Name: Gail Provancha

Date: August 7, 2023

Course: IT FDN 110 A Foundations of Programming: Python

Assignment: Assignment05

GitHub: https://github.com/provgl1/IntroToProg-Python

Working with Dictionaries and Files

Introduction

This document will go over the steps needed to write a Python Script that provides a menu for the user to make selections that can add, delete, and save to a "to do list". This script will be utilizing lists, dictionaries, while and for loops. Below I will go into more detail regarding how the script is to be written.

Creating a Header, Declare Variables and Pseudo-Code

To start, you will be adding to a preexisting script and naming it "ToDoList.py". You will want to update the header for the script, which will include a title, description and a change log. The header is to provide information regarding the script for individuals, including yourself, on the purpose of the script and when and who made modifications.

The next section is to declare your variables, even though not required since Python creates variables when values are assigned, it still is useful information for future code updates and readability.

And finally, pseudo-code is used to create an outline of the steps needed to write this script. This helps break the script into sections and explains what the programmer is trying to accomplish in easily understandable terms. (Figure 1).

```
# ------------------------------------------------------------------------ #
# Title: Assignment 05
# Description: Working with Dictionaries and Files
#              When the program starts, load each "row" of data
#              in "ToDoToDoList.txt" into a python Dictionary.
#              Add each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# Gail Provancha, 8.6.2023, Added code to complete assignment 5
# ------------------------------------------------------------------------ #

# -- 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

# -- Processing -- #
# Step 1 - When the program starts, load the data you have
# in a text file called ToDoList.txt into a python list of dictionaries rows.

# -- Input/Output -- #
# Step 2 - Display a menu of choices to the user
# Step 3 - Show the current items in the table
# Step 4 - Add a new item to the list/Table
# Step 5 - Remove a new item from the list/Table
# Step 6 - Save tasks to the ToDoToDoList.txt file
# Step 7 - Exit program
```

Figure 1: Header, Declare variables, and Pseudo-Code

Step 1 – Load the Data into a List

You will use the open() function, which requires the name of the file "objFile" and the mode "r" to read the file. The file will already need to exist for the "r" to work.

The "for loop" will be used to go through all the rows of data, splitting the data into a list based on ",". The rows of data will then be saved to a dictonary "dicRow"and in turn appended to a list "lstTable". Once the last row of information in the collection is read, the "for loop" will automatically end, which makes it different from a "while loop" which requires a condition to be met.

Since the text file "objFile-ToDoList.txt" needs to already exist, error handling was added using the "Try-Except." If an error occurs processing the block of code under "try:", then the process will jump to the "except:" which will print a message to the user (Figure 2).

```
22      # -- Processing -- #
23      # Step 1 - When the program starts, load the data you have
24      # in a text file called ToDoList.txt into a python list of dictionaries rows.
25      try:
26          objFile = open(objFile, "r")  # open text file
27          for row in objFile:  # loop through the rows in the text file
28              row = row.split(",")  # split each row by a comma to return a list
29              dicRow = {"Task": row[0], "Priority": row[1].strip()}  # strip removes spaces
30              lstTable.append(dicRow)  # adds each new dictionary row to a table
31          objFile.close()
32      except:
33          print("The file '"+objFile + "'does not exist.")  # if text file does not exist
```

Figure 2: Reading File into List

Step 2 – Display a Menu of Options

A "while loop" will be used to execute a block of code until the condition becomes "False". To start, a menu will be printed to show the different choices (Figure 3).

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

Figure 3: Menu Options

The "if", "elif", and "else" commands are sequences that are similar to an "If/Then" statement. Once a condition in the sequence is meet, then that block of code associated with the condition is performed and the rest of the

conditions are ignored. The script will continue to loop, providing the menu options, until the user selects option 5, at which point the "while" condition has been meet and the loop stops (Figure 4).

```
    # Step 3 - Show the current items in the table
    if (strChoice.strip() == '1'):
        # TODO: Add Code Here
        #print(lstTable)
        for row in lstTable:
            print(str(row["Task"]) + ","+ str(row["Priority"]))
        continue
    # Step 4 - Add a new item to the list/Table
    elif (strChoice.strip() == '2'):
        # TODO: Add Code Here
        continue
    # Step 5 - Remove a new item from the list/Table
    elif (strChoice.strip() == '3'):
        # TODO: Add Code Here
        continue
    # Step 6 - Save tasks to the ToDoToDoList.txt file
    elif (strChoice.strip() == '4'):
        # TODO: Add Code Here
        continue
    # Step 7 - Exit program
    elif (strChoice.strip() == '5'):
        # TODO: Add Code Here
        break  # and Exit the program
```

Figure 4: if, elif, sequence

Step 3 - Menu: Option 1, Show current data

When menu option 1 is selected, the current items in the "lstTable will be displayed. The "for loop" will be used to go through all the rows, printing each Task and Priority in the table (Figure 5) and (Figure 6).

```
    # Step 3 - Show the current items in the table
    if (strChoice.strip() == '1'):
        print("Task" + ":" + " Priority")  # Header
        for row in lstTable:
            print(str(row["Task"].title()) + ": " + str(row["Priority"].title()))  # print each row as loops
        continue
```

Figure 5: code: display items in table

```
    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 Windows: Low
Dishes: High
```

Figure 6: running script in Pycharm

Step 4 - Menu: Option 2, Add a new item

When menu option 2 is selected, the user will be requested to provide a Task and that Task's Priority, which will be appended to the Table "lstTable" (Figure 7) and (Figure 8).

```
55
56          # Step 4 - Add a new item to the list/Table
57          elif (strChoice.strip() == '2'):
58              strTask = input("Task: ")
59              strPriority = input("Priority: ")
60              lstTable.append({"Task": strTask, "Priority": strPriority})
61              continue
```

Figure 7: code: add new item

```
    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

Task: brush dog
Priority: medium
```

Figure 8: running script in Pycharm

Step 5 - Menu: Option 3, Remove an existing item

When menu option 3 is selected, the user will be requested to provide a Task, which will be removed from the list. If the task is removed, the user will be notified, otherwise the menu options will be displayed (Figure 9) and (Figure 10).

```
    # Step 5 - Remove a new item from the list/Table
    elif (strChoice.strip() == '3'):
        strTask = input("What task do you want to remove? ")  # get user input
        for row in lstTable:
            if row["Task"].lower() == strTask.lower():
                lstTable.remove(row)  # if row matches user input, remove row
                print(strTask + " removed")
        continue
```

Figure 9: code: delete item

```
    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 do you want to remove? clean windows
clean windows removed
```

Figure 10: running script in Pycharm

Step 6 - Menu: Option 4, Save data to File

When the menu option 4 is selected, the items in the "lstTable" will be saved back to the "ToDoList.txt" file. This will involve opening the text file, writing information to that text file, and closing the file. The information will not be saved until the file is closed (Figure 11) and (Figure 12)

```
    # Step 6 - Save tasks to the ToDoToDoList.txt file
    elif (strChoice.strip() == '4'):
        objFile = open("ToDoList.txt", 'w')  # create or open text file
        for row in lstTable:  # loop through the rows in the table
            objFile.write(str(row["Task"].title()) + ',' + str(row["Priority"].title()) + '\n')
        objFile.close()  # close the text file
        print("Data saved for future use")  # Display message to user
        continue
```

Figure 11: code: save to text file

```
    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 for future use
```

Figure 12: running script in Pycharm

Step 7 - Menu: Option 5, Exit Program

When the menu option 5 is selected, the script will end. Notice the "break" is used, which causes the "while" loop at the beginning of the script to become "False" and the loop to end (Figure 13).

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

Figure 13: code: Exit the program

Testing the Script in CMD line

Now the script will be run from Command Prompt to make sure it also runs as expected. Open a command prompt window "CMD". Type in "cd" and then the pathway to the directory where the python script is saved "C:_PythonClass\Module05\Assignment05". Click Enter. Next type "python" followed by the name of the python script "ToDoList.py". Click "Enter" again to see the script run. Provide the necessary inputs as if you are the user and verify the result is what you expected (Figure 14).

Figure 14: running script in Command Line

Open up the “ToDoList.txt” file to verify the results have changed based on the new inputs (Figure 15).

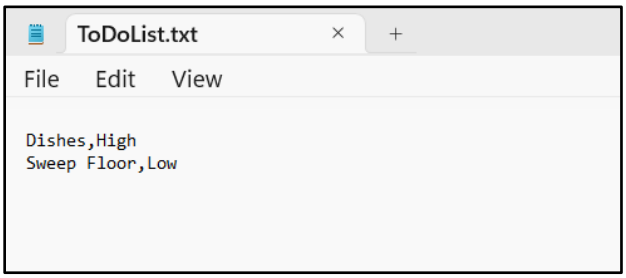

Figure 15: Testing in Command Line, results

Summary

In summary, this document goes over the steps needed to write a Python script that provides a menu where a user can make different selections to update a to-do-list, including viewing their current list, adding tasks, removing tasks and saving the latest version of their list. This script introducted dictonaries and reinforced learnings around loops, lists, and reading & writing to text files.