Programming Project 06

This assignment is worth 40 points (4.0% of the course grade) and must be **completed and turned in before 11:59 on Monday, Feb 25th. Two weeks because of the midterm!!!**

Assignment Overview

The goal of this project is to gain more practice with file I/O, lists and functions.

Background

Data mining is the process of sorting through large amounts of data and picking out relevant information. It is usually used by everyone from financial analysts to scienstist to extract information from the enormous data sets. These large data sets and the trend of analyzing them has come to be know as "Big Data" http://en.wikipedia.org/wiki/Big_data

In this project, we want to do some preliminary data mining of the prices of Apple stock. Your program will calculate the monthly average prices of Apple stock from 1984 to 2013. You will report facts about the monthly highs and lows for this data.

Project Specifications

1. A file of Apple's daily stock's prices will be given to you, whose name is table.csv. This file could be opened by notepad or similar text editor, and is delimited by commas. If you open it with Excel, it will show you the data as a spreadsheet.

2. You must implement the following functions:

   a. get_input_descriptor()
   In this function, you are required to repeatedly prompt for the name of an input file until the user enters filename and the file can be opened for input. Return a file descriptor attached to the opened file.

   b. get_data_list(file_object, column_number)
   In this function, you are required to read the file of Apple's data. The function is flexible as it can read the data for any column of the data (0 through 6). If you read column 6, you are gathering the data for the "Adjusted Daily Close". If you read column 5, you are gathering data for the "Volume" that day. The function returns a list that consists of tuples. Each tuple is of the form: (date, column_data). For example: ('2013-02-08', 474.98) if we were collecting data from column 6.

   c. average_data(list_of_tuples)
   In this function, take in an argument that is the list of tuples generated by get_data_list above. You will average the data for each month, and regenerate a list of tuples. A tuple here will have the form: (data_avg, date). For example: (2972945.4545454546, '07:1985') . Note the date does not contain a day any more.

   d. main()
   In this function, you:
   - call get_input to get a file descriptor

- prompt for the column to average
- call the get_data function
- call the average_data function
- print the highest 6 averages (for the column selected) and the lowest 6 averages. Print that data with the month-year information.

Deliverables
proj06.py – your source code solution (remember to include your section, the date, project number and comments).

1. Please be sure to use the specified file name, ie. "proj06.py"
2. Save a copy of your file in your CSE account disk space (H drive on CSE computers).
3. You will electronically submit a copy of the file using the "handin" program: http://www.cse.msu.edu/handin/webclient

List of Files to Download
table.csv

Assignment Notes:

1. When reading the input file, you should be careful about the first line which does not contain any data.
2. Remember the split() function, which takes as an argument the character to split on, and returns a LIST of STRINGS
3. Don't forget to convert each string stat to a number.
4. Since there are so many fields, do some testing (E.g. output some parsed data) to make sure that you get the correct data.
5. The sorted function should be useful.

   my_list = [(3,2), (1,2), (2,5)]

   sorted_list = sorted(my_list) # sorted_list will be [(1,2), (2,5), (3,2)], sorts on **first** value in each tuple

6. Remember to close your files after you have opened them

   file_obj.close() # Very important to close the file!

7. To create a tuple, remember you just use a comma, so a 2-item tuple could be created like this:
   my_tuple = x,y
   When you store your date and monthly average, it would probably be easiest to store the date in a string already properly formatted, e.g. "11-2007".
   To append this tuple to a list you can just say my_list.append(my_tuple). Then to access the different items in the tuple you index into the list twice, so for example if you appended the above tuple as the first item in a list:
   my_list[0][0] would return x
   my_list[0][1] would return y

Example Output

```
In [121]: main()
Open what file:fred
Bad file name, try again
Open what file:irving
Bad file name, try again
Open what file:table.csv
What column:6
Lowest 6 for column 6
Date:08-1985, Value:  1.70
Date:09-1985, Value:  1.77
Date:06-1985, Value:  1.84
Date:10-1985, Value:  1.90
Date:07-1985, Value:  1.93
Date:11-1985, Value:  2.19
Date:05-1985, Value:  2.21

Highest 6 for column 6
Date:09-2012, Value:674.54
Date:08-2012, Value:635.39
Date:10-2012, Value:628.17
Date:04-2012, Value:597.19
Date:07-2012, Value:592.33
Date:03-2012, Value:569.11
```