Richard Sauer

May 12, 2023

Foundations of Programming: Python

Assignment 05

GitHub: richsau/IntroToProg-Python: This repository will be used for reviewing homework files. (github.com)

Lists and Dictionaries

Introduction

The goal of this assignment was to modify an existing program and add functionality that loads data from a file called "ToDoList.txt" in the form of Tasks and Priorities. As data is read in, it's converted to a dictionary and added to a table in memory. A menu is then presented to the user allowing them to:

1) Show the current data
2) Add a new item
3) Removed and existing item
4) Save data back to the file
5) Exit the program

Code then needs to be added for each of the menu items for the program to complete the tasks.

Declaring Variables

The application uses several variables and types. These consist of strings used for user input, strings that are used in multiple locations in the code, a dictionary to hold rows of data, a table to hold rows of dictionary data, and a couple of Booleans to help the flow of the program. (Figure 1)

```python
12   # -- Data -- #
13   # declare variables and constants
14   objFile = None  # An object that represents a file
15   strFile = "ToDoList.txt"
16   strData = ""  # A row of text data from the file
17   dicRow = {}    # A row of data separated into elements of a dictionary {Task,Priority}
18   lstTable = []  # A list that acts as a 'table' of rows
19   strChoice = ""  # A Capture the user option selection
20   strTaskTxt = "Task"
21   strPriorityTxt = "Priority"
22   seperatorText = "----------------"
23   strTaskItem = ""
24   strTaskPriority = ""
25   bolDirtyFlag = False
26   bolFoundItem = False
```

Figure 1. Declaring variables at the top of the application code.

Processing- Reading in Existing Data File

For the processing step, the application reads in the existing data file. As each row of data is read in, it's converted to a dictionary and then appended to a table. (Figure 2)

```
29   # -- Processing -- #
30   # Step 1 - When the program starts, load data from a text file
31   # called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)
32   objFile = open(strFile, "r")
33   for row in objFile:
34       lstRow = row.split(",")  # Split() returns a list
35       dicRow = {strTaskTxt:lstRow[0], strPriorityTxt:lstRow[1].strip()}  # Convert list to dictionary
36       lstTable.append(dicRow)  # Adding dictionary to a table
37   objFile.close()
```

Figure 2. Reading data from a file and converting it to a dictionary before appending it to a table.

Displaying a Menu to the User

A menu of choices is displayed to the user. It's contained in a **while(True)** loop so that it's repeated until the user chooses to exit. (Figure 3)

```
39   # -- Input/Output -- #
40   # Step 2 - Display a menu of choices to the user
41   while (True):
42       print("""
43       Menu of Options
44       1) Show current data
45       2) Add a new item
46       3) Remove an existing item
47       4) Save Data to File
48       5) Exit Program
49       """)
50       strChoice = str(input("Which option would you like to perform? [1 to 5] - "))
51       print()  # adding a new line for looks
```

Figure 3. Displaying a menu to the user.

Displaying the Data in the Table

Menu item #1 is to show the current data. This code first prints out a header using the already defined variables **strTaskTxt** and **strPriorityTxt**, along with a dotted line contained in variable **seperatorTxt**. It then uses a **for** loop to display each row in the table. It uses the same formatting as the header to make all of it line up. (Figure 4)

```
52        # Step 3 - Show the current items in the table
53        # Print a header followed by each task and priority
54        if (strChoice.strip() == '1'):
55            print("\n{:25} {:20}".format(strTaskTxt, strPriorityTxt))
56            print("{:25} {:20}".format(seperatorText, seperatorText))
57            for row in lstTable:
58                print("{:25} {:20}".format(row[strTaskTxt], row[strPriorityTxt]))
59            continue
```

Figure 4. Displaying the data contained in the task table.

Adding New Data

Menu item #2 is adding new data in the form of a task and priority. This code asks the user to enter a new task with is stored in the ***strTaskItem*** variable. It follows by asking the user to enter a priority for that task. This is stored in the ***strTaskPriority*** variable. These are then added to a dictionary, ***dicRow***. The row is then appended to the table. A Boolean dirty flag is also set to True since there is now unsaved data in the dictionary. (Figure 5)

```
60        # Step 4 - Add a new item to the list/Table
61        # Get task and priority from the user
62        # then create a dictionary from the input
63        # and then append it to the table.
64        elif (strChoice.strip() == '2'):
65            strTaskItem = input("Please add a new {0}: ".format(strTaskTxt.lower()))
66            strTaskPriority = input("Please set a {0} for \"{1}\": ".format(strPriorityTxt.lower(), strTaskItem))
67            dicRow = {strTaskTxt:strTaskItem, strPriorityTxt:strTaskPriority}
68            lstTable.append(dicRow)  # adding dictionary to a table
69            bolDirtyFlag = True
70            continue
```

Figure 5. Getting new data from the user and adding it first to a dictionary and then appending it to a table.

Removing Data

Menu item #3 is removing an existing task from the task table. This code first asks the user what task they would like to remove. It then uses a for loop to check that value against the task names in each row of the table. Both the user entered item and the task in the table are compared using the lower case version of the strings using the ***.lower()*** method. If it finds a matching item, that row is removed from the table using the ***.remove()*** method. The code then lets the user know that the item was removed. A Boolean flag, ***bolFoundItem***, indicating that the item was found is set to ***True*** as well as the dirty flag since the data was modified. After the loop, a check is made using the found item Boolean to see if it needs to let the user know that the task was not found. (Figure 6)

```
71        # Step 5 - Remove a new item from the list/Table
72        # Get a task to remove from the user
73        # then search for the task in the table.
74        # If the task is found, remove it from the table else
75        # inform the user that the task was not found.
76        elif (strChoice.strip() == '3'):
77            strTaskItem = input("Enter {0} to be removed: ".format(strTaskTxt.lower()))
78            bolFoundItem = False
79            for row in lstTable:
80                if (row[strTaskTxt].lower() == strTaskItem.lower()):
81                    lstTable.remove(row)
82                    print("{0} \"{1}\" has been removed.".format(strTaskTxt, strTaskItem))
83                    bolDirtyFlag = True
84                    bolFoundItem = True
85            if not (bolFoundItem):
86                print("Sorry, {0} \"{1}\" was not found.".format(strTaskTxt.lower(), strTaskItem))
87            continue
```

Figure 6. Removing data indicated by the user from the table, if found.

Saving Data Back to the File

Menu item #4 is saving the existing table back to the file that was read in at the start of the application. This code opens the file using the write mode and then goes into a ***for*** loop to write each row to the file before closing the file using the ***.close()*** method. The dirty flag is set to ***False*** and then a message is printed to let the user know that the data was successfully saved, along with the name of the file. (Figure 7)

```
88        # Step 6 - Save tasks to the ToDoToDoList.txt file
89        # For each row in the table, write the task and priority back to the file.
90        elif (strChoice.strip() == '4'):
91            objFile = open(strFile, "w")
92            for row in lstTable:
93                objFile.write(str(row[strTaskTxt]) + ',' + str(row[strPriorityTxt]) + '\n')
94            objFile.close()
95            bolDirtyFlag = False
96            print("Data saved in file \"{0}\".".format(strFile))
97            continue
```

Figure 7. Writing the table back to the file and setting the dirty flag to False.

Exiting the Program

Menu item #5 is exiting the program. The code first checks to see if the dirty flag, ***bolDirtyFlag***, is set to ***True***. If it is, then a message is printed for the user letting them know they have unsaved data and asking if they are sure they want to exit. If they respond with a "y" or the dirty flag was False, the program

breaks out of the menu loop and exits. If the user responds with anything other than "y", the program aborts the exit using a ***continue*** statement so that the menu loop can be presented again. (Figure 8)

```
98      # Step 7 - Exit program
99      # Make sure there was no unsaved data
100     # then follow the user's instructions.
101     elif (strChoice.strip() == '5'):
102         if (bolDirtyFlag):
103             strChoice = input("You have unsaved data. Are you sure you want to exit? (y/n) ")
104             if strChoice.strip().lower() == "y":
105                 break  # exit the program
106             else:
107                 continue  # abort exit and return to menu
108         break  # and Exit the program
```

Figure 8. Checking the dirty flag to see if the user needs to give permission before exiting the program.

Testing the Results

In testing the application, it worked as expected using sample input data. I first ran the application from within the PyCharm IDE using each menu item. (Figures 9 - 12)
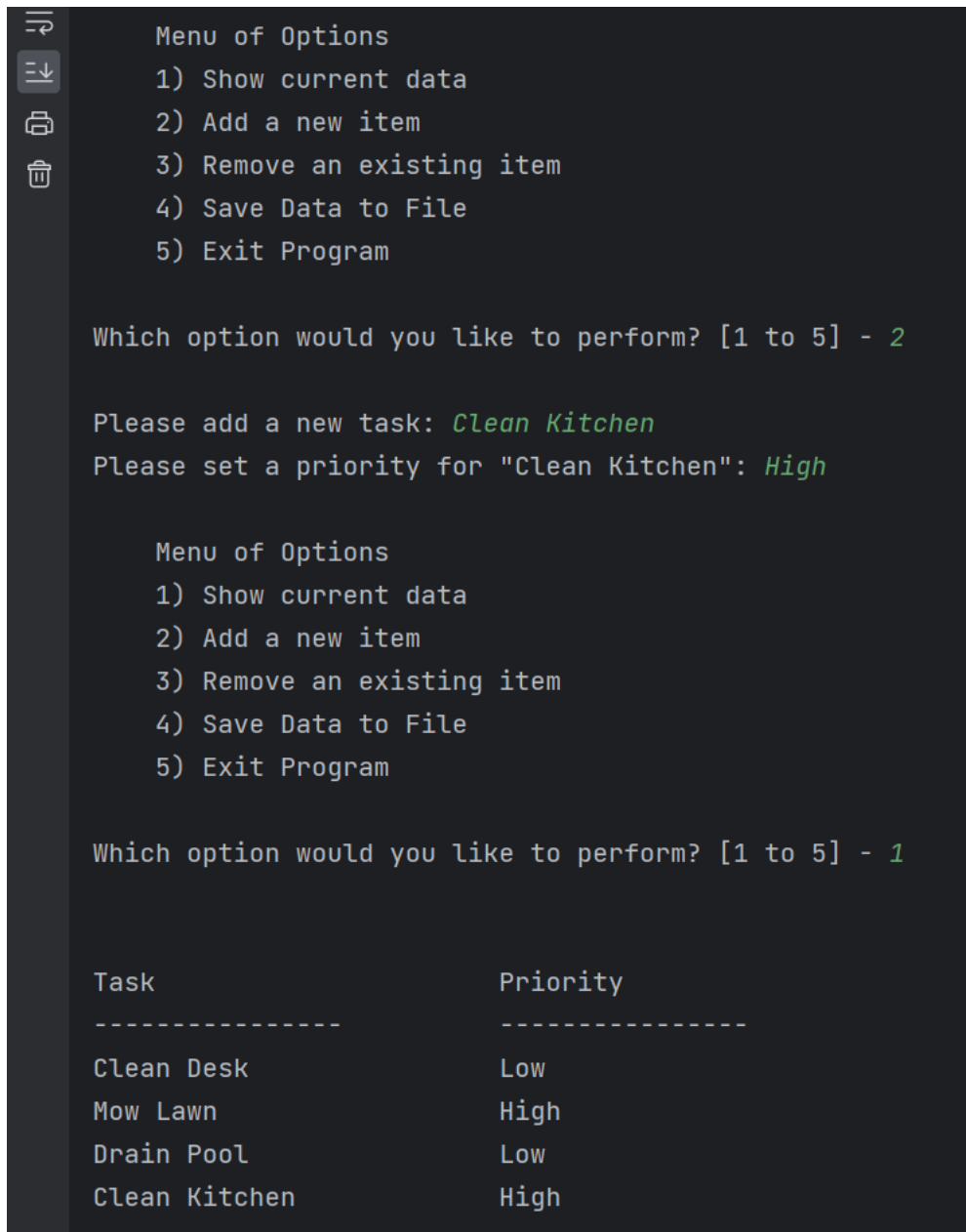
```
"C:\_PythonClass\Module05 - Lists and Dictionaries\Assign

    Menu of Options
    1) Show current data
    2) Add a new item
    3) Remove an existing item
    4) Save Data to File
    5) Exit Program


Which option would you like to perform? [1 to 5] - 1



Task                    Priority
----------------        ----------------
Clean Desk              Low
Mow Lawn                High
Drain Pool              Low
```

Figure 9. Testing menu item #1 – Show current data.

```
    Menu of Options
    1) Show current data
    2) Add a new item
    3) Remove an existing item
    4) Save Data to File
    5) Exit Program

Which option would you like to perform? [1 to 5] - 2


Please add a new task: Clean Kitchen
Please set a priority for "Clean Kitchen": High

    Menu of Options
    1) Show current data
    2) Add a new item
    3) Remove an existing item
    4) Save Data to File
    5) Exit Program

Which option would you like to perform? [1 to 5] - 1


Task                    Priority
---------------         ---------------
Clean Desk              Low
Mow Lawn                High
Drain Pool              Low
Clean Kitchen           High
```

Figure 10. Menu option #2 – Add a new item and then show current data.

```
Which option would you like to perform? [1 to 5] - 3


Enter task to be removed: Clean Bathroom
Sorry, task "Clean Bathroom" was not found.


    Menu of Options
    1) Show current data
    2) Add a new item
    3) Remove an existing item
    4) Save Data to File
    5) Exit Program


Which option would you like to perform? [1 to 5] - 3


Enter task to be removed: Drain Pool
Task "Drain Pool" has been removed.


    Menu of Options
    1) Show current data
    2) Add a new item
    3) Remove an existing item
    4) Save Data to File
    5) Exit Program


Which option would you like to perform? [1 to 5] - 1



Task                        Priority
---------------             ---------------
Clean Desk                  Low
Mow Lawn                    High
Clean Kitchen               High
```

Figure 11. Menu item #3 - Trying to remove an item that doesn't exist and then successfully removing one that does.

```
Which option would you like to perform? [1 to 5] - 5


You have unsaved data. Are you sure you want to exit? (y/n) n


    Menu of Options
    1) Show current data
    2) Add a new item
    3) Remove an existing item
    4) Save Data to File
    5) Exit Program


Which option would you like to perform? [1 to 5] - 4


Data saved in file "ToDoList.txt".


    Menu of Options
    1) Show current data
    2) Add a new item
    3) Remove an existing item
    4) Save Data to File
    5) Exit Program


Which option would you like to perform? [1 to 5] - 5



Process finished with exit code 0
```

Figure 12. Menu items # 4 & 5 - Testing out the dirty flag, saving the data, and exiting the program.

I verified that the program can be run from a command line. (Figure 13)

```
Command Prompt          ×     +    ∨

C:\_PythonClass\Module05 - Lists and Dictionaries\Assignment05>python ToDoList.py

    Menu of Options
    1) Show current data
    2) Add a new item
    3) Remove an existing item
    4) Save Data to File
    5) Exit Program

Which option would you like to perform? [1 to 5] - 1


Task                      Priority
------------------        ------------------
Clean Desk                Low
Mow Lawn                  High
Clean Kitchen             High

    Menu of Options
    1) Show current data
    2) Add a new item
    3) Remove an existing item
    4) Save Data to File
    5) Exit Program

Which option would you like to perform? [1 to 5] - 5


C:\_PythonClass\Module05 - Lists and Dictionaries\Assignment05>
```

Figure 13. Running the application from the command line.

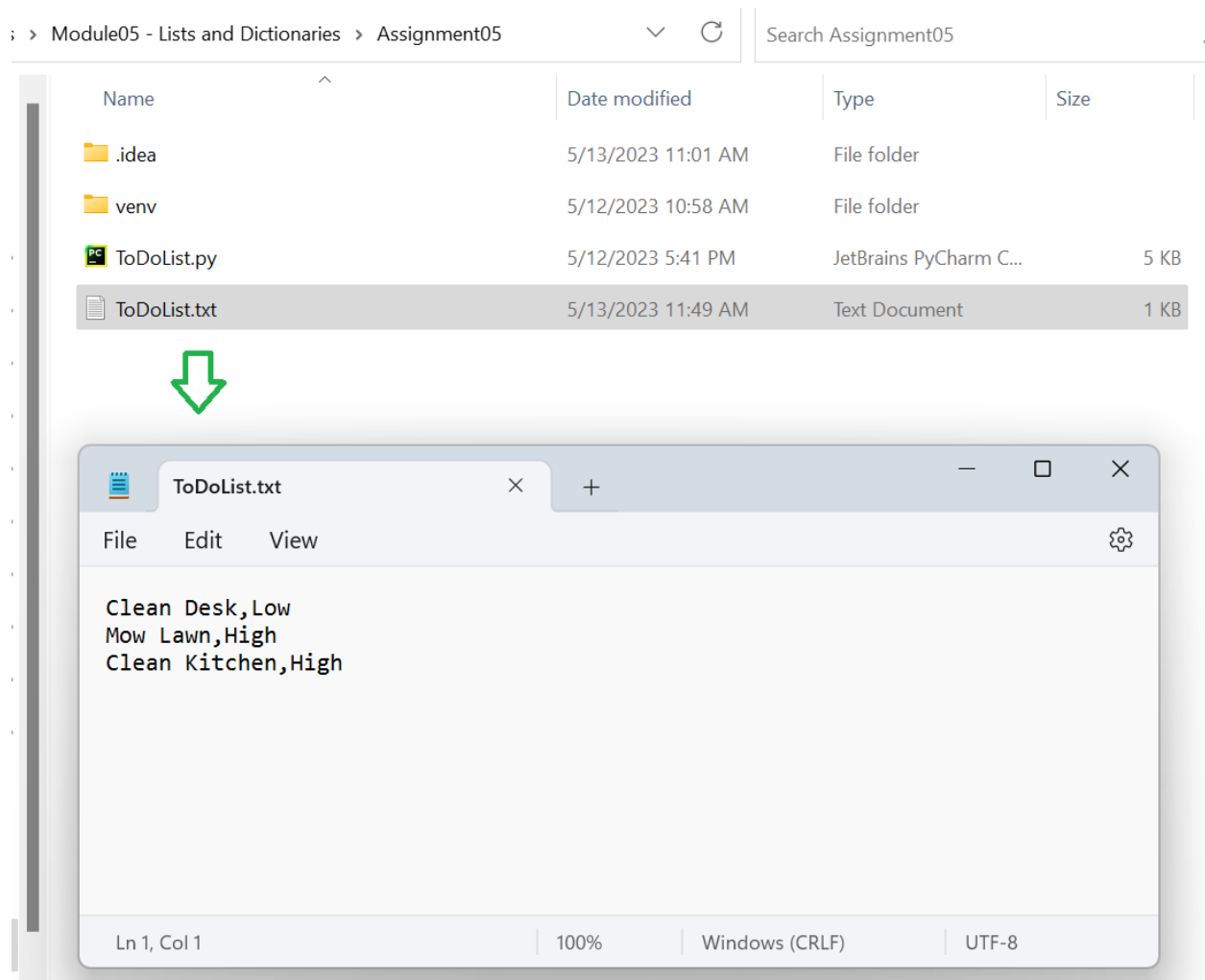I then verified the data was saved correctly by looking at the output file. (Figure 14)

Figure 14: Verifying results in the output file.

Summary

The assignment was to start with existing code that provided the framework of an application that displayed a menu for editing a task list. My job was to read in an existing data file and then implement the features described in the menu of the application. These features were 1) Display the task data, 2) Add a new task, 3) Remove a task, 4) Save the task data back to the data file, and 5) Exit the application.

I then tested the application in both PyCharm and the command line and verified the contents of the data file.