Austin Biehl

August 10, 2020

Foundations of Programming: Python

Assignment 05

# Modifying a To-Do List Manager

## Introduction

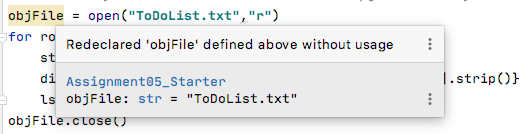
Throughout this document, I will be reviewing the script that I modified to manage a to do list for the user, allowing them to display, add, save, and remove items. This assignment was unique in that I started with a pre-created script framework, and then modified and added to the existing code in order to create a new program for the user. This week’s assignment also introduced us to GitHub as a platform for collaborating on others’ code.

## Writing the Script

This week’s assignment was to modify a preexisting script that was similar in function to week four’s program. The script came with some predefined variables, as well as some pseudo code that described the general objectives and steps needed to complete the program. Similar to last week, the program needed to allow the user to add, display and save their current information. However, this week we were asked to add an additional functionality—removing a piece of information that had previously been added.

The code that we were modifying and adding to contained a few features that I find unusual, or that I had at least not seen in the past. For example, in the first section where the variables were being defined, we were given objFile = “ToDoList.txt”. Previously, I had always created the file fresh during the processing section of the program.

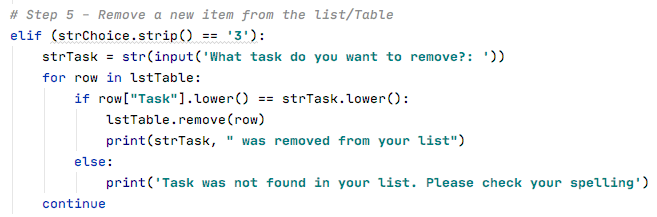
When I went to write this section of the program, PyCharm displayed a message letting me know that the objFile variable had already been defined (Figure 1).



***Figure 1: PyCharm suggestion regarding redeclared variable.***

Since the program still ran properly, I chose not to resolve this, as it seemed like it be more a stylistic choice than anything else. However, it did make me wonder about the best practice of file naming when writing programs of this nature.

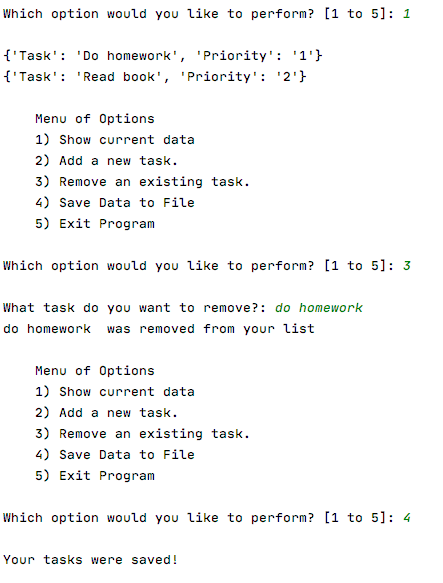
The most difficult portion of the assignment for me was enabling the program to remove a task from the list that the user had created. Past assignments have focused on creating and storing information, so it was a new challenge to need to remove something. However, I finally realized that adding removing a row required basically the same code as adding a row (Figure 2).



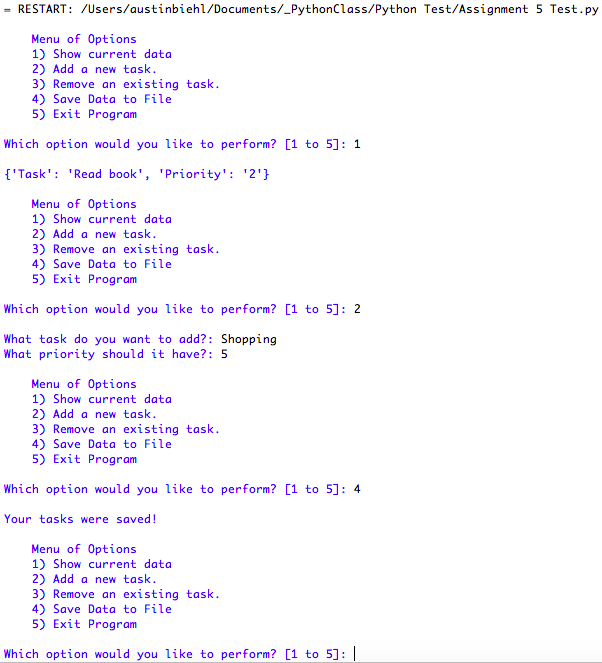
***Figure 2. Removing a row from the table***

As you can see, I used the same strTask variable that I’d used when capturing a task from the user. However, I then was able to check if the task that they entered was already in the table using if row[“Task”].lower() == strTask.lower(). If that statement was true, I was then able to remove the row that that task occurred in using lstTable.remove(row). This was a good lesson in remembering not to overcomplicate problems, and iterate upon preexisting code to solve new problems.

With this problem solved, the rest of the tasks were fairly straightforward and similar to the previous week and I was able to come to a final solution (Figure 3).



***Figure 3: Program running in PyCharm***



***Figure 4. Program running in Python Shell***

## Summary

While I did come to a functional program, I cannot say that it seemed particularly user friendly. One thing that I’m realizing with the use of the while loop is that it forces the user to look at information that they may not need. For example, the menu is repeated multiple times, which takes up a lot of room and makes the program feel unwieldy. I also thought that it would be nice if there was some way to order the list, or have the program sort the tasks from highest to lowest priority. Moreover, the current iteration of the program allows the user to put in tasks with the same priority, which doesn’t seem very helpful for the user. Overall, I am looking forward to learning about ways to further customize the user’s experience when interacting with Python programs.