Chelsea A. Wright

May 11, 2019

Python – Fundamentals of Programming

IT FDN 100 A

Assignment 05

Interactive To Do List

# Introduction

This document describes the steps taken to create an interactive To Do list, which allows the user to select various options to edit and display the To Do list. This program makes use of a two-key Python dictionary which organizes data into key-value pairs. Users are offered the following options to edit and display the dictionary items, which are stored in a table:

1) Show current data

2) Add a new item

3) Remove an existing item

4) Save Data to File

5) Exit Program

The sections below describe creating a text file with data, writing this data to a dictionary and scripts for each of the options listed above.

# Starting the Assignment

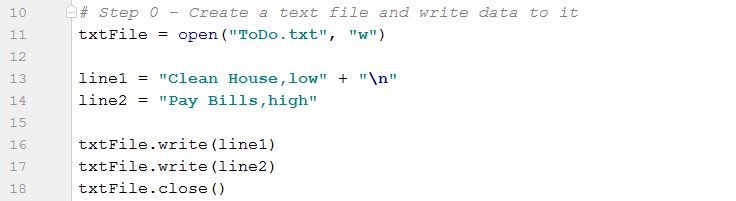
Using PyCharm, I created a new project: *Assignment05* and new Python file: *ToDo.py*. I then copied and pasted in the assignment instructions and hints from the *Assignment 05 starter.py* file, provided by Professor Root.

# Creating the Text File

The first task in Assignment 05 is to create a text file and write two specified lines of data to it. I could have created the text file either with a text editor, such as Notepad or Notepad++, but I decided to write a script in Python to accomplish this task.

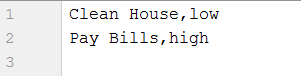
As shown in Figure 1, I opened a new file called *ToDo.txt*, in write mode and then defined the variables *line1* and *line2,* respectively,as the two strings of data provided for Assignment 05. I used the *write()* method to write teach line to my text file, and then closed the file.

Figure 1: Create a Text File with Data



Opening the newly created text file in PyCharm, I can see that the data have been written to the file. (Figure 2)

Figure 2: Data in Todo.txt File

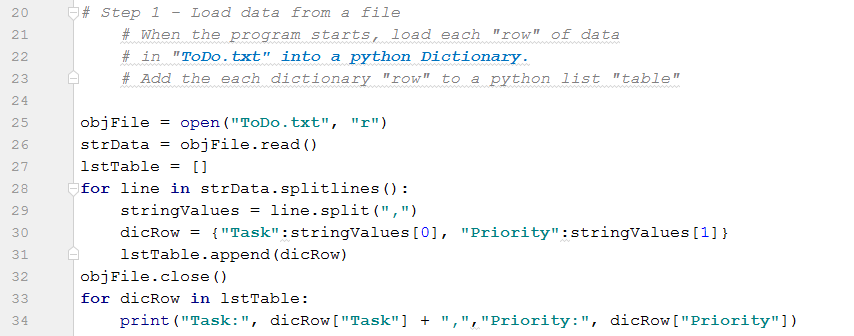


# Load Data into Python Dictionary

When the program starts, each row of data is loaded into a Python dictionary. As shown in Figure 3, the script opens the ToDo.txt file written in the previous step in *read* mode and puts the data into a list object called *lstTable*.

The *splitlines()* method is used to split each line at the comma and creates a new dictionary row for each line where the first list item is assigned to the key *‘Task:’* and the second list item is assigned to the key *‘Priority:’*. These dictionary rows are appended to the *lstTable* object to create a two-key dictionary.

Figure 3: ToDo.txt File Data Loaded into Python Dictionary



The statement in lines 33 and 34 prints each row in the list/table in key-value pairs. Figure 4 shows the resulting ToDo dictionary.

Figure 4: ToDo Dictionary



# Display a Menu of Choices to the User

The program then displays a menu of choices to the user. (Figure 5 and Figure 6) User input is assigned to the *strChoice* variable. I wrote the rest of the program step by step, beginning with Option 1.

Figure 5: List of Options for User Input – Script

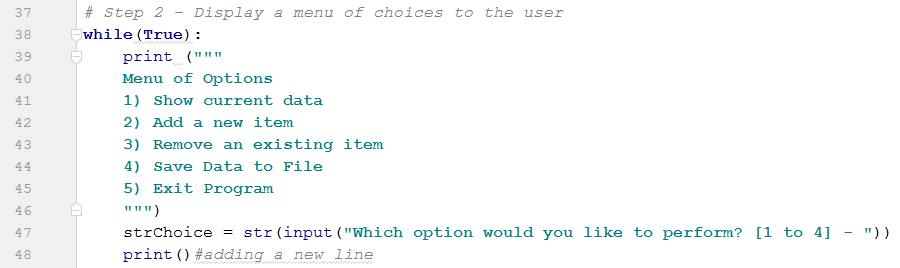
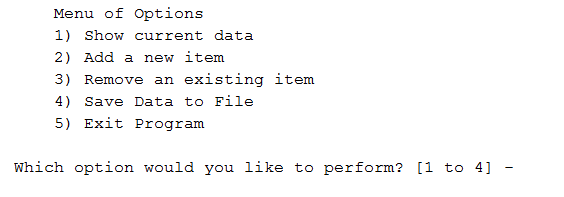


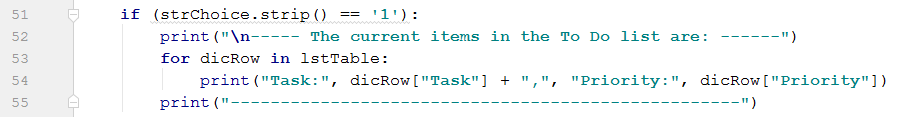
Figure 6: List of Options User Interface



# Show Current Data

If the user enters ‘1’, the program displays a list of current data to the user. As subsequent sections will show, the program operates as a big *if-elif* statement. The *if* block is shown in Figure 7. The commands for the remaining options in the user menu are nested as *elif* statements within the *if* block. The first statement (line 51) tests to see whether it is true that the user entered *‘1’*. If *True*, the program prints each row of that is currently in *lstTable*.

Figure 7: Show Current Data



Since the table has not been affected since it was written, the current list of items will appear at this stage as it does in Figure 4.

# Add a New Item

If the user enters ‘2’, the program asks the user to define a new task for the ToDo list and to assign the task a priority, *(low, medium, or high)*. Figure 8 shows the script which evaluates the first *elif* statement within the nested *if-elif* block. The new values are assigned to the variables: *newTask* and *newPriority*, respectively. On line 61, a new dictionary row is created from the user input, and then appended to the current *lstTable*. After each script block, I’ve chosen to display the current items in the ToDo list.

Figure 8: Adding a New Item to the List/Table

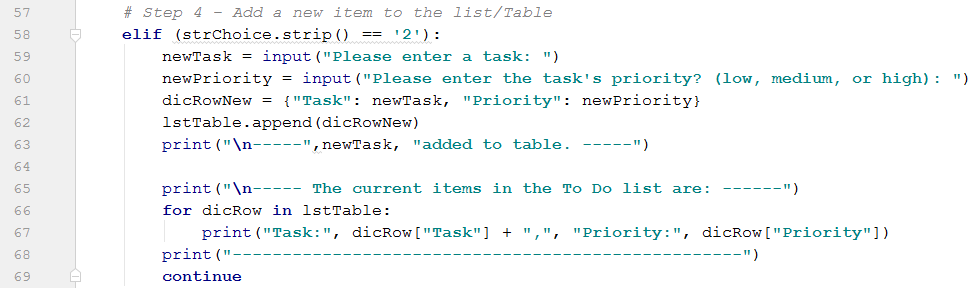
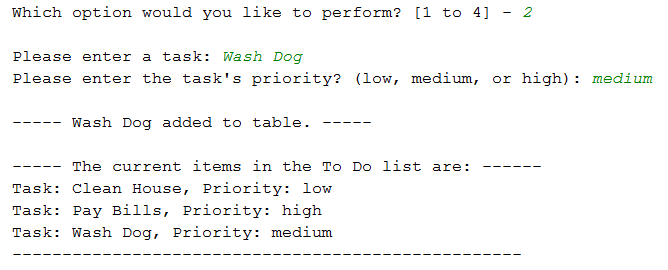


Figure 9 shows an example of the task addition script at work. Here we have added the new Task: *Wash Dog* to the ToDo list and assigned a Priority of *medium*.

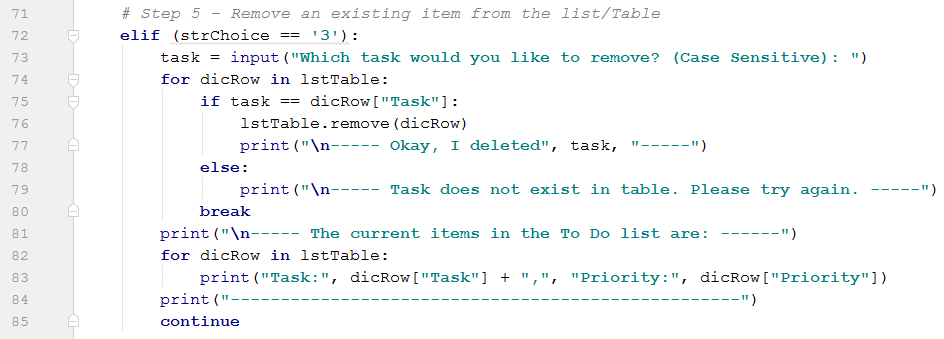
Figure 9: Adding 'Wash Dog' to the ToDo list



# Remove an Existing Item

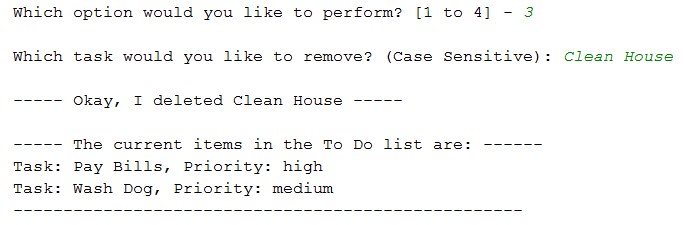
The third menu option gives the user the opportunity to delete an item from the current ToDo list. (Figure 10) This code block uses a *for* loop to go through the current *lstTable* and evaluate whether the *task* exists in the table. If the *task* does exist in the current *lstTable*, the *remove()* method is used to remove the dictionary row that includes the specified task. Specifically, this command removes both key-value pairs (Task and Priority) from the dictionary row. Figure 11 shows an example where the user has chosen to remove ‘Clean House’ from the ToDo list. Note that the entry is case sensitive! In this case, if the user entered ‘clean house’, they would receive the message, “Task does not exist in table. Please try again.”

Figure 10: Removing an Item from the ToDo List



As shown in the figure below, when ‘Clean House’ is removed from the current ToDo list, the entire dictionary row is removed.

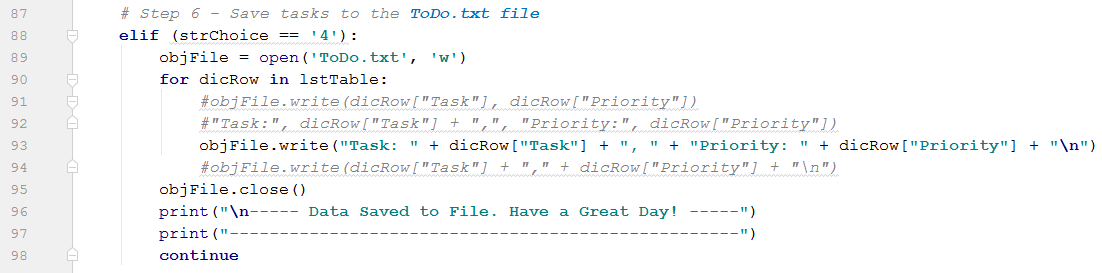
Figure 11: Removing 'Clean House' from the ToDo List



# Save Data to File

Option 4 allows the user to save the data to the ToDo.txt file. As shown in Figure 12, if the user opts to save the data to the file, the program opens the ToDo.txt file in write mode. The *write()* method in the *for* loop writes each dictionary row in the *lstTable* object to the text file on a new line. The file is then closed and a message is displayed to the user.

Figure 12: Save Data to File – Script



Building on the example used throughout this document, when the script shown in Figure 12 is executed, the result is the text file shown in Figure 13.

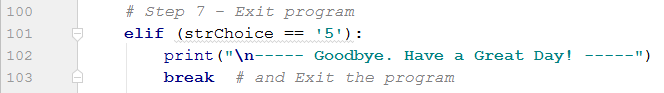
Figure 13: Data Written to Text File

## 

# Exit Program

Finally, the last option in the menu is exiting the program. When this option is selected, the program breaks and is therefore complete.

Figure 14: Exit the Program



## Summary

This document has described the steps taken to create an interactive To Do list, which allows the user to select various options to edit and display the To Do list. This program builds on techniques learned thus far in this course and involves creating, editing and writing to a Python dictionary

# Bibliography

Dawson, M. (2010). Python Programming for the Absolute Beginner, Third Edition. In M. Dawson, *Python Programming for the Absolute Beginner, Third Edition* (p. 3). Boston: Course Technology, a part of Cengage Learning.

Root, R. (2019, May). *\_Mod5PythonProgrammingNotes.docx.* Retrieved from UW Canvas: IT FDN 100 AModules.

Root, R. (2019, May).  *Assignment05\_Starter.py.* Retrieved from UW Canvas: IT FDN 100 AModules.