George Biehl

August 8, 2020

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

Assignment05

# Lists and Dictionaries

## Assignment Overview

This program reads a file into memory, stores the information in a particular manner, responds to user choices for viewing and altering the information, and finally saves the information back to the original file.

Reading a text file into memory assumes the file exists and consist of lines of data. Each line holds a “task” and a “priority” separated by a comma and each line ends with a carriage return. As each line is read from the file, a new dictionary pair is created. In addition, the comma and carriage return are removed. When the reading process is finished, memory holds a list of dictionary pairs which can be conveniently thought of as a table (each pair forming one row).

Once this table is created the user is presented with a menu of choices – display the table, add an entry, remove an entry, save the table back to the file (overwriting the original in the process), and to exit.

## Creating the Text File

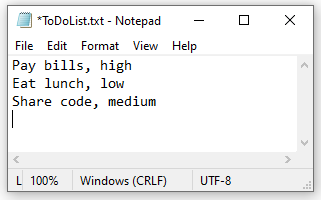
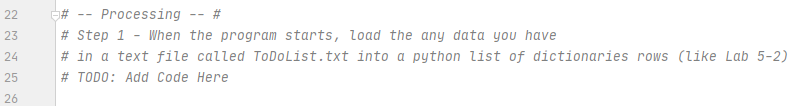
The text file I used was named ToDoList.txt and is shown in Figure 1. While this seemed trivial, I did get caught at file by not have a carriage return at the end of the third line.

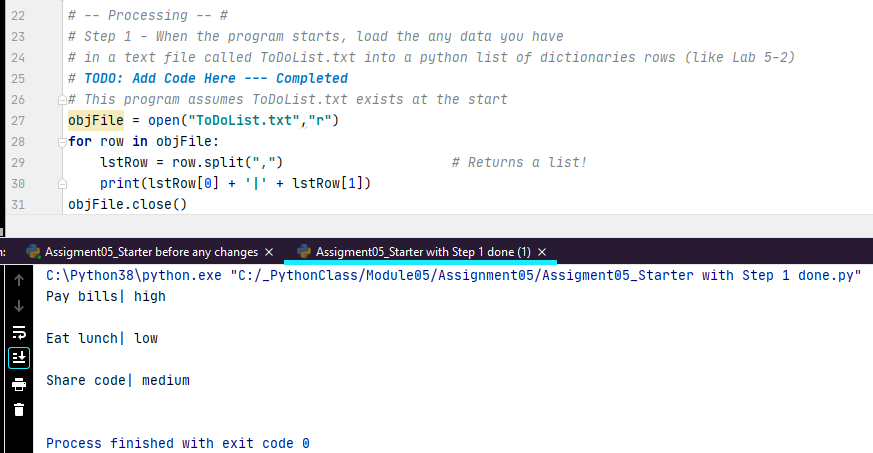
Figure 1: Text file

## Building on the Assigment05\_Starter.py File

The assignment simulated receiving a stub program with the requirement to add code only in the indicated locations. I first ran the given stub program to be sure that it contained no errors. I then began to work on each of the “Add Code here” requirements in order.

As an example, the first “Add Code here” occurred in Assigment05\_Starter.py on line 25 as shown in Figure 2. At this early stage I focused on simply making sure I was actually reading the file (!) so I didn’t worry about the getting it into the correct format in memory. It worked as shown in Figure 3. Later, I came back and re-coded this section so that the correct list of dictionary pairs was created in memory along with stripping off the commas and carriage returns (Figure 4)

Figure 2: First occurrence of “Add Code here”



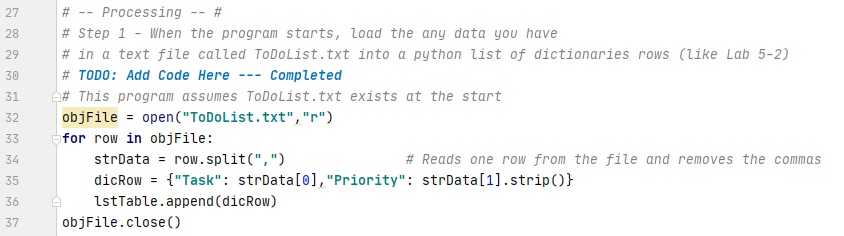
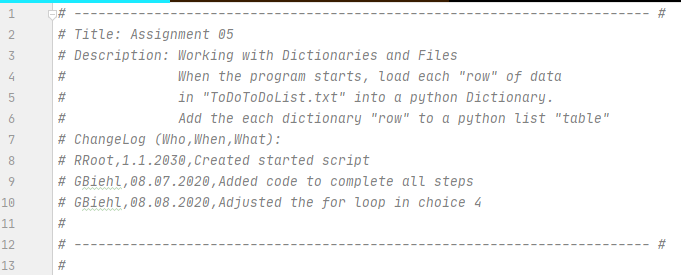
Figure 3: First attempt at reading from the file and printing data

Figure 4: Rewritten code to create required list of dictionary pairs

After completing each of the “Add Code here” parts my program was completed as shown in Figures 5-7.

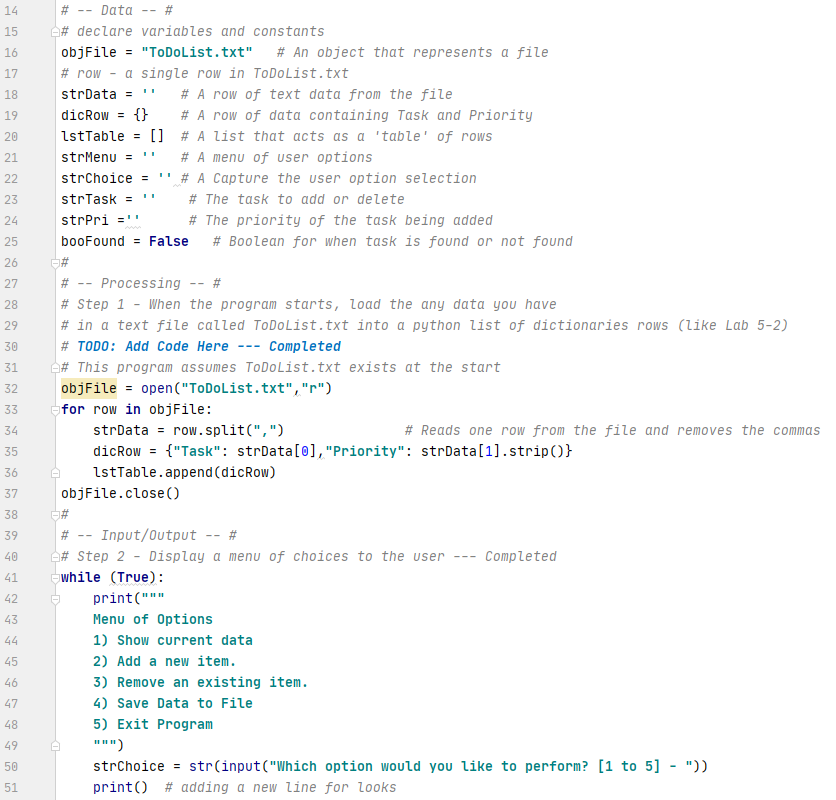
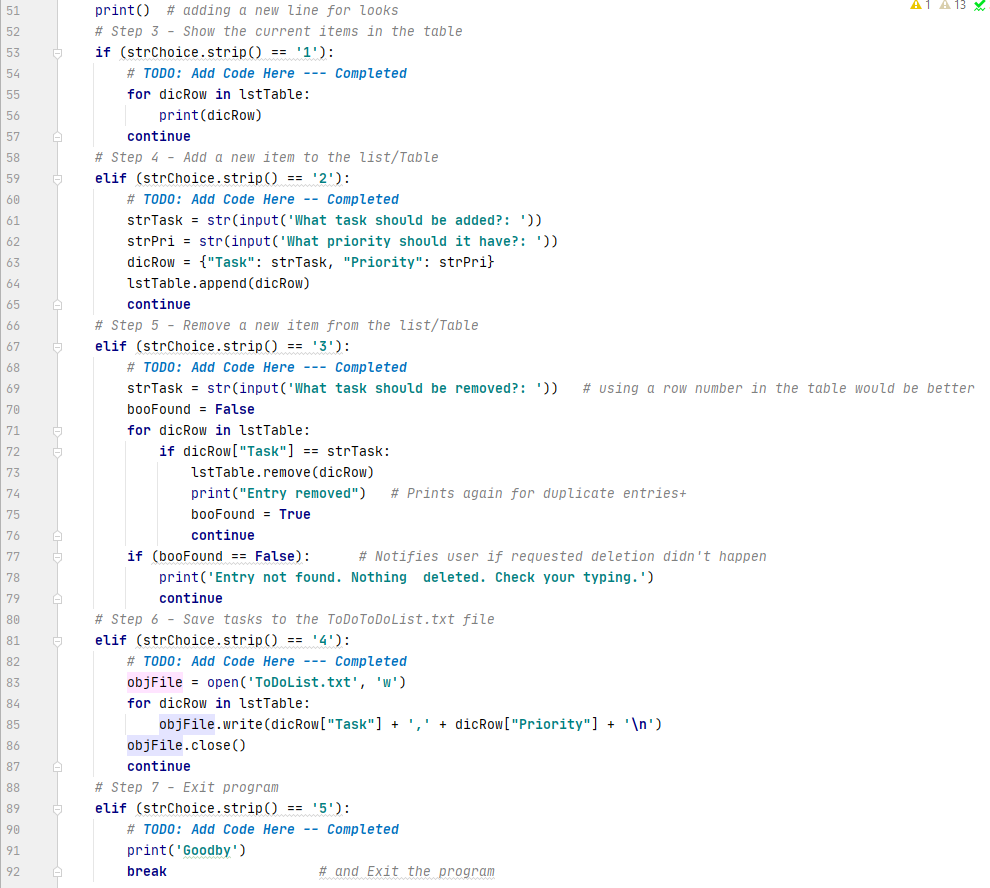
Figure 5: Program header

Figure 6: Code sections which read the file and present options

Figure 7: Code sections which respond to menu choices 1-5

## Demonstrating Program Operation

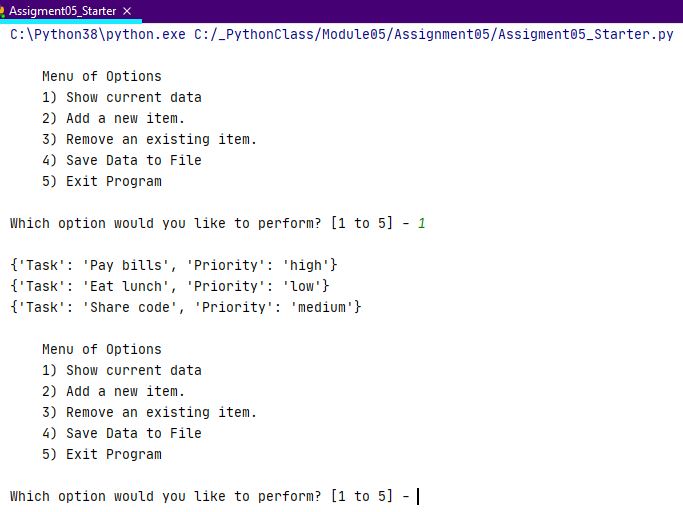
Figures 8-15 show the results of testing each menu option using the ToDoList.txt shown in Figure 1.

Figure 8: Python – Show Current Data

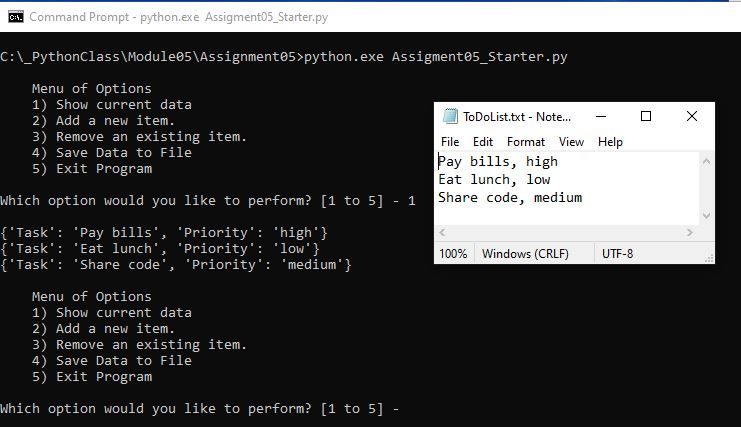
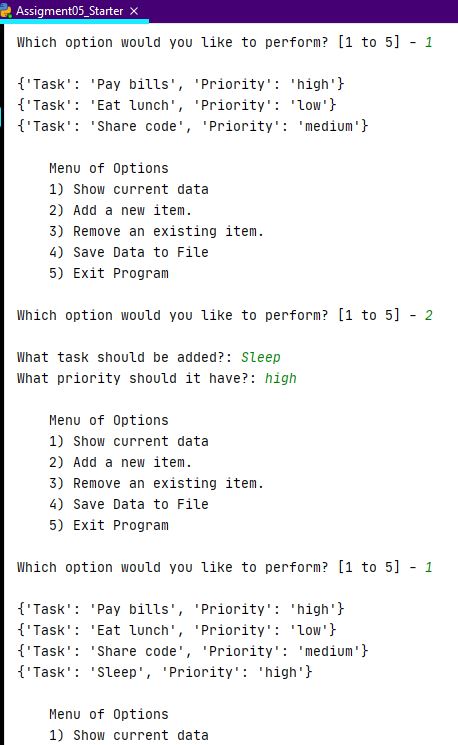
Figure 9: Command Line – Show Current Data

Figure 10: Python – Add a new item

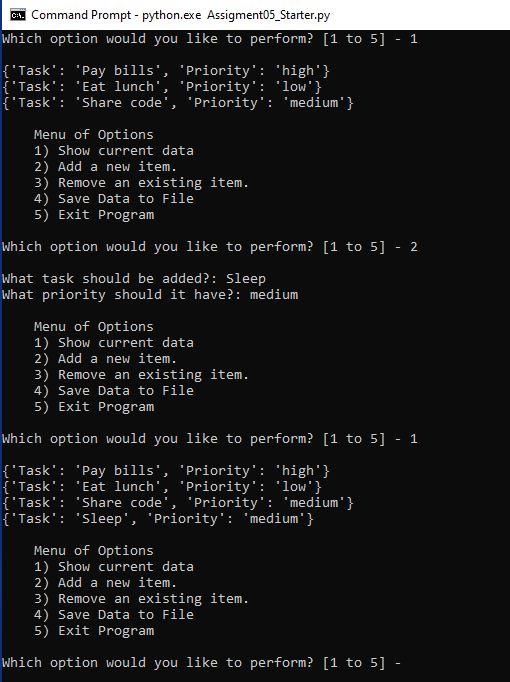
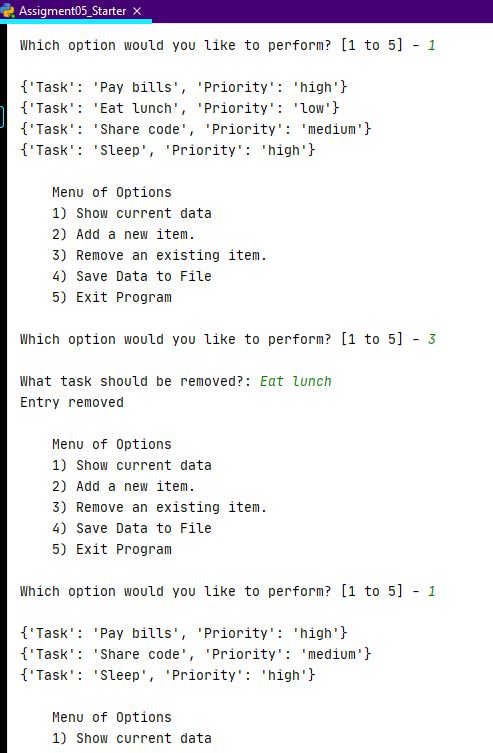
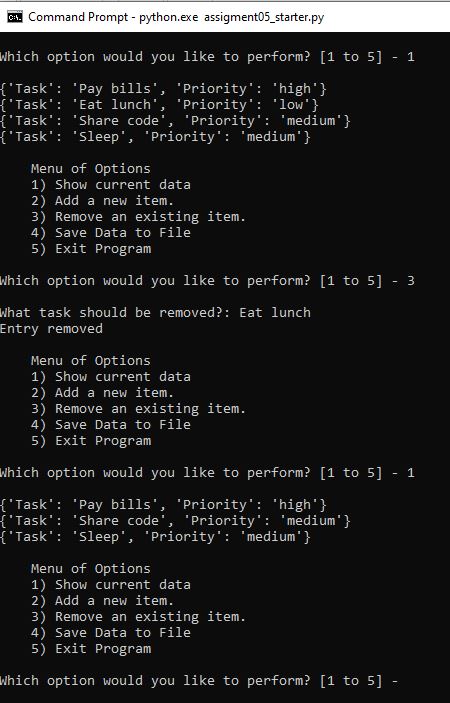
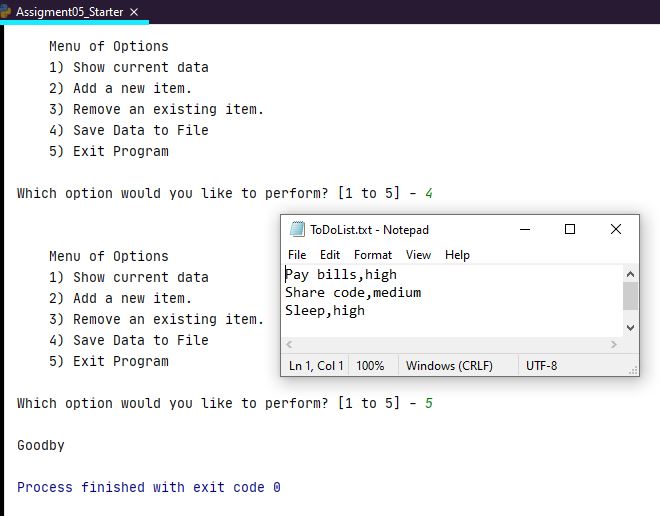


Figure 11: Command Line – Add a new item

Figure 12: Python – Remove an item

Figure 13: Command Line – Remove an item

Figure 14: Python – Save data to file and Exit

## 

Figure 15: Command Line - Save file (insert shows saved file) and Exit

## Summary

The program works as required but there are a few features that I would improve. The menu choices take up a great deal of space. The cycle of display menu + show current data almost fills the output screen by itself making it hard to note changes after additions or deletions.

I would also like to see the rows have numbers so that, especially for deletions, the user is not asked to correctly type the task where small misspellings of a task require a whole additional cycle of menu, show data, menu, now try the deletion again. Typing just the “row to delete” would be much easier. Adding row numbers would mean changing the table structure, of course.