Oscar Lopez

CSCI 250

September 29, 2019

LopezCompanyList Data Analyzer Analysis

**Purpose**: Allows a user to perform some basic stock data analysis such as exporting data to some file, searching a company stock symbol and receive information on that company, and getting the top 15 companies with the highest market capital.

Main functions in the program:

1. clean\_data(data)
2. parse\_file\_as\_dict(\*filename)
3. parse\_file\_as\_list(\*filename)
4. output\_data\_file(filename, data)
5. search\_symbol(symbol, dict\_data)
6. get\_top\_15\_marketcap(data)
7. display\_menu()
8. prompt\_choice(prompt)

**Steps involved in the main program**

* Before user intervention, data is parsed and prepared by using parse\_file\_as\_dict(\*filename) and passing in data file names companylist\_nasdaq.csv, companylist\_nyse.csv, and companylist\_amex.csv. Similarly the same is performed using parse\_file\_as\_list(\*filename)
* The menu then gets displayed for the user using function display\_menu().
* Program then runs in a while loop infinitely until a break is included. As it does this:
  + The program then prompts the user for a choice by calling promptChoice(‘> ‘)
  + If the user enters some random non numerical value, the user console prints an error message indicating that the user must enter some valid numerical value. The loop then continues and prompts the user for a choice.
  + If the user enters some numerical value that is not included in the choices made by the menu, the user console prints an error message indicating that the user must enter a choice from the menu provided. The loop then continues and prompts the user for a choice.
  + If the user enters 1, the user is then prompted to enter a file name. output\_data\_file(filename, data\_list) is then called and the user entered file name is used as the filename parameter of the function. Confirmation info is then printed to the user console informing that the file is now exported as the file name they entered.
  + If the user enters 2, the user is prompted to enter the company symbol they wish to look for. search\_symbol(company\_symbol, data\_dict) is then called and the user entered company symbol is used for the parameter ‘company\_symbol’. The result is stored in a variable named ‘result’. If result is None, the user console prints a message indicating that the company symbol is not found in the dataset. Otherwise, a message is printed on the user console displaying the results of the searched symbol.
  + If the user enters 3, get\_top\_15\_marketcap(data\_list) is called in a for loop as an enumerated list. A formatted string is then printed to the user console displaying the position where the company is among the 14 other companies, its company symbol, and its market cap.
  + If the user enters 4, an exiting message is printed to the user console and the loop is then broken by using ‘break’.

**Function Details and Logic**

1. clean\_data(data)
   * A simple data cleaning function that removes any unnecessary quotes, replaces “,” with tabs for easier delimitting of data, and removes the last comma on each line.
   * Runtime analysis: O(1) since all operations involved are of O(1) runtime.
2. parse\_file\_as\_dict(\*filename)
   * A function that parses one or more files in a dictionary of data.
   * Involves using the clean\_data function as it iterates through each line in the file and splits each line as a list. The list is then iterated and each element is defined as a dictionary key : value, which is returned at the end of the function.
   * Runtime analysis: Since the function must iterate through each line in the file and through each element in the line and parse it in a dictionary the runtime analysis here is o(n^2) .
3. parse\_file\_as\_list(\*filename)
   * A function that parses one or more files in a list of data.
   * Similar to parse\_file\_as\_dict, however instead simply appends the split line of the file to another list of data.
   * The aggregated list of data is then returned at the end of the function.
   * Runtime analysis: Still involves iterating through each line and iterating through each of its elements, so its runtime is o(n^2).
4. output\_data\_file(filename, data)
   * A function that simply writes data from a given list of data and append it to some output file with the filename entered by the user.
   * Since the data here is a list, it can easily be sorted using the sort function since the first element of the list is the company tag, which is the value that has to be sorted in alphabetical order.
   * Uses two for loops to iterate through each line in the file and through each element of the line to append to the file.
   * Runtime Analysis: Although the sort function itself has a default runtime of o(nlogn) thanks to quicksort, the whole function requires two for loops and iterates through each element of the list, so the runtime here again is o(n^2).
5. search\_symbol(symbol, dict\_data)
   * Simply looks for a key that matches the symbol given by the user in the function.
   * If the key is found, the value then goes through its dictionary value and returns its details in a formatted string.
   * If the key is not found, None is returned.
   * Runtime Analysis: Since dictionaries are hashmaps, hashmaps generally have a runtime of o(1), so the runtime here is o(1) since nothing is iterated through and simply looks for a key that matches.
6. get\_top\_15\_marketcap(data)
   * This function goes through a list of data and appends each company’s symbol and market cap into a list.
   * This list is then sorted using the sort() function.
   * After sorting the data a for loop iterates through the first 15 lines of data and appends it to another list called top\_15\_marketcap. This list is then returned at the end of the function.
   * Runtime Analysis: Since the function must iterate through each line of data to parse and sort it later on, the runtime here is o(n).
7. display\_menu()
   * Simple function that returns a String representing a menu.
   * Runtime Analysis: o(1).
8. prompt\_choice()
   * Simple function that returns the integer value of input from the user.
   * If no integer value is given, it returns None.
   * Runtime Analysis: o(1).