George Biehl

August 10, 2020

IT FDN 110 A Su 20: Foundations of Programming: Python

Assignment06

# Functions and Classes

## Introduction

This assignment modifies the To-Do List script from Assignment05. The script is altered to make use of functions and classes. For example, instead of beginning the main body of the script with reading the file, now named ToDoFilet.txt into memory, the program instead calls a function which does the job. In a similar manner all the menu choices are also performed by functions.

The functions are also grouped into two logical classes. A “Processor” class holds functions which

read from, and write to, ToDoFile.txt and change the data held in memory. An “IO” class interacts with the user - printing to the screen and receiving user input.

## Creating a Stub Script

Before added any serious code, I first reviewed the logic. There were some changes from the starter file we used in Assignment05. In addition to the function calls, the file name had changed slightly, and the menu choices were different.

With the file name corrected, the program ran without error so it was a stub program. However, not all the functions provided any output to the screen to show that the function call actually happened and that control returned to the main body afterward. I added some print commands to make the operation clearer. As an example, I added the following code just before the “read\_data\_from\_file” function returned:

print(**'still in function - the file just closed after reading'**)

In the main program I added a similar line to confirm control had gotten back to the correct point in the main program. Once I was satisfied with the flow these print commands were removed.

## Expanding from the Stub Script

Since we were given the complete read\_data\_from\_file function, I next focused on its counterpart – the write\_data\_to\_file function. I reasoned that if the writing process produced a file which matched the original, I would be assured that my writing function was correct.

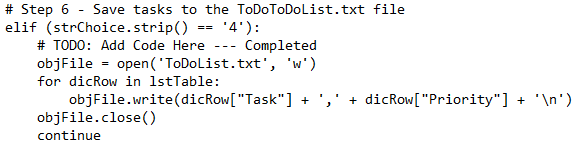
Shown in Figure 1 is the code from Assingnment05 which performed the writing-to-file function for the same data structure (a list of dictionaries).

Figure 1: Code for writing to a file in Assignment05

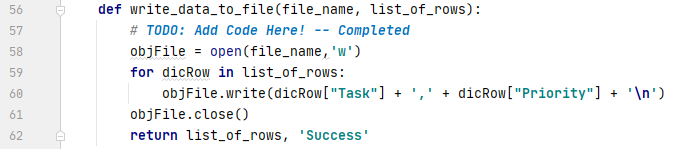
The four lines in the center dropped right into the write\_data\_to\_file function as shown in Figure 2. I only needed to change the name of the file and the name of the table.

Figure 2: Code for writing to a file in Assignment06

In a similar manner for most of the functions, pieces code from Assignment05 moved from the main script to their respective function. I had more difficulty when I was writing the function for Option 1 – Add a new task. Although the course video and notes did say it, it was still a while before I realized that two variables listed in the return statement automatically created a list for use in the main script. I also had difficulty with Option 2 – Removing a task. Again, the lecture had talked about for loop problems but, yet again, I failed coding the task deletion until getting more literal instructions.

In creating the functions I did not employ any global variables. In most cases I adjusted the variable names inside the function so as to make their local nature clear.

## Additions to the Basic Script

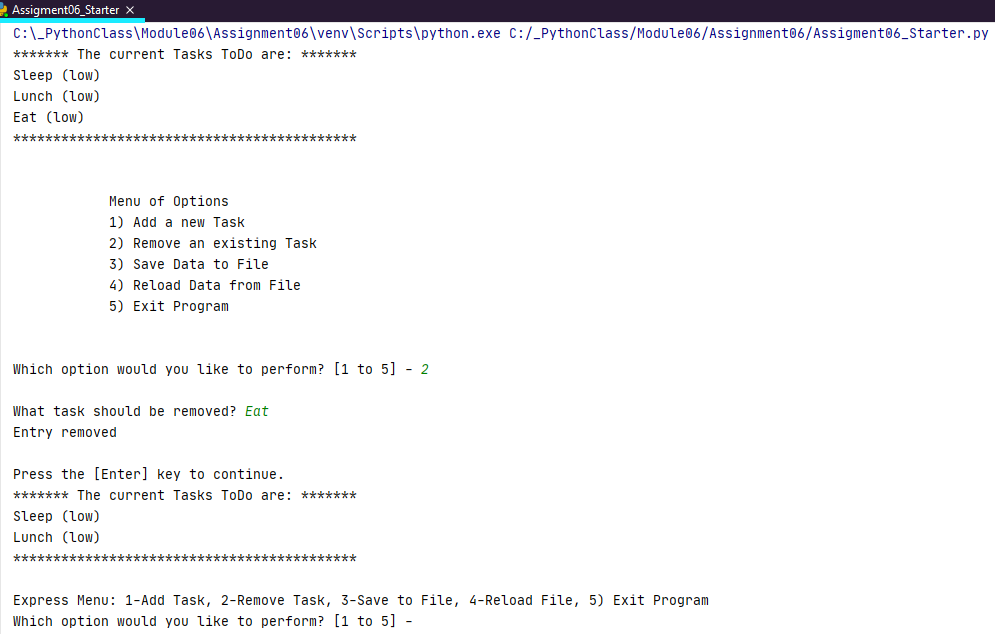
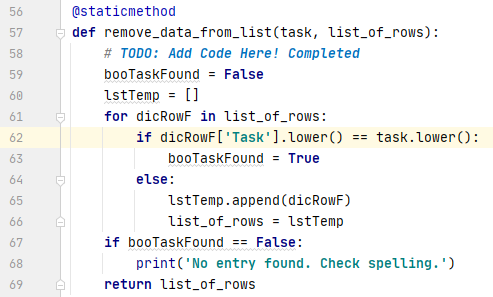
The “Menu of Options” consumed 7 lines for printed plus blank lines before and after. I altered the print\_menu\_Tasks function so as to show this menu only once. On subsequent iterations the user would see only a condensed, one-line menu as shown in Figure 3.

Figure 3: The Express Menu prints after the full menu has been shown once

After choosing “2) Remove an existing Task,” the user’s entry might not match a task already in the list due to a capitalization difference – “Sleep” vs “sleep” for example. I temporarily adjusted both the entered “task” and the table entry to lower case before comparing them to solve this problem. I also entered a message if the entry was not found. Code changes are shown in Figure 4 and output examples are in Figure 5.

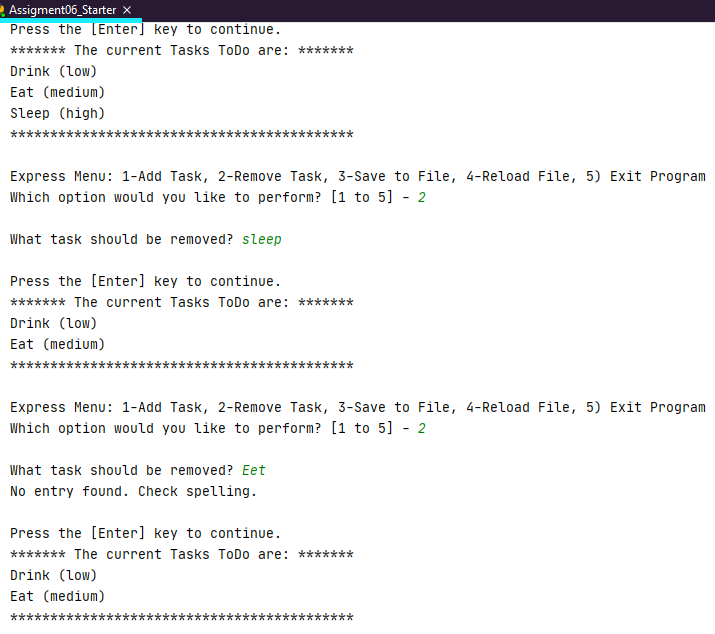
Figure 4: Task removal function handles capitalization and spelling errors

Figure 5: Option 2 with capitalization and misspelling errors

## Sample Output from a Command Prompt

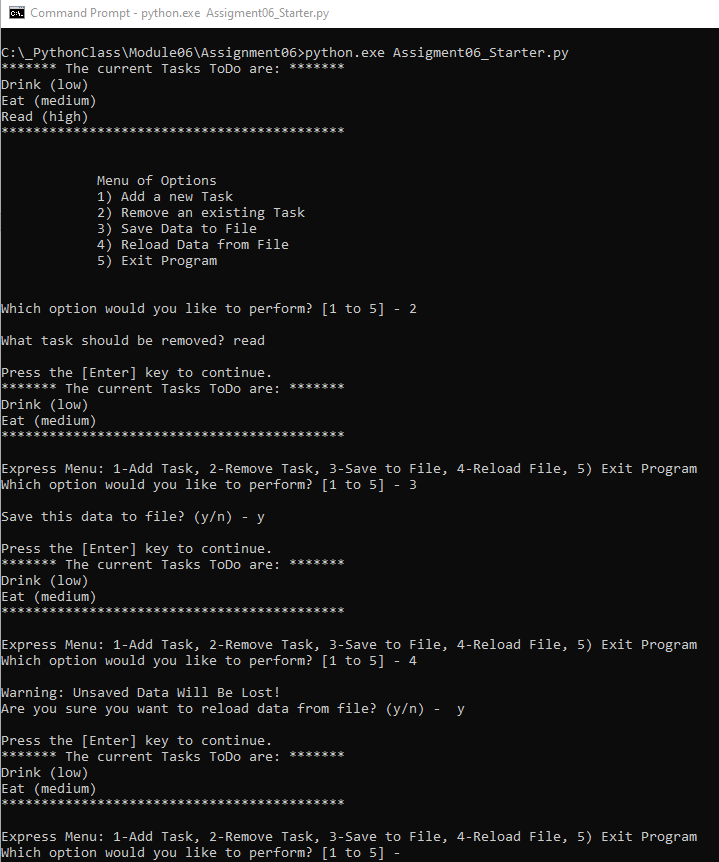
Figure 6 shows the output running the script from a dos prompt.

Figure 6 – Script running in an OS shell

## Summary

For this assignment most of the function were straight-forward to create. I did (and still) have difficulty with the “Remove an existing Task” function. The code I used works and I see why it works but I’m not clear on why the simple for loop I originally had didn’t work. Also, I’m not handling the return correctly. Ignoring the “success” message was necessary as I ran low on time!

I did make the program more robust by allowing for spelling and capitalization errors in the “Remove an existing Task” function. I also added a much shorter menu to improve usage of screen space.