

Making a Menu Driven Stock Tracking App

In this assignment you will be competing the stock tracking application we started in class and adding improvements to the application. You will start with the framework code we developed in class. Currently the application requires the user to first enter a list of stocks to track since we did not add any mechanism for data persistence. The completed app will allow continue to run until the user exits and will provide the following functionality:

1. Select from among a number saved tracking lists to begin following; assume you can only track one list at a time
2. Add new lists to track, which should automatically be saved for future application runs
3. Delete a tracking list
4. Edit an existing tracking list (add or delete stock tickers from a selected list)
5. Exit the application

While the application explicitly requires you to write or refactor code to meet new requirements, consider that some of these tasks will start off similarly. For example if I am going to select a list to track, then first I will have to see all of the available lists. Likewise, if I am going to edit or delete a list, I will also need to see all of the available lists. Since all of these functionalities start the same way, it makes sense to create a “helper” function to “show_lists”, which can be called from within track, edit and delete. Additionally, both add_list and edit_list require save list functionality. This make a function that saves files another good candidate for a helper function.

In terms of saving lists it is recommended that you create a directory (think os module) explicitly for this within the application, perhaps naming this directory watchlists. This will make it convenient to run the application from any computer. *Note: be sure your application only creates this directory if it does not currently exist.* Watch lists should be saved to .txt files in the watchlist directory.

Make the following additions and improvements to the application

Modify the track function: when the user selects this option, they should be presented with an enumerated list of watch list names from which to choose. In the special case that no watch list exists in the watchlist directory, the application should print a message saying so suggesting the user add a list, and then return to the menu.

Modify the add_list function: the completed function should save the watchlist out to a .txt file in the watchlist directory. Once the user has added all the desired symbols, the function should prompt them for a name to save the watch list as. *It should not allow the user to save a watch list to a file name that already exists*

Delete a list: users choosing this menu option should be presented with an enumerated list from which to delete. Users should enter the number corresponding to the list that wants deleting. Once making a choice the application should confirm that the user actually has chosen the correct list to delete, and should start the process over if they do not confirm.

Edit a list: this function should start by displaying a list of watch lists available for editing. Once the list has been chosen, the user will then need to choose whether to add or delete symbols. Once they choose to add or delete, they should be prompted to continue adding or deleting until perhaps entering an empty string. When choosing to delete, the available tickers should be presented as an enumerated list so the user will simply choose the number of the symbol to delete. *Be sure to save the revised list out to the text file with the same name as the original.*

Sample interactions:

```
StockTracker Menu
1. Track watchlist
2. Edit watchlist
3. Add watchlist
4. Delete watchlist
5. Exit

Enter your selection: >? 1
1 - babo
2 - china
3 - default
Enter list to track
>? |
```

Figure 1. User selected 1 from main menu, now needs to choose a list to track.

```
StockTracker Menu
1. Track watchlist
2. Edit watchlist
3. Add watchlist
4. Delete watchlist
5. Exit

Enter your selection: >? 3
Enter a symbol: >? f
Enter another symbol: >? ge
Enter another symbol: >? tsla
Enter another symbol: >?
Enter a name for watchlist
>? auto|
```

Figure 2. User chose menu option 3, entered symbols, signaled they are done with an empty string. Now they are prompted to enter a file name for the new list

```
Quit, or continue? >? q

StockTracker Menu
1. Track watchlist
2. Edit watchlist
3. Add watchlist
4. Delete watchlist
5. Exit

Enter your selection: >? 4
1 - auto
2 - babo
3 - china
4 - default
Enter list to delete:
>? |
```

Figure 3. User has chosen option 4 from the menu and must now choose a list number to delete