Geol 497D/GeoSci697D - Spring 2020
Programming and Data Analysis

William P. Clement and Scott Marshall

Lab 8: More File I/O

Chapters in Learning Scientific Programming with Python (LSPP):
Chapter 6.1.5, especially page 196.
Statistical methods; Chapter 6.3

In this laboratory exercise, you will analyze data sets with a more complex layout. Think of this laboratory exercise as a nice way to use all of the coding skills you have learned thus far to solve an interesting geologic problem of local (Part I) and global (Part II) interest. You may use any combination of vectorized code and loops that you find necessary to solve the tasks at hand. Your final code will be handed in via a single zip file following the instructions at the end of the lab. Make sure to comment your code and include the proper python header information in your script.

Part I: Processing Automated Well Data from MacLeish Field Station

The department of geology has set up an automated continuous monitoring station for MacLeish Field Station in Whately, MA. The monitoring station consists of a well with a temperature-pressure transducer. The pressure transducer can be used to calculate the depth of the water table in the well. The downside of this automated data is that the device writes a new file every month, so there are a lot of different data files. Your task is to write a python script that will process and visualize all of the stream data. Specific instructions are below.

1. Make a new script called `processWaterLevel.py`
2. Go to the course website and download and extract the zip file (MFS1.tar.zip) for this lab. It contains several .csv files. Your script should process all of the .csv files in the current directory. This means that as new files are added in, your script should automatically detect all of the files when it is executed. Therefore, you cannot hard code in a list of the filenames. You will have to use 'ls' as a function to generate a list of the currently present files that end in .csv and loop through the file list. You will likely use wild card characters to make the list. For example, MF*.csv will list all files that start with MF and end with .csv. Make sure that you test for file opening errors. You do not need to test for file closing errors. As each file is opened, your script should print a message to the screen. For example:

   Processing 13081400.csv

3. As your script reads in each file it should store all of the data in appropriate variables. Your code should vertically concatenate (`np.vstack`, LSPP, p. 196) and write all of the data into a single file called `allData.csv`. This will probably require a separate loop from your file processing loop in question 2 above. You should use `np.savetxt` to write the file. Be careful to make sure that your code doesn't keep adding to `allData.csv` each time you run your script. The safest was to do this is to start your script off opening `allData.csv` for writing.

4. As always, your script should visualize the entire data set (from all files). Make two plots in the same figure. The first plot should have time in days on the x-axis and the water level in meters on the y-axis. Plot this with a solid blue curve (no symbols). Recall that what the device actually measures is water level above the transducer in meters. Also, because the pressure transducer sits 24 cm below the water level, you will have to add 24 cm to all of the depth measurements to get the correct stream stage. Also note that there is no time column for time in decimal days. You can create this data set based on the length of the total concatenated dataset and the fact that the dataset has 10 minute sampling with no gaps. HINT: use `np.linspace` or `np.arange`.

Part II: Processing a Paleoclimate Record Based on δ^{18}O Values in Foraminifera

Climate change is a hot topic right now (pardon the pun), and much climate work involves processing large quantitative data sets. For this task, you will process a data file full of oxygen isotope ratios derived from microfossils called foraminifera (commonly called "forams"). δ^{18}O is the ratio of ^{18}O :^{16} O isotopes. When ocean water evaporates, ^{16}O evaporates preferentially. This means that when the overall climate is colder, the ^{16}O evaporate goes into vapor/clouds and then precipitates out as snow/rain out on land, which partially gets locked up in ice sheets. Therefore, during ice ages, the ratio of ^{18}O to ^{16}O in the oceans goes up because more ^{16}O is removed from the oceans and is instead locked up in ice on land. If all change in δ^{18} is due to temperature (a big and sometimes incorrect simplification), the general rule is that a change of -0.2‰= 1 °C change. This at least gives you some idea of what kind of changes are significant.

Your next task is to write a python script that will process and visualize an oxygen isotope data set extracted from marine forams identified in the Integrated Ocean Drilling Program (IODP) core ODP-677. Forams can be benthic (bottom-dwelling) or planktonic (living in the shallow depth zones of the oceans), and as such, they have slightly different sensitivities to climate. For that reason, the data set provides δ^{18} estimates from both types of forams. This is a highly cited data set and comes from Shackleton et al. (1990). Specific instructions are below:

1. Make a new script called `process018.py`

2. Your script should read in two files, `odp677_benthic.dat` and `odp677_planktonic.dat`, provided on the course website. To read in the file you should use the pandas command `pd.read_fwf`. You need to read the header in the files to determine the format of the data.

   The source data file has data in columns, which are labeled in the file. You must write your code to read in the file as it is presented. As always, your script should check for errors when opening the file. Once the data has been read in, your script should print a message stating how many data points were read (i.e. the total number of rows in the file). For example:

Found X Total Benthic Isotope Values Found X Total Planktonic Isotope Values

3. The years before present data should be converted to thousands of years before present. Once this is done, your script should print out a message to the screen saying how many total values were kept for each data set. For example:

   Found X Benthic Oxygen Isotope Values
   Found X Planktonic Oxygen Isotope Values

4. You should make two plots on the same figure. Both plots have the x-axis as "Time Before Present (kyr)", and the top graph, plotted with a dashed red curve, will have the y-axis as "Planktonic δ^{18}O (permille)" and the bottom plot, drawn with a solid blue curve will have the y-axis labeled "Benthic δ^{18}O (permille)". Please use `plt.tight_layout()`, so your plot fits tightly around your data.

5. Based on this data, paleoclimate researchers have shown that the periodicity of climate fluctuations (i.e. the wavelength of your plots), made a significant change and that these changes can be related to physics of Earth's orbit. You can read about the details about how Earth's orbital parameters can fluctuate, and affect climate here: `http://earthobservatory.nasa.gov/Features/Milankovitch/milankovitch_2.php`.

   The short version is that paleoclimate researchers use the data you just plotted to show that the wavelength of climate fluctuations was approximately 100,000 years in recent times, suggesting that Earth's orbital eccentricity was controlling more recent climate changes. Then at some point in the past, the wavelength becomes much shorter, say around 40,000 yrs. This would be consistent with climate changes being driven dominantly by changes in Earth's axial obliquity (see NASA website for details).

   Your final task is to visually look at your plots, decide where you think this wavelength shift happened in the past, and mark it with a black vertical line that spans the entire vertical range of your plots. On your plot, use `plt.vlines` to plot a line at your choice for the wavelength shift. Also, use `plt.text` to plot the value on the figure. Use the string value 'Frequency Change' as the text. You should know how to figure out the min/max of a dataset, so this will give the y-values for your vertical line. The x-values, you will visually choose, so the line is right at the time wavelength shift. Your script should then print out a message to the screen stating when the wavelength shift occurred in thousands of years ago (ka).

   Wavelength Shift: x ka

   I will be very flexible on this, but give it your best shot.

Shackleton, N.J., A. Berger, and W.R. Peltier, 1990. An alternative astronomical calibration of the lower Pleistocene timescale based on ODP Site 677. Transactions of the Royal Society of Edinburgh: Earth Sciences, 81, 251-261, 1990.