Michelle Diaz

February 14, 2023

IT FDN 110 A Wi 23: Foundations of Programming: Python

Assignment05

Assignment 05: Lists and Dictionaries

Introduction

This document outlines the process used to create a script that provides the user several options for managing a To-Do list (Figure 1). The menu options are:

```
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
```

Figure 1: List of options user has to manage the To-Do list

The menu is provided after every operation except when the user chooses to exit the program.

Creating the Program

Initializing the Program

There are a couple of steps to starting the program. A declaration of variables is included to organize some of the variables being used in the script (Figure 2).

```
# -- Data -- #
# declare variables and constants
strFile = "ToDoList.txt"  # A data file
strData = ""  # A row of text data from the file
dicRow = {}  # A row of data separated into elements of a dictionary
{Task,Priority}
lstTable = []  # A list that acts as a 'table' of rows
lstRow = []  # A list that stores each dictionary entry for printing
strMenu = ""  # A menu of user options
strChoice = ""  # A Capture the user option selection
```

Figure 2: Declaration of variables in the script

Before the user is offered any options, the script checks if the text file already exists, and if it does then the data is loaded into the lstTable as dictionary objects (Figure 3). If the file does not exist, then the program prints a blank line and then proceeds to the next step. I found that I couldn't use a break or continue statement because the 'try' operation is not a loop. Printing a blank line is how I handled including the expected 'except' operation.

```
# -- Processing -- #
# Step 1 – When the program starts, load any data you have
# in a text file called ToDoList.txt into a python list of dictionaries rows
(like Lab 5-2)
try:
    objFile = open(strFile, 'r')
    for row in objFile:
        strData = row.split(',')
        dicRow = {"Task": strData[0], "Priority": strData[1].strip()}
        lstTable.append(dicRow)
    objFile.close()
except:
    print()
```

Figure 3: Reading existing data into the lstTable variable as dictionary objects

Once that is done, the menu is then presented to the user and the choice is populated into a temporary strChoice variable that will be used to navigate through the various options of the while loop (Figure 4).

```
# -- Input/Output -- #
# Step 2 - Display a menu of choices to the user
while True:
    print("""
    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
    """)
    strChoice = str(input("Which option would you like to perform? [1 to 5] -
"))
    print()  # adding a new line for looks
```

Figure 4: The user option is stored in a string for use in the subsequent while loop

Showing Current Data

To show the current data, I set the program up to check if there is any length to the lstTable object. If not, the user is informed that there are no tasks in the To-Do list and to choose another option. Otherwise, the existing data that was read into the lstTable object during initialization and any added data from the user is displayed row by row (Figure 5).

```
# Step 3 – Show the current items in the table
if strChoice.strip() == '1':
    if not len(lstTable):
        print("There are no tasks to list. Please choose another option.")
        input("Press Enter to continue…")
    else:
        for row in lstTable:
            print(row)
        input("Press Enter to continue…")
        continue
```

Figure 5: The program checks if the list is empty and displays any existing data

Adding A New Item

New items are added by collecting two strings from the user for the task and priority and then storing the two strings in a dictionary with the keys "Task" and "Priority" (Figure 6). Once this is done, the dictionary is appended to the end of the lstTable list object. When the strings are stored in the dictionary object, any white space is stripped from the end of the line.

```
# Step 4 – Add a new item to the list/Table
elif strChoice.strip() == '2':
    strTask = input("Enter a task: ")
    strPriority = input("Enter the task priority: ")
    dicRow = {"Task": strTask, "Priority": strPriority.strip()}
    lstTable.append(dicRow)
    input("Press Enter to continue…")
    continue
```

Figure 6: New data is collected from the user and stored in a dictionary which is added to lstTable

Removing An Existing Item

This particular action is one of the tasks I had the most challenging time with. The way I have constructed it gets the job done but could be made more robust if I had more time. If the user input doesn't match any of the dictionary keys for "Task" then the program will just loop through the menu again. If I had more time, I would make sure any lower- or upper-case combinations are matched with the corresponding dictionary row in the list. I would also ensure if there were typos or non-existent tasks input by the user that there would be a notification that this is the case.

In this scenario, the task to be deleted is input by the user and the lstTable list object is traversed until the key matches the input. Once a match is found, the row is deleted from the table lstTable and the user is notified that the task has been deleted and the 'for' loop is broken and the program continues (Figure 7). If the task is not found in the list, the program finishes searching the lstTable and then continues with prompting the user for another action.

```
# Step 5 – Remove a new item from the list/Table
elif strChoice.strip() == '3':
    task = input("What task would you like to delete?: ")
    for row in range(len(lstTable)):
        if lstTable[row]['Task'] == task:
            del lstTable[row]
            print("\nThe " + task + " task has been deleted.")
            input("Press Enter to continue…")
            break
    continue
```

Figure 7: Task is searched for in the list and then deleted once found

Saving Data to A File

Once the user selects to save the data in the list to a text file, the rows in the lstTable object are populated into individual lines representing the tasks the user has populated. The line is written to the text file and a new line is inserted at the end of every row (Figure 8). Here I am able to reference the data using the keys I have assigned within the dictionary objects ("Task" and "Priority").

```
# Step 6 - Save tasks to the ToDoList.txt file
elif strChoice.strip() == '4':
    txtFile = open("ToDoList.txt", "w")
    for row in lstTable:
        txtFile.write(str(row["Task"]) + ',' + str(row["Priority"]) + '\n')
    txtFile.close()
    print("Data was saved to ToDoList.txt")
    continue
```

Figure 8: Data is stored to the 'ToDoList.txt' file

Exiting the Program

Exiting the program is done via a simple break statement (Figure 9).

```
# Step 7 - Exit program
elif strChoice.strip() == '5':
    print("Exiting program...")
    break  # and Exit the program
```

Figure 9: The program notifies the user and terminates

Testing the Program

To test the program, I progressed through each option and made sure they worked right before moving onto the next one. Some of the errors I got while testing were when I was trying to reference dictionary entries from within the list, and it took me a few tries to understand how to do this. Once I was able to get all the elements in the program functional, I tested them all together and also checked the text file during different steps in the program. Figures 10 thru 19 illustrate the flow of the program when tested in PyCharm.

```
C:\Users\mdiaz\venv\Scripts\python.exe C:\_PythonClass\Assignment05\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


There are no tasks to list. Please choose another option.
Press Enter to continue...
```
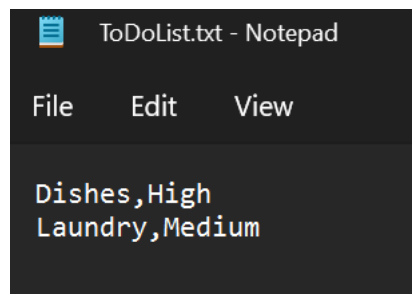
Figure 10: The program working in PyCharm

```
    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

Enter a task: Dishes
Enter the task priority: High
Press Enter to continue...
```

Figure 11: The program working in PyCharm

```
    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

Enter a task: Laundry
Enter the task priority: Medium
Press Enter to continue...
```

Figure 12: The program working in PyCharm

```
     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 was saved to ToDoList.txt
```

Figure 13: The program working in PyCharm

```
ToDoList.txt - Notepad

File     Edit     View

Dishes,High
Laundry,Medium
```

Figure 14: The text file after the first save

```
     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


What task would you like to delete?: Dishes


The Dishes task has been deleted.
Press Enter to continue...
```

Figure 15: The program working in PyCharm

```
    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 was saved to ToDoList.txt
```

Figure 16: The program working in PyCharm

```
📄  ToDoList.txt - Notepad

File     Edit     View

Laundry,Medium
```
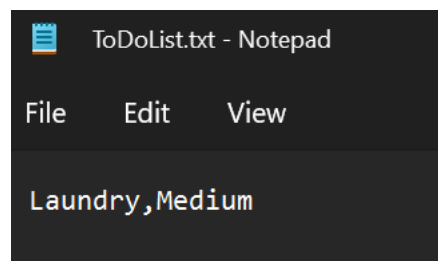
Figure 17: The text file after the second save

```
    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': 'Laundry', 'Priority': 'Medium'}
Press Enter to continue...
```

Figure 18: The program working in PyCharm

Figure 19: The program working in PyCharm

The test was done similarly in command prompt and the outcome was the same. Figure 20 thru 23 demonstrate the code working in command prompt.

Figure 20: The program working in Command Prompt

```
    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

Enter a task: Laundry
Enter the task priority: Medium
Press Enter to continue...

    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': 'Dishes', 'Priority': 'High'}
{'Task': 'Laundry', 'Priority': 'Medium'}
Press Enter to continue...
```

Figure 21: The program working in Command Prompt

```
    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 was saved to 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] – 3

What task would you like to delete?: Dishes

The Dishes task has been deleted.
Press Enter to continue...
```

Figure 22: The program working in Command Prompt

Figure 23: The program working in Command Prompt

Summary

This assignment was challenging since it really required a solid understanding of how to access dictionaries and how to manipulate dictionary objects within lists. I can see the value of using dictionaries and will be studying more on them to gain a more thorough understanding of them in preparation for the next assignments.