CSE 231 Fall 2008

# 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, October 13th, 2008.**

## 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 business intelligence organizations, and financial analysts, but is increasingly being used in the sciences to extract information from the enormous data sets generated by modern experimental and observational methods. (from wikipedia: http://en.wikipedia.org/wiki/Datamining)

In this project, we want to do some preliminary data mining to the prices of Google stock. Your program will calculate the monthly average prices of Google stock from 2004 to 2008, and tell us the 6 best and 6 worst months for Google.

## Project Specifications

1. A file of Google stock's historical prices will be given to you, whose name is table.csv. This file could be opened by notepad or wordPad, and is delimited by comma. If you open it with Excel, comma will not be shown.

2. A template program will be given to you, which is used for your program's frame. There are three functions with their simple descriptions, and you are required to pad them to make them work correctly.

a. get_data_list(FILE_NAME)
In this function, you are required to read the file of stock's historical prices. You should use FILE_NAME instead of hard code "table.csv" in this function, that way if we wanted to use a differet table at any time we could just change the call to the function and not have to change the function itself. After reading each line, you will split it into a list, and append this list to another main list, suppose its name is "data_list". So, data_list is a list of lists, a.k.a. 2-D list. At the end of this function, return data_list.

b. get_monthly_averages(data_list)
In this function, you will use the data_list generated by the first function as the parameter. Use Date, Volume, and Adj Close to calculate the average monthly prices.
How to calculate the average price? Suppose one day's volume and close price are $V_1$ and $C_1$ respectively, then that day's total sales equals $V_1 * C_1$. We will use the "Volume" column for the day's volume and the "Adj. Close" column for the day's close. Now suppose another day's volume and close price are $V_2$ and $C_2$. The average of these 2 days is the sum of the total sales divided by the total volume. So, the average price of these two days is calculated in this way:
average price = $(V_1*C_1 + V_2*C_2) / (V_1 + V_2)$

To average a whole month you just add up the total sales (V*C) for each day and divide by the sum of all the volumes ($V_1 + V_2 + \ldots + V_n$)
For each month create a tuple with 2 items, the average for that month, and the date (you only need the month and year). Append the tuple for each month to a list (e.g. monthly_averages_list), and after calculating all the monthly averages, return this list. We use tuples here because once these values are calculated we don't want to accidentally change them!

c. print_info(monthly_averages_list)
In this function, you need to use the list of monthly averages calculated in the 2nd function. Here you will need to find and print (to a file) the 6 best (highest average price) and 6 worst (lowest average price) months for Google stock. You will print to a file named "monthly_averages.txt". You will first print a header like "6 best months" and then print the 6 best months, 1 month per line, from **highest** price to **lowest**, in the following format: month-year, average_price (to 2 decimal places).
You will then print a blank line and then another header like "6 worst months" and print the 6 worst months, 1 month per line from **lowest** price to **highest**, in the same format as for the best months.
Sample output can be found below.
This function does not return anything

3. If you don't call these functions, they are useless. Thus, you should write codes to call them. The steps are already given in the template program.

**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
template.py

**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. List's sort function and reverse function should be useful.

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

   Li.sort()    # Li will be   [(1,2), (2,5), (3,2)], sorts on **first** value in each tuple

   Li.reverse()   # Li will be [ (3,2), (2,5), (1,2)]

6. To open a file for output, remember:

   ```
   fileDescriptor = open(‘fName.txt’,’w’)

   fileDescriptor.write(‘stuffToWrite’)
   ```

   - note you can only write strings here, must convert everything else to a string in order to write it to the file!

   ```
   fileDescriptor.close()  # Very important to close the file!
   ```

7. To create a tuple, remember you just need () and a comma, so a 2-item tuple could be created like this:

   ```
                   myTuple = (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 myList.append(myTuple).  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:

   ```
           myList[0][0] would return x
           myList[0][1] would return y
   ```

**Example output for table.csv:**

```
6 best months for google stock:
12-2007, 693.76
11-2007, 676.55
10-2007, 637.38
01-2008, 599.42
05-2008, 576.29
06-2008, 555.34

6 worst months for Google stock:
08-2004, 104.66
09-2004, 116.38
10-2004, 164.52
11-2004, 177.09
12-2004, 181.01
03-2005, 181.18
```