Lin Martin

August 9, 2021

IT FDN 110 A

Assignment 05

# <https://github.com/LinMartinN7/IntroToProg-Python>

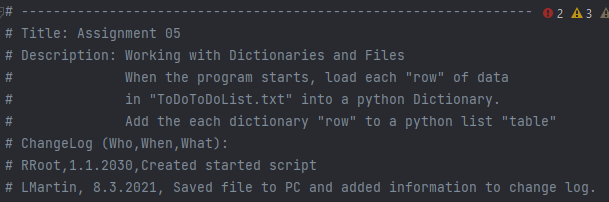
# Working with Dictionaries and Files

## Introduction

In the fifth week of Foundations of Programming: Python, our class was tasked with creating a script that would store a To-Do List. This list must capture tasks from the user and the priority of each task. It also must allow users to review the task they have put in, delete erroneous tasks, and save the file to the computer.

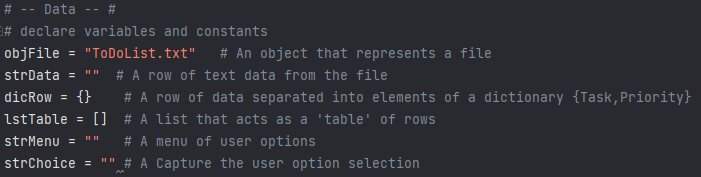
## Setting up the Script

This week, our class was provided a starter script to use for the assignment. The purpose of this was to gain experience working with code that had been created by another developer. After downloading the Module05 zip file, I extracted the files within and saved them to my One Drive folder where I store all my course data. Ext, I copied the Assignment05\_Starter file and placed it directly into the \_PythonClass folder. I opened the file in PyCharm and added my information to the header.



***Figure 01: Adjusted header to reflect my interaction with the Assignment05\_Starter.py file for Assignment 05.***

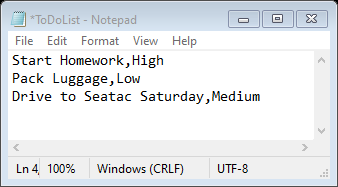
The variables and constants were already declared in this file, so it was not necessary for me to re-create this part of the script. I proceeded further down the code and began working on Step 1.



***Figure 02: Previously declared script variables and constants by Professor Root.***

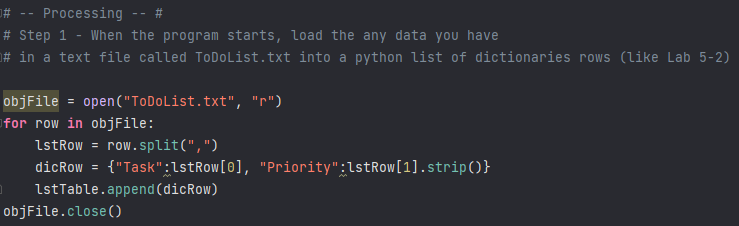
## Loading Data from Pre-Existing File

For Step 1 of the assignment, we had to load any pre-existing data that the user may have in their To-Do List. I decided to create a text file with some sample data for this assignment to test the functionality of the script once it was finished. I created a text file called ToDoList.txt and added a few items to it, including priorities.



***Figure 03: ToDoList.txt sample data to test the Assignment05\_Starter.py script file.***

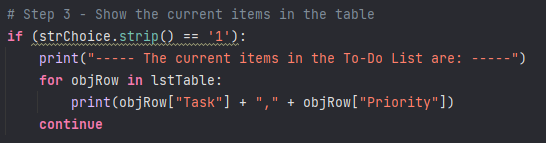
Next, I added code to the script that would open the ToDoList.txt file and read the existing data back to the user.



***Figure 04: Code to read data from ToDoList.txt sample in the Assignment05\_Starter.py script file.***

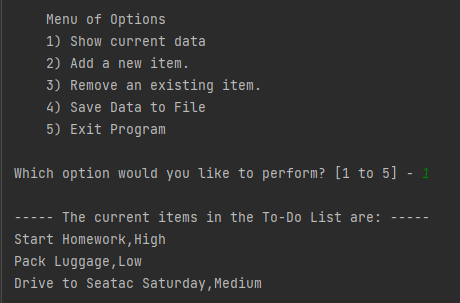
## Showing Current Data

Because this was an existing script file, I was able to skip Step 2. The user menu was already created and populated with choices, so there was no need to re-create it. I proceeded to Step 3 and began to add the script to add a new item to the table. I tested the script in PyCharm to verify that the file was able to read the To-Do List data from the sample .txt file.



***Figure 05: Script having Assignment05\_Starter.py display data from ToDoList.txt sample file.***

I decided to add a print statement at the beginning of the script, allowing the user to see that the items shown were currently in the ToDoList.txt file. While not necessary, I felt this gave the script a friendlier user interface.



***Figure 06: Successful test of Assignment05\_Starter.py script reading data from ToDoList.txt sample file.***

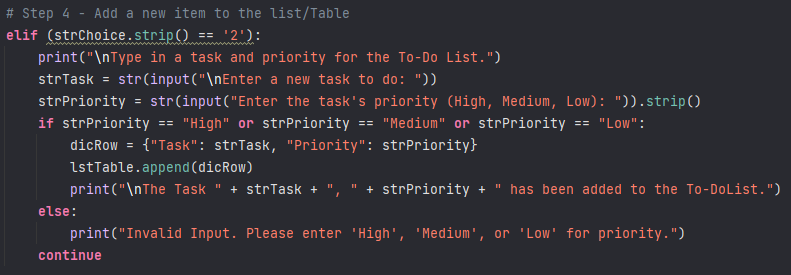
## Adding New Items to the To-Do List

For Step 4, we were tasked with creating an option to add new items to the To-Do List for the user. This section proved challenging for me, and but I managed to create the script after a couple of tries. Again, to make the script more user friendly, I decided to add a print statement showing the user what had been added to the script file. This feature also would help the user catch errors when entering in the task.



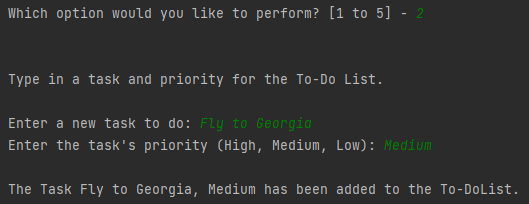
***Figure 07: Print statement showing the task and priority that were added to the ToDoList.txt file.***

After finishing my code, the grades for Assignment 04 were released. I decided to implement one of Sophia’s suggestions from this assignment when she graded it, which was to add code to force the users to type in the correct option when saving the document. I took this advice and applied it to Step 4, specifically where the users needed to enter in a priority rank for the task created. I believe this feature is great for maintaining data integrity, reducing time the user must take to identify and correct the error later.



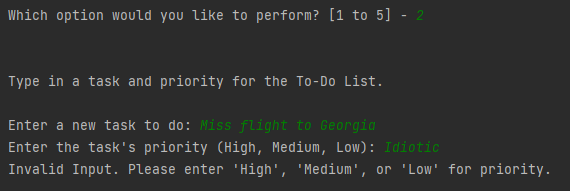
***Figure 08: Script for task and priority input values.***

Once I had finalized this section of the script, it was time to test this option within PyCharm. I first ran the script and entered in a task with the correct priority ranking to verify it worked. The script ran successfully.



***Figure 09: Successful test of incorrect input priority message.***

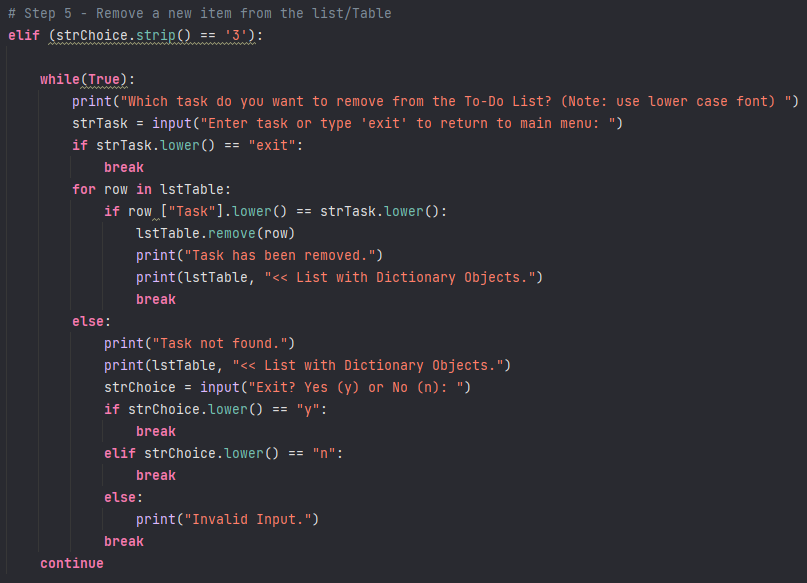
Next, I tried to add an item with an incorrect priority rank to see if I would catch the error. It worked, telling the user that the input was wrong and re-directed them to the menu.



***Figure 10: Successful test of incorrect priority message being input by user.***

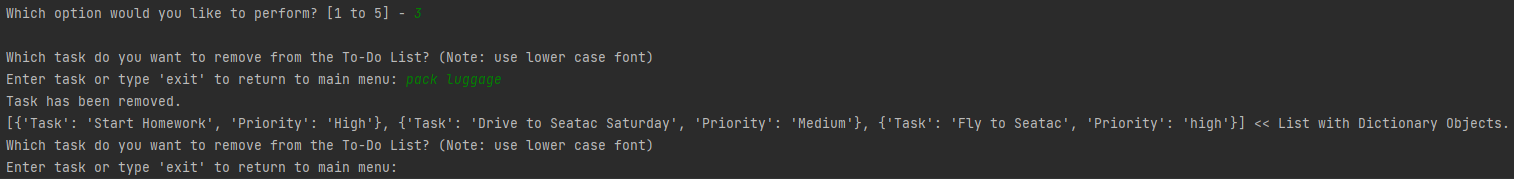
## Removing Items from the Table

The next option in the menu is to remove items from the list, which is Step 5 of the script. For this section of the assignment, I decided to make my code more advanced by allowing users to select which task they wanted to delete instead of automatically deleting the most recently created item. This allows the user to have more accessibility with the script, being able to delete out of date tasks.



***Figure 11: Script for removing items from the To-Do List.***

Next, I tested the script by attempting to delete an item from the To-Do List.



***Figure 12: Successful test of removing task from To-Do List.***

I verified that the task **Pack Luggage** was deleted by comparing it to the original list of tasks.

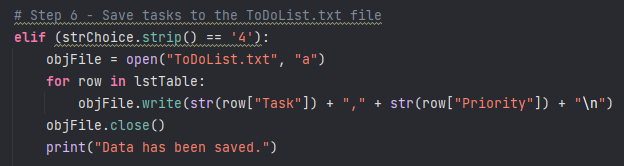
|  |  |
| --- | --- |
| **Original List** |  |
| **Revised List** |  |

***Figure 13: Original and revised To-Do List items.***

Working on this section of the script proved to be challenging. With my original version of the script, the tasks were not being registered by the deletion prompt if they were not the first item in the list. I did not realize it was only recognizing the first task in the file because I was so fixated on it not deleting tasks. It wasn’t until quite some time had passed that I consulted with my girlfriend about the problem. She quickly pointed out that I was missing a break between the *while(True)* and *else:* statements, and that my *else:* statement needed to be pulled back one level of indentation to be recognized by the script. While incredibly frustrating, this was a good reminder that sometimes the errors are quite small and hiding in plain sight.

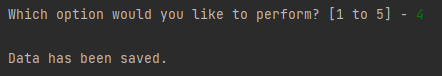
## Saving the Data

The fourth option of the script was to allow users to save their data to the ToDoList.txt file. This was Step 6 of the script and I felt very comfortable writing this section of code. Like previous assignments, I wrote the code to save the two pieces of data, *Task* and *Priority* into the file with a comma separating them. I also had the file save the data as a script so it could be retrieved later, and to create a new line after the new entry to prevent data from being overwritten in future attempts.



***Figure 14: Script code to save file to ToDoLIst.txt file.***

With the code written within Step 6, I decided to test it from the menu command within PyCharm. It appeared to work correctly, receiving the prompt that the data had been saved.



***Figure 15: Testing script in PyCharm to see if data was saved.***

I chose to double-check that the data was saved and opened the ToDoList.txt file directly from within PyCharm. It was here that I realized the file was indeed saving the data, but it was duplicating the tasks by re-exporting the entire list to the file. I changed the append “a” option to write “w” to test whether the records would be duplicated again. After changing the ToDoList.txt file data to match the original text, I ran the script again. This time the file was correct, only listing the three tasks that showed in the script.

|  |  |
| --- | --- |
| **Previous List with Append Option** | **Corrected List with Write Option** |
|  |  |

***Figure 16: Corrected To-Do List script results within ToDoList.txt file from append “a” to write “w”.***

## Exiting the File

The last part of the script was to allow the user to exit the program, which was Step 7 of the code. This section of code was the easiest to complete. I added a print statement to let the user know that the file was being prepared to be closed. I added a *EndProgram* command to prompt the user to press enter to close the file. I thought this would be a nice feature to have because it adds to the user interface.

## Testing the Script in Command Prompt

Once the script was completed, I saved it and proceeded with testing the file in Command Prompt. The first few times I tried to run the script, I kept receiving an error about the ToDoList.txt file not existing or being open. I remembered that when testing in the command prompt, I need to copy the text files to my user account on the C Drive before it will work properly. Once I copied the file over, the test worked perfectly. Command Prompt ran the script correctly and the file was updated with the changes made.

|  |  |
| --- | --- |
| **Opening the File** |  |
| **Show Current Data** |  |
| **Add a New Item** |  |
| **Remove Existing Item and Exit Option** |  |
| **Save Data to File** |  |
| **Exit Program** |  |

***Figure 17: Testing Assignment05 script in Command Prompt.***

## Summary

With Assignment 05, I think my struggles with the assignment have helped me come to understand the purpose of this script more clearly. Much of my struggles were due to small erroneous things, such as spelling errors, indentation errors, and capitalizing things that needed to be lower case. I know that as time goes on and I gain more experience with Python, these will be less troublesome items in the future for me.

However, having these struggles did force me to conduct research the issues online, re-watch the Module 05 video and class lecture multiples times, which ended up helping me understand the coding section of this assignment more thoroughly. I am also thankful that I had my girlfriend as a soundboard for my struggles during this assignment. Her input helped me understand that I was overthinking the problem, which ended up wasting a lot of time because I could not see how small the error truly was.

I am nervous and excited for the next assignment and look forward to reviewing my peer’s work in the group discussion. I have previously used GitHub for Professor Root’s SQL programming course, so I am comfortable uploading my script and course document on the site.