Rockey Aye

Instructor: Randal Root

IT FDN 100 B Su 18

August 13, 2018

**Assignment 5 Knowledge Document**

**Introduction**

This document shows the steps that were taken to perform Assignment 5. In addition, it includes the discoveries and lesson learned items relating to the concept of using list and dictionary. Python script, “test.py”, is designed to start with loading the initial list of tasks and priorities in the ToDo.txt. Then, the program is intended to read the data from ToDo.txt and convert it into the dictionary table format. After the conversion, the program executes the selection of 5 different options, which are independently and specifically designed to be performed and executed the specific sets of function. The key concept of Assignment 5 is to explore the list and dictionary, which are mutable, are designed, executed, and operated.to modify the data. (Dawson 123) In addition, “test.py” is designed to load each row of additional data, load those data in the dictionary “row”, and displays the data while saving the data back to ToDo.txt, as needed. (Root – Module 5) Appendix section shows “test.py” Python script for Assignment 5.

**Topic**

After completing the module, the list and dictionary are mutable and can be modified within the Python program. In addition, Assignment 5 uses the templates to show the concept of data, processing, and presentation section of writing a Python file. The Python script, “test.py” explores and demonstrates the concept of building, editing, and modifying the dictionary and list based on the user input. Using “.open”, “.write”, and “.close” function, a user can store the new data back into the file. (Root – Module 5)

**Requirements and Understandings**

The basic requirements of the program are following. (Root – Assignment 5 Module)

1. Create a text file called Todo.txt using the initial data
2. When the program starts, load each row of data from the ToDo.txt text file into a Python dictionary.
3. After you get the data in a Python dictionary, Add the new dictionary “row” into a Python list object.
4. Display the contents of the List to the user.
5. Allow the user to Add or Remove tasks from the list using numbered choices.
6. Save the data from the table into the Todo.txt file when the program exits

My understanding on the requirements on Assignment 5 is that “test.py” starts with loading the existing data from ToDo.txt file. While loading, the data is transformed to the “dictionary” table format. The executions of the program are based on the available selections to perform the operation. Currently, those options are: show current data, add a new item, remove an existing item, save data to file, and exit program. Depending upon a user’s decision, the modified data is either saved or disregarded. Furthermore, as part of the requirement, I have introduced while not function to keep looping the entry until the user provides the response to prevent the blank response.

**Summary of Python Script**

First, I created a “test.py” for Assignment 5 under C:\\_PythonClass folder. In order to run the program logically and properly, the “test.py” is started by building the pseudo code to outline the flow of the logical steps. Following is the logic of Assignment 5.

# Loading data from ToDo.txt in Python dictionary

# Display a menu of choices to the user

# Display current task ToDo list to the user

# Add a new item to the list/Table

# Remove a new item to the list/Table

# Save tasks to the ToDo.txt file

# Exit program

First step is to load the existing data from ToDo.txt file in the dictionary table format. Figure 1a shows the Python script setup the conversion of loaded data. I created “task, priority” to split the existing data into the dictionary. Task was designed to represent the key of the data while priority was designed to represent the value of the data. Then, I used “disRow” to create the row of dictionary data, which I used to represent under “lstTable” to form the dictionary table.

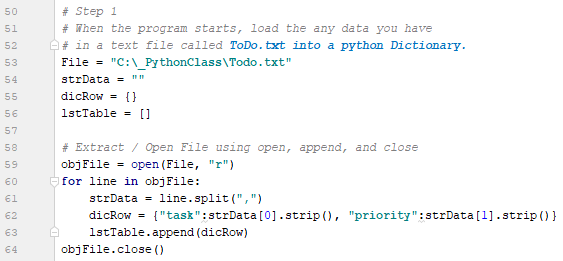


Figure 1a. Script Loading ToDo.txt file into Python Dictionary.

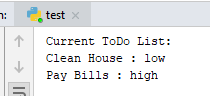


Figure 1b. Result of Python Script for Original Data.

Then, the “while loop” was set up to show the menu of choices by incorporating with if/elif/else statement, which is intended to either keep running the program or end the program due to a user’s decision. Figure 2 shows the concept of “while loop” to control the entire program to keep executing various functions / operations of a user’s choice until a user ends the program. “strChoice” with input function was used to designate the desired option.

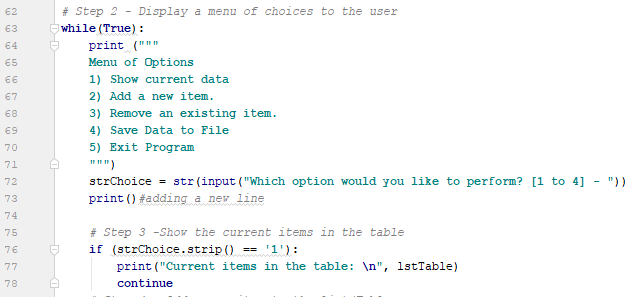


Figure2. Concept of “while loop” for the entire “test.py”.

After the setup of “while loop”, based on the selection of 5 different options, showing the current data of the dictionary table was designed as the first option. In order to accomplish the task, “if” statement was assigned as 1 to show the current data in dictionary table. In addition, I used “for loop” to account for all the data in the current or original condition of the file. Thus, “print” function was used to print out and show the current condition of data. After the execution is completed, due to “continue”, it will execute “while loop” to a user for the next user’s decision. Figure 3a shows the script and 3b shows the result of the script.

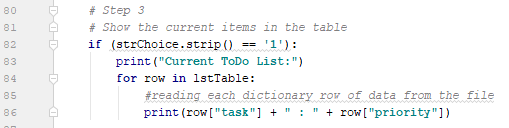


Figure 3a. Python Script of 1st Option for Showing Current Data.

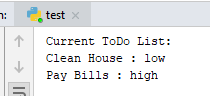


Figure 3b. Result of 1st Option.

The second option in the list of the choices is to add a new item / task and its priority to the existing list. In order to accomplish this task, “elif” function was used to assign as 2 as a user types “2”, it executes the function of adding a new item. Using input function, “task” and “priority” variable were assigned to request the input from a user. Since “while not” function was used, the script is preventing any user blank response. In addition, “.strip( ).title( )” was used to strip out extra space and capitalize the first letter of the word when it prints out and save the data. Once both entries were entered, “dicRow” was assigned immediately to convert those new entries in the dictionary form by assigning with the following: {"task":strTask,"priority":strPriority}. After that, “.append” function was used to append into the original list in “lstTable”. Using print function, “test.py” prints the new data entry along with the updated data in the table. I used “print” function to print out he added / updated list to let a user know what items are in the list. The “continue” function was used; thus, at the end of the execution, the program executes the “while loop” for a user’s decision on the next step. Figure 4a shows the Python script of adding a new task and its priority and 4b shows the result of the script that is exactly how the script is designed.

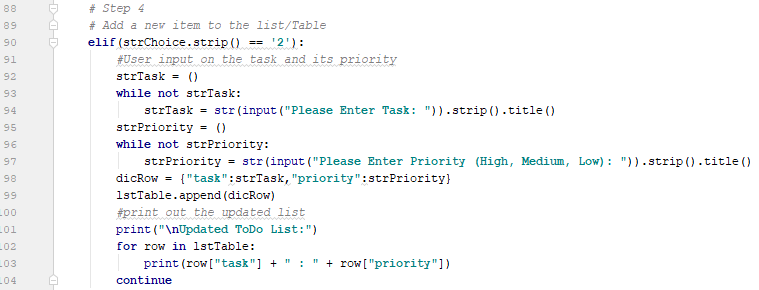


Figure 4a. Python Script of 2nd Option for Adding Task.

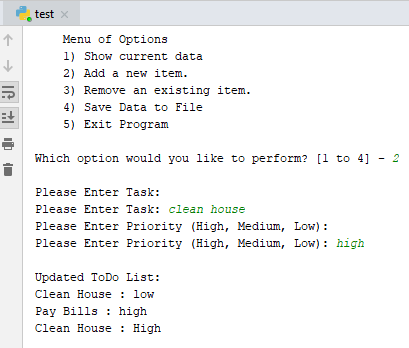


Figure 4b. Result of 2nd Option.

As for the third option of removing an existing item, the concept of item removal is similar to the script designed for adding a new item under 2nd option. The “elif” was used to assign as 3 so that when a user enters 3, the script executes the task removal operation. I used the same setup of code for requesting “task”, “priority”, and “while not loop” function to be deleted by using the combination of a Boolean function and input function. Using input function, “strDelete” was assigned to request the input from a user. Since “while not” function was used, the script is preventing any user blank response. In addition, “.strip( ).title( )” was used to strip out extra space and capitalize the first letter of the word when it prints out and save the data. “BlankRemoved”, as the Boolean function, I designed a script to delete if and when a user enters a task that is matched in the list. In order to check the entire dictionary, using “while (intRow < len(lstTable)):” function, I designed to check every row in the data table. When the entry matches with one of the tasks in the list, it will delete the task from the list. Otherwise, the Program is designed to provide the “can’t find” message under the “if / else” and print function. Figure 5a shows the Python script. In Figure 5b, it shows the test case of removing “move chair” from the list and the program successfully removes the task from the list. Figure 5c shows the test case of removing “open door”, which is not even in the original or updated list, and the program generates the message of “is not found” message. Thus, it proves that the code that I designed is working properly.

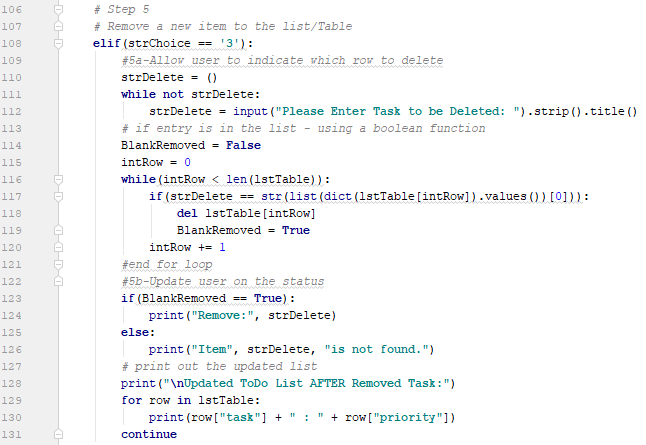


Figure 5a. Python Script of 3rd Option for Deleting Task.

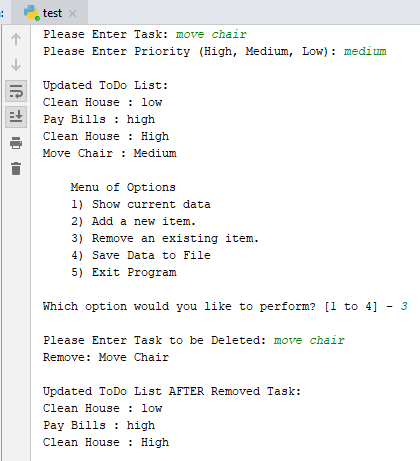


Figure 5b. Result of Deleted Task when Task is **Found** to be Deleted.

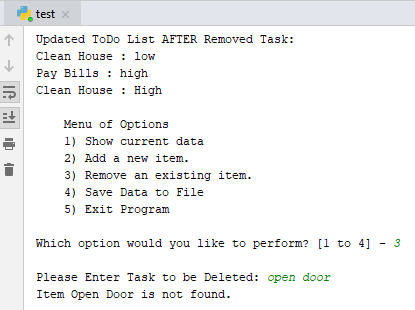


Figure 5c. Result of Deleted Task when Task is **Not Found** to be Deleted.

For the fourth option of saving data by updating ToDo.txt file, the “elif” function was assigned as 4 to execute the option of saving function. When a user decides to save, I designed the script to execute the input function asking a user’s decision on saving the data or not saving it. I used “if / else” statement with the combination of “strSave” input function so that if a user decides to save the script is designed to execute “open”, “.write with ‘w’”, and “.close” operation to create a file, update, and save the data. Figure 6a shows the script of “if / else” statement for saving or disregarding the data. Figure 6b shows the performance of the script. Figure 6c shows the saved data in ToDo.txt.

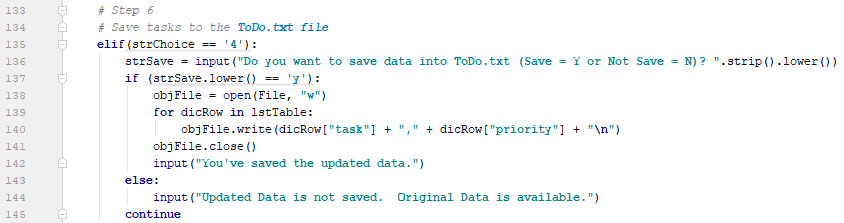


Figure 6a. Python Script of 4th Option for Saving Task in ToDo.txt.

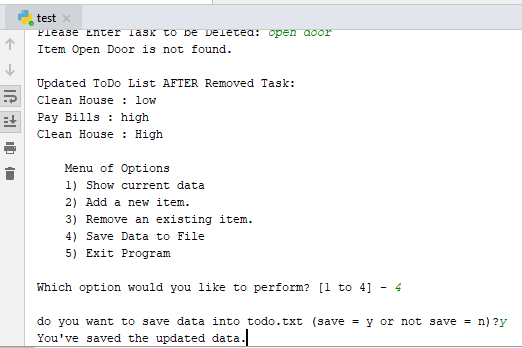


Figure 6b. Result of Saving Script.

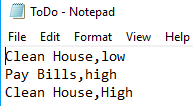


Figure 6c. Updated Data in ToDo.txt.

The final operation is to exit the program if a user decides to end the program. In this step, the Python script was designed to end the “while loop” by using “else” function to “break” the “while loop”. Thus, the “elif” function was assigned as 5 to execute the program to end the program. Figure 7a shows the Python Script and 7b shows the result of “break” function.



Figure 7a. Python Script of 5th Option for Ending the Program.

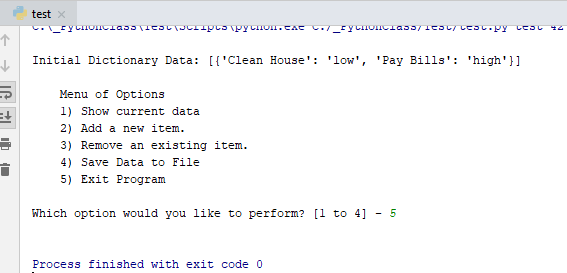


Figure 7b. Result of Ending the Program.

**Test Case Scenario**

Following is the description of test case scenario for evaluating the Python Script for Assignment 5. The test case scenario was designed to align with the requirements of Assignment 5.

Step 0 is to load initially load the converted dictionary data from ToDo.txt.

Step 1 is to perform Option 2 to add “clean table” as the task and “medium” as its priority.

Step 2 is to perform Option 2 to add “buy grocery” as the task and “high” as its priority.

Step 3 is to perform Option 1 to see the latest task list.

Step 4 is to perform Option 3 to remove “buy grocery” and “high”.

Step 5 is to perform Option 4 to save the data.

Step 6 is to perform Option 5 to end the program.

Step 7 is to validate the updated list in ToDo.txt.

**Result of Test Case Scenario**

I have summarized the results and observations of the test case scenario in Table 1 with the screenshot. Validating the code improves the consistency and reliability of the code performance.

Table 1. Test Result and Validation of the Program.

|  |  |  |
| --- | --- | --- |
| **Step** | **Result Description** | **Result** |
| 0 | By selecting Option 1, the initial data was loaded successfully. |  |
| 1 | By selecting Option 2, the first entry of “clean table” and “medium” were made successfully. The current ToDo list was successfully updated with both entries. The first letter of each word was capitalized. |  |
| 2 | By selecting Option 2, the second entry of “buy grocery” and “high” were made successfully. The current ToDo list was successfully updated with both entries. The first letter of each word was capitalized. |  |
| 3 | By selecting Option 1, the latest ToDo list reflected all the updated entries. |  |
| 4 | By selecting Option 3, “buy grocery” is deleted from the updated ToDo dictionary list. It is by the latest updated ToDo list. |  |
| 5 | By selecting Option 4,The updated list was saved into ToDo.txt successfully when “y” was entered to save the data. |  |
| 6 | The program was ended successfully when Option 5 was selected. |  |
| 7 | ToDo.txt was updated successfully with the latest updated data. |  |

Work Cited

Dawson, Michael. *Python Programming for the Absolute Beginner*. Third Edition. Boston: Course Technology Cengage Learning, 2010. Print.

Root, Randal. "Assignment 5 Module." *IT FDN 100 B Su 18: Foundations Of Programming: Python.* UW PCE, Web. August 7, 2018. <https://canvas.uw.edu/courses/1243112>.

Root, Randal. "Python Module 05." *Python Module 05.* You Tube, Oct 1, 2016. Web. July 31, 2018. <https://www.youtube.com/watch?v=UigL0Cd4BRk&list=PLfycUyp06LG_Ui56yzznsHhfEosKZSahX>.

Appendix

*#-------------------------------------------------#  
# Title: Working with Dictionaries  
# Dev: Rockey Aye  
# Date: Aug 13, 2018  
# Change Log: New  
# Change Description: Rockey Aye, 8/9/2016, Create code to complete assignment 5  
#https://www.tutorialspoint.com/python/python\_dictionary.htm  
#-------------------------------------------------#  
  
#-- Data --#  
# declare variables and constants  
# objFile = An object that represents a file  
# task = Key of dictionary for the row of data from the file  
# priority = Value of dictionary for the row of data from the file  
# dicRow = A row of data separated into elements of a dictionary {Task,Priority}  
# lstTable = A dictionary that acts as a 'table' of rows  
# dicNewRow = A New Dictionary row based on New Entry of Task and Priority  
# delRow = Task and Priority to be deleted  
  
#-- Input/Output --#  
# User can see a Menu (Step 2)  
# User can see data (Step 3)  
# User can insert or delete data(Step 4 and 5)  
# User can save to file (Step 6)  
  
#-- Processing --#  
# Step 1  
# When the program starts, load the any data you have  
# in a text file called* ***ToDo.txt into a python Dictionary.****# Step 2  
# Display a menu of choices to the user  
  
# Step 3  
# Display current task* ***ToDo list to the user****# Step 4  
# Add a new item to the list/Table  
  
# Step 5  
# Remove a new item to the list/Table  
  
# Step 6  
# Save tasks to the* ***ToDo.txt file****# Step 7  
# Exit program  
#-------------------------------  
  
# Step 1  
# When the program starts, load the any data you have  
# in a text file called* ***ToDo.txt into a python Dictionary.***File = **"C:\\_PythonClass\Todo.txt"**strData = **""**dicRow = {}  
lstTable = []  
  
*# Extract / Open File using open, append, and close*objFile = open(File, **"r"**)  
**for** line **in** objFile:  
 strData = line.split(**","**)  
 dicRow = {**"task"**:strData[0].strip(), **"priority"**:strData[1].strip()}  
 lstTable.append(dicRow)  
objFile.close()  
  
*# 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 4] - "**))  
 print()*#adding a new line  
  
 # Step 3  
 # Show the current items in the table* **if** (strChoice.strip() == **'1'**):  
 print(**"Current ToDo List:"**)  
 **for** row **in** lstTable:  
 *#reading each dictionary row of data from the file* print(row[**"task"**] + **" : "** + row[**"priority"**])  
  
 *# Step 4  
 # Add a new item to the list/Table* **elif**(strChoice.strip() == **'2'**):  
 *#User input on the task and its priority* strTask = ()  
 **while not** strTask:  
 strTask = str(input(**"Please Enter Task: "**)).strip().title()  
 strPriority = ()  
 **while not** strPriority:  
 strPriority = str(input(**"Please Enter Priority (High, Medium, Low): "**)).strip().title()  
 dicRow = {**"task"**:strTask,**"priority"**:strPriority}  
 lstTable.append(dicRow)  
 *#print out the updated list* print(**"\nUpdated ToDo List:"**)  
 **for** row **in** lstTable:  
 print(row[**"task"**] + **" : "** + row[**"priority"**])  
 **continue** *# Step 5  
 # Remove a new item to the list/Table* **elif**(strChoice == **'3'**):  
 *#5a-Allow user to indicate which row to delete* strDelete = ()  
 **while not** strDelete:  
 strDelete = input(**"Please Enter Task to be Deleted: "**).strip().title()  
 *# if entry is in the list - using a boolean function* BlankRemoved = **False** intRow = 0  
 **while**(intRow < len(lstTable)):  
 **if**(strDelete == str(list(dict(lstTable[intRow]).values())[0])):  
 **del** lstTable[intRow]  
 BlankRemoved = **True** intRow += 1  
 *#end for loop  
 #5b-Update user on the status* **if**(BlankRemoved == **True**):  
 print(**"Remove:"**, strDelete)  
 **else**:  
 print(**"Item"**, strDelete, **"is not found."**)  
 *# print out the updated list* print(**"\nUpdated ToDo List AFTER Removed Task:"**)  
 **for** row **in** lstTable:  
 print(row[**"task"**] + **" : "** + row[**"priority"**])  
 **continue** *# Step 6  
 # Save tasks to the* ***ToDo.txt file* elif**(strChoice == **'4'**):  
 strSave = input(**"Do you want to save data into ToDo.txt (Save = Y or Not Save = N)? "**.strip().lower())  
 **if** (strSave.lower() == **'y'**):  
 objFile = open(File, **"w"**)  
 **for** dicRow **in** lstTable:  
 objFile.write(dicRow[**"task"**] + **","** + dicRow[**"priority"**] + **"\n"**)  
 objFile.close()  
 input(**"You've saved the updated data."**)  
 **else**:  
 input(**"Updated Data is not saved. Original Data is available."**)  
 **continue  
  
 elif** (strChoice == **'5'**):  
 **break** *#and Exit the program*