standard variable values you might need. You will, at least, use the adopted values for the distance of the Sun from the Galactic centre (R0 = 8.5 kpc)  and the rotational speed of the Sun around the Galactic centre (V0 = 220 kms-1) in the calculations. You’ll also need a value for the constant Galactic rotation speed, V, you found in Activity 5.3 which will be of the same order as V0.
2. Read the data from the file you’ve just created into a Pandas DataFrame (or a numPy array – we’ll use a DataFrame from here on as an example).
3. Add a new column (indexed as ‘R’) to the DataFrame containing the calculated values using a rearranged Equations 5.12 from earlier. (R=VR0/[V0+(Vobs/sin(l)]).
4. Print out the longitude/R pairs if you want to plot the arms by hand or…
5. Optionally, use Matplotlib to plot out longitude vs R. (use a ‘polar’ projection

HINT: Use numPy sin() and be careful to convert degrees to radians if appropriate.

HINT: There is a notebook in the Python Resources called "ARROW\_5.8\_Hints" that contains hints, tips and code snippets that you might find useful. “