CIS 480 - Python - Fall 2005
WEEK 7 LAB EXERCISE and Homework #5

week 7 lab exercise due: Thursday, October 6th, END of lab
HW #5 due: Thursday, October 13th, 12:00 noon

Until I develop more formal opening comment block standards (which I plan to do), begin **each python module** that you write with at least the following opening comments:

* a comment containing the name of the module (the **file name** of the module, please --- **lab06.py**, for example)
* a comment containing your name, and
* a comment containing the date that your module was last modified

WEEK 7 LAB EXERCISE

For this lab exercise, you are **encouraged** to check your answers with classmates. In fact, you are **required** to do so with at least one classmate who has not yet had his/her work checked yet before you have me check your work; write the name(s) of those you checked your work with below:

_____

(The point here is to have to argue, er, discuss with others why your answers are correct if they think they are not, or why others' are not correct if they think they are, but you do not...)

1. Answer the following questions in the space provided:

(a) What Python function returns a file object, ready to be used for reading, writing, or appending?

_____

(b) Write a call of your answer to **(a)** that will open a file in the current working directory named **tps_report57.txt** for reading, assigning the returned file object to a variable **tps_in**.

_____

(c) Write a call of your answer to **(a)** that will open a file in the current working directory named **letter_stats.txt** for writing, assigning the returned file object to a variable **letters_out**.

_____

(d) Now use **tps_in** in a statement that will read the next line from **tps_report57.txt** into a variable **next_line**.

_____

(e) And, use **letters_out** in a statement that will write the contents of the string variable **next_char_stats** into the file **letter_stats.txt**.

_____

(f) Write a the **two** statements that, according to course style standards, we should write when we are finished reading from **tps_report57.txt** and **letter_stats.txt**.

2. Write a function **make_spamfile** that will ask the user (interactively) for the name of a file (in the current working directory) in which it should repeatedly write the word 'spam', and then ask how many times it should write 'spam'. It should then write 'spam' that many times into a file of that name, following course style standards.

When this runs like you would like, check your work with at least one classmate, and then write your name on the 'Next:' list on the board to get your work checked.

All of the above must be completed before the end of your lab time.

HOMEWORK #5

Create a Python module **hw05.py**. Within it, include the following:

0. Import the module **hw04.py**; we'll be using some of its functions in this homework.

1. Remember **ct_letter_freq** from HW #4, problem #1? and **freq_bar_chart**, for HW #4, problem #2?

Write a function **show_file_ct** that:

* asks the user for the name of a file in the current working directory whose letter frequencies he/she desires,

* slurps all of that file's contents into a single string, [yes, I know this is dangerous if the file is too big. But, it's file method practice.]

* uses **ct_letter_freq** to create a dictionary of the letter frequencies from that file using that single string.

* prints to the screen a message including the name of the file, and then

* uses **freq_bar_chart** to then display the letter frequencies for that file.

This is very interactive, so you do not need to add tests to it to **hw05_test.py**. (You should test it yourself until you are satisfied that it works, however!) I'll just have to test it myself to verify this... 8-)

2. But, what if you'd like to keep these letter frequencies for some other purposes?

Write a function **save_letter_freq** that:
* expects three parameters:
  * a dictionary of letter frequencies, and
  * a string containing the name of the file from which these letter frequencies were computed, and
  * the name of a file (to be in the current working directory) in which these letter frequencies are to be saved.
* It should create a file (named based on the 2nd parameter) that contains:
  * on its first line: the name of the file from which these frequencies were computed,
  * on each subsequent line: one of the letters, then a space, then that letter's frequency.

Again, this is very interactive, so you do not need to add tests to it to **hw05_test.py**. (You should test it yourself until you are satisfied that it works, however!) I'll just have to test it myself to verify this... 8-)

3. Now that we have **save_letter_freq**, it would be nice to offer that option to our users of **show_file_ct**, wouldn't it?

ADD the following to your function **show_file_ct**:
* before it ends, ask the user IF he/she would like the letter frequencies saved to a file.
* If so --- it should ask for what file, and call **save_letter_freq** appropriately.

Again --- too interactive to easily test in a **hw05_test.py** (yet, anyway).

4. And, of course, we need some file-reading practice. So, write a function **freq_from_file** that:
* expects one parameter: a string, the name of a file in the current directory which you can ASSUME is properly filled with letters and frequencies (as produced by **save_letter_freq**).
* It should return a **dict** that is filled with these letter frequencies.
* (remember: for this function, at least, it'll ignore the "source" file name that's one the first line of this file...)

This isn't interactive, but there's no time to plug in the proper testing verbiage here, so you've lucked out. No hw05_out.py required.

5. And, a little more file-reading practice. Write a function **display_saved** that takes as its argument a file name, that can be assumed to be in the current directory and be formatted like **save_letter_freq** creates.

It'll do what it needs to do to call **freq_bar_chart** (from HW #4, #2) to display that file's frequencies as a bar chart --- AND precede this chart with the name of the file from which these were computed.

Again, you lucked out --- no **hw05_test.py** this week.

And, when you are satisfied, (and by the due date and time given at the beginning of this handout,) use **~st10/480submit** to submit your final version of **hw05.py.** (And remember --- you can submit more than one version of these before the deadline, if inspiration strikes after a submission. As the syllabus notes, I'll simply grade the latest version that was submitted before the deadline.)