Sarah Anderson

November 11, 2022

Intro to Programming – Python

https://github.com/sarahanderson94/IntroToProg-Python

Assignment 05 – Creating a To Do List Program

Introduction

This week's assignment continued to focus on using collections of data and building off topics from previous weeks. It demonstrated how to use lists to read in data from files and to write data to files and introduced another data collection type called dictionaries. The program created this week stores data for a to do list. The program reads in an existing file containing data for a to do list and then presents several options to the user to either view the current to do list, add more rows, remove a row, and save the data.

Program Design

The shell of the program is very similar to the Home Inventory program, but this program instead starts with a section for declaring variables and a processing section that reads in data from an existing file. The additions of the data and processing sections is an effort to start getting comfortable following the computer science design principle known as Separation of Concerns (SoC). The goal of SoC is to separate code into distinct sections. Concerns can vary in complexity and specificity. Well separated programs provide better opportunities to easily reuse code and work on sections independently of the program. The rest of the setup is very similar to Home Inventory in that the code consists of several If-Elif statements nested in a while loop to allow for the user to loop through the menu of options available to them.

Declaring Variables

This program begins with a Data section, which is shown in Figure 1. This is the section where many of the variables and objects used in the program are declared. One benefit of declaring the variables here is to follow the principle of Separation of Concerns. This also helps make the code easier to read by having a section for the data. The variables in this program have been declared and initialized with empty values. Later in the program they will receive different values to store data.

```
# -- Data -- #
# declare variables and constants
strFile = "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
strTask = ""  # Capture task value
strPriority = ""  # Capture priority value

```

Figure 1 – Data Section of To Do Program

Processing – Load Existing Data

The next section of the program is the processing section, shown in Figure 2. Here data from an existing text file is loaded into the program. An object, “objFile” is created in this section to read in the text file that is stored in the object, “strFile”. The “r” tells the function open() to read a file. A For loop is used to read each row from the file into a list. The split() method removes the comma between elements in each row of the text file. Each row is then put into a dictionary. Dictionaries are similar to other sequences like lists and tuples, but the information is stored in a pair: a key and its value. The to do list program has two keys, Task and Priority. The Task stores values about a task, such as ‘mow the lawn’ and the Priority key stores the level of importance for the task. The keys are very similar to a column header of a table. Each dictionary is appended into a table called “lstTable”.

```
# -- Processing -- #
# Load any data from a text file called ToDoList.txt
# into a python list of dictionaries rows

objFile = open(strFile, "r")
for row in objFile:
    # removes the comma between elements in the text file
    lstRow = row.split(",")
    # use strip function to remove the \n from printing out
    dicRow = {"Task": lstRow[0].strip(), "Priority": lstRow[1].strip()}
    lstTable.append(dicRow)
objFile.close()
print("Successfully loaded data")
```

Figure 2 Processing – read in file

Creating the Menu Options

This program has five options for the user, code shown in Figure 3:

1. Viewing the current data
2. Adding a new row
3. Removing a row
4. Saving the data
5. Exit the program

This section takes in a value from the user to determine which option to run through. The below sections describe the process to create each menu option's code.

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

Figure 3 Menu of Options

1. View Current Data

If the user chooses option 1, then they expect to see the current data printed out to them. This is easy to accomplish by using the For loop and the print function (Figure 4). The For loop iterates through each dictionary row in the table and prints it out. The output is shown is Figure 5. Printing each dictionary row includes the curly brackets, the key, and value. This code could be improved by changing the formatting that the data is displayed to the user.

```
52      if (strChoice.strip() == '1'):
53          print("Current Data: \n")
54          for row in lstTable:
55              print(row)
56          continue
```

Figure 4: Print Current Data Code

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

Current Data:

{'Task': 'clean bathroom', 'Priority': 'high'}
{'Task': 'grocery shopping', 'Priority': 'low'}
{'Task': 'buy xmas gifts', 'Priority': 'high'}
```

Figure 5: Option 1 User Output

2. Add an Item to the Table

If the user choose option 2, then they expect to add a new row. The two variables in this section store the task value and the priority value. The append method allows a new dictionary row to be added to the table. The strip and lower methods are used to remove any extra spaces off of the user entered data and convert the string to lower case. Figure 6 displays the code used to create this section.

```
58      elif (strChoice.strip() == '2'):
59          strTask = input("Enter a task: ")
60          strPriority = input("Enter a priority (high, medium, low): ")
61          lstTable.append({"Task": strTask.strip().lower(), "Priority": strPriority.strip().lower()})
62          continue
```

Figure 6: Add data to table

3. Remove a Row from the Table

The third menu option allows the user to remove a row of data. The code is shown in Figure 7 The program asks the user what task they want to remove and stores the string in variable “strRemove”. The strip function is used to make sure there aren’t any extra spaces after the user’s data. A For loop is used to compare the string from the user to all the values for the key “Task”. If strRemove matches a value of “Task”, then the method remove() takes that row out of the table and breaks out of the For loop. If a match is not found, a “not found” message displays to the user.

```
64      elif (strChoice.strip() == '3'):
65          print(lstTable)
66          strRemove = input("Enter a task to remove: ")
67          strRemove.strip()
68          for row in lstTable.copy():
69              if row.get("Task") == strRemove.lower():
70                  lstTable.remove(row)
71                  break
72          # Placing the else statement outside the For loop prevents
73          # the program from printing multiple lines of "row not found"
74          else:
75              print("row not found")
76          continue
```

Figure 7 Removing a Row

4. Saving the Data

The last option that required some code is saving the data to a file. If the user enters “4”, a file object is opened. The “w” tells the open function to write to the file. A For loop that iterates through the table writes each dictionary row to the file. Then file object closes. Figure 8 shows this code.

```
79      elif (strChoice.strip() == '4'):
80          # Open a new text file to save the data entered by the user
81          objFile = open(strFile, "w")
82          print("Data saved")
83          for row in lstTable:
84              objFile.write(row["Task"] + ',' + row["Priority"] + '\n')
85          objFile.close()
86          continue
```

Figure 8 Saving to the File

Screenshots of Code Running in PyCharm and Command

Figure 9-11 and Figure 12 show the code running in Pycharm and Command, respectively.

```
"C:\_Python Class\Assignment05\venv\Scripts\python.exe" "C:\_Python Class\Assignment05\Assigment05_Starter.py"
Successfully loaded 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] -
Current Data:

{'Task': 'clean bathroom', 'Priority': 'high'}
{'Task': 'grocery shopping', 'Priority': 'low'}
{'Task': 'buy xmas gifts', 'Priority': 'high'}
```

Figure 9 Starting the code in Pycharm and viewing the current data

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

Enter a task: Walk dog
Enter a priority (high, medium, low): 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] - 3

[{'Task': 'clean bathroom', 'Priority': 'high'}, {'Task': 'grocery shopping', 'Priority': 'low'}, {'Task': 'buy xmas gifts', 'Priority': 'high'}, {'Task':
Enter a task to remove: clean bathroom
row removed
```

Figure 10: Entering a New Task into the Table and Removing a Row in PyCharm

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

Data saved

    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

Closing Program. Good Bye

Process finished with exit code 0
```

Figure 11: Saving the Data and Ending the Program in PyCharm

```
C:\Users\sarah>Python.exe "C:\_Python Class\Assignment05\Assigment05_Starter.py"
Successfully loaded 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] - 1

Current Data:

{'Task': 'grocery shopping', 'Priority': 'low'}
{'Task': 'buy xmas gifts', 'Priority': 'high'}
{'Task': 'walk dog', 'Priority': '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] - 3

[{'Task': 'grocery shopping', 'Priority': 'low'}, {'Task': 'buy xmas gifts', 'Priority': 'high'}, {'Task': 'walk dog', '
Priority': 'high'}]
Enter a task to remove: buy xmas gifts
row 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
```

Figure 12: Program Running in Command Console

Summary

This assignment gave me a better understanding of how to read and write to files and the benefits of using a dictionary versus a list or a tuple. I am looking forward to learning how to create functions in the next assignment.