CSC108H Lab 6: Files

As in previous weeks, decide which of you will be the first driver and which will be the first navigator. Switch roles when directed in the lab handout.

At the end of the lab, show your TA your work to get credit for the lab. At the same time, please return this handout to your TA. We will post the handout on the course website at the end of the week.

Objectives

• Learn different ways to read from and write to text files. • Practice processing input from files.

Text Files: the Basics

In this section, you will practice opening, reading, and writing text files. Use dir and help to get information on file methods provided by Python. Before you begin, download the text files data1.txt and data2.txt from the Labs page on the course web site.

1) Do the following in the Python shell:

(a) Open data1.txt for reading. Associate the open file with the name input_file.

(b) Read all of the lines from the file using the readline method repeatedly. How many times did you have to execute the method? What happened when the file was completely read? Close the file.

(c) Open the file again but use a for loop to read all of the lines from the file. In the body of the loop, print each line. *Switch roles*.

2) Open a new python file in the editor. Write a program that opens the file data2.txt. Do not use choose_file; instead, just provide the name of the file to open as a str. (As long as you save your new python file in the same folder as data2.txt, Python will be able to find data2.txt.)

Add a loop to your program. It is similar to the one you wrote in the shell, but it should print only lines containing the string "lol" in any mixture of upper and lower case. (Hint: Use str's find method to locate "lol", and consider converting text from the file to lowercase first.) Make sure that your output is not double spaced. *Switch roles*.

3) Modify your program to open a new text file named output.txt for writing. Instead of printing the lines with some form of "lol", write them to the file.

File Processing

This section of the lab involves processing temperature data stored in a file. That file happens to be located on the web. Remember that, whether it is on the web or on your local machine, a file is just a file. The only difference is in how you open it.

Download files.py from the Labs page of the course website. It contains headers (with docstring comments) for functions that you need to implement. These functions analyze temperature data that is in a certain format.

1. Finish implementing the helper function open_temperature_file. It will need to use urllib.urlopen to open this URL:

> http://robjhyndman.com/tsdldata/data/cryer2.dat

This data file contains a description at the top ("Average monthly temperatures in Dubuque" etc.), before the actual data. As the docstring says, your function open_temperature_file should advance past that description in the file, so that when your next reads from this URL, it will get actual data.

2. The data is organized so that each row has 12 numbers, each corresponding to a month of the year. Finish implementing the function avg_temp_march and make sure that it produces the correct result for the data at the URL above. Of course, given how the data is organized, this means looking at the values in the column corresponding to March. Hint: Cast the result to a float along the way. **_Switch roles._**

3. Next, implement function avg_temp. It will do the same thing as avg_temp_march, but it is more general: it finds the average temp for any month, not just for March. Copy your avg_temp_march code to the avg_temp function and make a small change to it, so that it works with any month.

4. Implement the function higher_avg_temp. Call the function with the URL above (rather than with an already open reader) and open the URL in the function. **_Switch roles._**

5. Implement the function three_highest_temps. Call the function with an open reader for the URL above. Switch roles

6. Implement the function below_freezing. Call the function with an open reader for the URL above. Show your TA that it works. **_Switch roles._**

Analyzing Python Files: Bonus Material

There is lots to do in this lab, so you may not get to this section. If not, go back and complete it once the lab materials are posted at the end of the week.

Python files are just text files with a specific, detailed format. Whenever you run a Python program, the interpreter opens the file, reads it, and executes the commands in the file. The interpreter knows what to do because the Python language follows very strict grammar (or "syntax") rules, and the exact meaning of every type of statement has been defined. For example, the value to the right of a single = needs to be evaluated and then assigned the variable named to the left of the =. The process of reading a program and breaking it into recognizable pieces is known as parsing. This question asks you to parse a Python file, looking for specific Python structures.

1. Write a function find_function_names that accepts an open reader as input and returns a list containing all of the function names in the file. Use a for loop to process each line in the file. Remember that function definitions are signaled by the keyword def and that the function's name is followed by a list of arguments surrounded by brackets. Hint: You may find startswith, find, and slicing useful.

2. Write a __main__ block to test find_function_names. The block should open the Python file files.py that you wrote for the previous section, call the function find_function_names, and then write the list of function names, one per line, to a new file named function_names.txt. Do not use choose_file to select the file to open; simply provide the filenames to open as strs. **_Switch roles._**

3. Write a second function get_comments that accepts an open reader as input and returns a list containing each comment in the file. This problem is tricky, since '" potentially indicates a multi-line comment. Instead of using a for loop to read one line at a time, read the entire program into a str at the beginning of your function. Then, use find to locate all occurrences of # and '". The text between a # and the end of the line is a comment. The text between two occurrences of '" is also a comment. (It could also signal a multi-line string, but for this exercise, assume this is not the case.)

4. Modify the __main__ block to run the new function and place the output in a second new file named "comments.txt".

Remember to show your TA your work and to return your handout before leaving class. Be prepared to discuss how the various methods for reading from files differ.