Meg Giddings
May 16, 2020
Foundations of Programming (Python)
Assignment05

To Do List: Dictionaries and Lists in Python

Introduction

This paper gives an overview of a program designed to manage elements of a "to do" list, including tasks and their corresponding level of priority. The starter code was provided as part of the assignment, and I supplemented the TODO sections with my own code. This program allows the user to display current tasks in the list, add tasks, remove tasks, and save the list of tasks to a file, all utilizing the dictionary data type to store the elements of the list.

Declaring Variables

The first section in my script declares variables (Figure 1). As this section was provided as part of the starter code, it highlights some of the challenges of working with someone else's code, because I didn't find the need to use all of the declared variables in the script, and even created others that more suited my needs.

```
# -- Data -- #
# Declare variables and constants
objFile = "ToDoList.txt"   # An object that represents a 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
strMenu = ""     # A menu of user options
strChoice = ""   # A Capture the user option selection
```

Figure 1: Declared variables for the ToDoList program

Loading Data

The first step in this script involves loading any existing data from the ToDoList.txt file into the program's memory (Figure 2).

```
# -- Processing -- #
# Step 1 - When the program starts, load any data you have
# in a text file called ToDoList.txt into a Python list of dictionary rows
loadFile = open(objFile, "r")
for row in loadFile:
    lstRow = row.split(",")
    dicRow = {"task": lstRow[0], "priority": lstRow[1].strip()}
    lstTable.append(dicRow)
loadFile.close()
```

Figure 2: Loading data from ToDoList.txt into the program

While the user cannot “see” this code being performed, it sets the groundwork for the rest of the script and all the actions to be performed upon the loaded data.

Creating a Menu

After the data from ToDoList.txt is loaded into the program, a menu is displayed to the user, laying out the options that can be performed (Figure 4). As part of a *while* loop, this menu repeats and displays to the user each time an action is completed. This section also captures the user’s choice as a string value and assigns it to a variable (Figure 3).

```
# -- Input/Output -- #
# Step 2 - Display a menu of choices to the user
while (True):
    print("""
    Menu of Options
    1) Show current tasks
    2) Add a new task
    3) Remove an existing task
    4) Save tasks 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 3: Displaying a menu of choices and capturing input for the user’s choice

```
    Menu of Options
    1) Show current tasks
    2) Add a new task
    3) Remove an existing task
    4) Save tasks to file
    5) Exit program

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

Figure 4: Menu displayed to user while running in PyCharm

Displaying Current Data

When the user selects menu option ‘1’, the program displays its current data (Figure 6). While this data could be displayed in its raw list form, I added some formatting in the code to make it appear more streamlined to the user (Figure 5).

```
    # Step 3 - Show the current items in the table
    if (strChoice.strip() == '1'):
        print("Task" + ' | ' + "Priority")
        for row in lstTable:
            print(row["task"] + ": " + row["priority"])
        continue
```

Figure 5: Utilizing a for loop to display current program data

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

Task | Priority
Vacuum: Low
Wash dishes: High
Fold laundry: Low
```

Figure 6: Option 1 (displaying current tasks) running in PyCharm

Adding Data

When the user selects menu option '2', the program prompts the user for some input – a new task and its corresponding level of priority (Figure 8). This new input is assigned to a dictionary element and appended to the existing data. I included a print statement to let the user know that the data has been added (Figure 7).

```
    # Step 4 - Add a new item to the list/Table
    elif (strChoice.strip() == '2'):
        print('Enter a task and the level of priority.')
        strTask = input('Task: ')
        strPriority = input('Priority: ')
        dicRow = {"task": strTask, "priority": strPriority}
        lstTable.append(dicRow)
        print("Task added!")
        continue
```

Figure 7: Capturing user input and appending data to the list

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

Enter a task and the level of priority.
Task: Make bed
Priority: High
Task added!
```

Figure 8: Option 2 (adding a task to the list) running in PyCharm

Removing Data

When the user selects menu option '3', the program again prompts the user for some input, in this case which task to remove from the list. This input is assigned to a variable and assessed through *if* statements (Figure 9). If the input the user enters matches a key in the list of dictionary items, the task is removed from the list, and the user is notified (Figure 10). If the input does not match, the user is notified that the task was not found (Figure 11).

```
    # Step 5 - Remove a new item from the list/Table
    elif (strChoice.strip() == '3'):
        strRemove = input("Task to remove: ")
        for row in lstTable:
            if row["task"].lower() == strRemove.lower():
                lstTable.remove(row)
                print("Task removed!")
                break
        if dicRow["task"].lower() != strRemove.lower():
            print("Task not found!")
        continue
```

Figure 9: Capturing user input and removing data from the list

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

Task to remove: Make bed
Task removed!
```

Figure 10: Option 3 (removing a task from the list) running in PyCharm

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

Task to remove: Clean windows
Task not found!
```

Figure 11: Option 3 running in PyCharm when the task is not in the list

Saving Data to a File

When the user selects menu option '4', whatever data is currently in the program's memory is written to the ToDoList.txt file (Figure 12).

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

Figure 12: Saving current program data to the ToDoList.txt file

Because the user can't really "see" anything happening, I added a print statement to notify the user that the action has been completed (Figure 13).

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

Tasks saved to file!
```

Figure 13: Option 4 (saving tasks to a file) running in PyCharm

Menu option '5' allows the user to exit the program. I chose to add some functionality to this option and allow the user an additional opportunity to save the program data before exiting (Figures 14 & 15).

```
    # Step 7 - Exit program
    elif (strChoice.strip() == '5'):
        strExit = input("Do you want to save your tasks before exiting? Enter 'y' or 'n': ")
        if strExit == "y":
            loadFile = open("ToDoList.txt", 'w')
            for row in lstTable:
                loadFile.write(str(row["task"]) + ',' + str(row["priority"]) + '\n')
            loadFile.close()
            print("\n" + "Tasks saved. Goodbye!")
        elif strExit == "n":
            break
        break  # and Exit the program
```

Figure 14: Prior to exiting the program, the user can save current data or exit without saving

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

Do you want to save your tasks before exiting? Enter 'y' or 'n': y

Tasks saved. Goodbye!
```

Figure 15: Option 5 (exiting the program) running in PyCharm

To verify that the data from the program's memory was saved to the file, I checked the contents of ToDoList.txt (Figure 16).

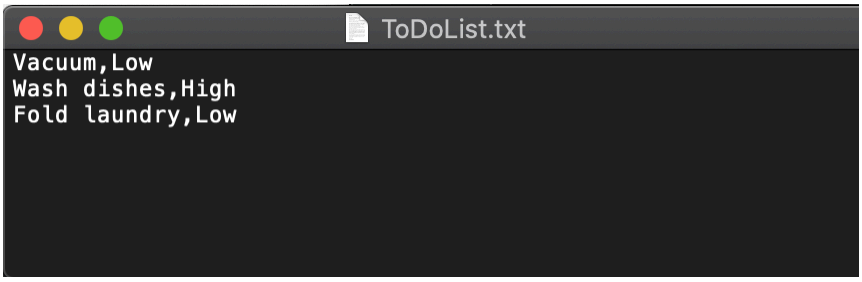

Figure 16: Verifying data saved to ToDoList.txt file

After running the full script numerous times in PyCharm, adding, removing, and displaying current elements of the list, I then ran it in Terminal (Figure 17).

```
    Menu of Options
    1) Show current tasks
    2) Add a new task
    3) Remove an existing task
    4) Save tasks to file
    5) Exit program

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

Task | Priority
Vacuum: Low
Wash dishes: High
Fold laundry: Low

    Menu of Options
    1) Show current tasks
    2) Add a new task
    3) Remove an existing task
    4) Save tasks to file
    5) Exit program

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

Enter a task and the level of priority.
Task: Wash car
Priority: High
Task added!

    Menu of Options
    1) Show current tasks
    2) Add a new task
    3) Remove an existing task
    4) Save tasks to file
    5) Exit program

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

Task to remove: Wash car
Task removed!

    Menu of Options
    1) Show current tasks
    2) Add a new task
    3) Remove an existing task
    4) Save tasks to file
    5) Exit program

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

Tasks saved to file!

    Menu of Options
    1) Show current tasks
    2) Add a new task
    3) Remove an existing task
    4) Save tasks to file
    5) Exit program

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

Do you want to save your tasks before exiting? Enter 'y' or 'n': y

Tasks saved. Goodbye!
```

Figure 17: Running all "options" of ToDoList.py in Terminal

Summary

This program demonstrates the use of the dictionary data type and using dictionaries within lists. It includes script for displaying list data, appending data to a list, removing data from a list, and saving data to a file. I found some aspects of this assignment challenging, often confusing dictionary and list functions and arguments. It was also a new challenge to work with someone else's code. So overall, I was excited that I was able to successfully make all aspects of this program function as intended. I am looking forward to learn more ways to build out a program like this, such as error handling and other techniques.