**Sandra D Harvie**

November 17th 2020

Foundations of Programming: Python

**MODULE 05: Lists & Dictionaries**

**ASSIGNMENT 05: Lists, Dictionaries, and GitHub**

1. **Introduction**

This paper documents my learning from Module 5, the goal of which was to learn about creating scripts using Lists and Dictionaries.

I have demonstrated my learning by

* Answering the questions provided in the Assignment
* Modifying the provided starter script to manage a "ToDoList"
* Running the script within both PyCharm and a command shell
* Creating a new Repository in GitHub and uploading my files to there for peer review
* Post my GitHub link to the message board
* Peer reviewing another student's files in GitHub and adding my comments to their post

1. **Background**

I watched the video "Mod05 Course Video" and followed along with the class notes "\_Mod5Python ProgrammingNotes.pdf". I also watched the Session 5 Q&A recording.

I read the fifth chapter of "Python Programming, 3rd Edition" entitled "Lists and Dictionaries: The Hangman Game".

I reviewed the suggested web tutorials which gave further information regarding Lists, Dictionaries and Reading Files. I also watched the additional YouTube video on reading from a text file. Finally I read through the provided GitHub article and gained some really interesting background on what GitHub is and some new (to me) cool terms.

1. **Questions**

• What is the difference between a List and a Dictionary?

Lists (and tuples and strings) use subscripts that are numerical whereas a Dictionary uses subscripts that are character data. Also a List is defined within square brackets [] while a Dictionary is defined within curly brackets {}.

• What is the between an Index and a Key?

An Index is numerical whereas a Key is character data.

• How do you read data from a file into a List?

1. Open (read) the file containing the list: objFile = open(strFile, "r")
2. Use something like a for loop to extract all the rows of data from the file.
3. Use the split function to split the data by comma and return a list: lstRow = row.split(",")
4. A simple way to display the data would be: print(lstRow) but it's also useful to remove any inadvertent spaces, and perform any other desired formatting at this stage.
5. Close the file: objFile.close()

• How do you read data from a file into a Dictionary?

Reading data from a file into a Dictionary is initially similar to reading into a List but then deviates at step d below.

1. Open (read) the file containing the list: objFile = open(strFile, "r")
2. Use something like a for loop to extract all the rows of data from the file.
3. Use the split function to split the data by comma and return a list: lstRow = row.split(",")
4. Use the keys for the dictionary to extract the data from the List:

dicRow = {"id":lstRow[0],"name":lstRow[1],"email":lstRow[2].strip()}

1. Append this new row of data to the table: lstTable.append(dicRow)
2. Close the file: objFile.close() Not technically required by Python but best practice.
3. Print the table to to the screen: print(lstTable)

• What is the programming pattern called “Separations of Concerns?”

SoC is a design principle in which the code is separated into distinct sections making it easier to understand. It also makes it easier to use parts of your script in other programs. Generally three sections are used; data, processing, and presentation. It is not always possible to follow this precisely depending on where code can be input in the script, but it is a best practice.

• How would you use a function to organize your code?

A Function is a named set of one or more statements. It is useful because it allows you to collect your Functions and store them in one location - possibly all together at the top of the script or even in a separate script altogether, a module. These Functions are then called when required. The further enables Separation of Concerns, and also the reusing of Functions in other programs.

• Why is a script template useful?

A script template is useful because it removes the requirement to recreate your script title block and sub headers every time you start a new script. This helps ensure a uniform layout of metadata that could be harvested later. It also decreases the chances you forgot something.

• Why is error handling using Try-Except recommended?

Try-Except allows the programmer to input custom user friendly error messages that allows the user to debug themselves without having to resort to other avenues like tech support.

• What is GitHub, and why is it used?

GitHub is a version control system. It allows you to store your code online where other developers can access it. This enables easy collaboration, updates and revision. Most importantly GitHub keeps track of these revisions so that everyone is clear what version they are using. Shared folders called Repositories are used to store all the versions for a particular project.

1. **Assignment**

A starter script has been provided that manages a "ToDoList" csv file made up of 2 columns; Task and Priority. Modify this script so that the Python Dictionary object is used to display the file contents add/remove Tasks and save the list back to a file.

* Create a sub folder called Assignment05
* Create a new project in PyCharm within the Assignment05 folder
* Add the starter file, "Assigment05\_Starter.py," to the project
* Add code to the script to perform the task described above
* Run the script in BOTH PyCharm and an OS command/shell
* Document the knowledge gained

1. **Discussion**

**Figure 1** displays my code in the file "*Assignment05.py*" in the PyCharm IDE.

The assignment was divided into 7 steps, each highlighted within the starter code with "*TODO: Add Code Here*" except for step 2 which required no additional code. Therefore I have divided my discussion into 7 associated sections. After completing each task I modified the code to read "*TODO: DONE*". Being able to track these *TODO* comments is very useful.

The program written in Assignment 4 overwrote the file every time. In Assignment 5 we start by reading the existing file in so that we can read/write the file over and over without it starting from scratch.

STEP ONE

* The objective of step one is to open the text file and display any data within it. I used the Try-Except statement to either display the list of tasks or display the message

The file 'ToDoList.txt' has not been created yet.

* The split function reads data in as a list
* The dictionary object (with 2 keys defined) extracts the data from the list
* The data is then appended to the list table and displayed

STEP TWO

* Step two creates the menu displayed to the user
* The use of the while loop allows the menu to be looped repeatedly until the user chooses "Exit".
* Since the starter code contains this, no additional code required here

STEP THREE

* The objective of step three (menu item 1) is to display Tasks in the file
* To cover the possibility that the file is empty I included the statement

Current Task list (if blank the Task List is empty):

* The for loop reads all rows of data in the table and displays them to the user with appropriate formatting

STEP FOUR

* The objective of step four (menu item 2) is to enable the user to add new Tasks/Priorities
* I defined two user inputs (Task and Priority)
* Further on in step five it became clear that it would be easier to delete tasks if case sensitivity was not a concern. Therefore in step four I have used .lower() to convert the user input to lower case.
* The user can select response 2 multiple times and each new set of data will be appended to the table

STEP FIVE

* The objective of step five (menu item 3) is to enable the user to delete Tasks
* The program asks the user "Which Task?"
* Case sensitivity has been removed from the search through the use of .lower()

STEP SIX

* The objective of step six (menu item 4) is to save the data to the ToDoList.txt file
* The file is opened and a for loop is used to write each row of data from the table to the file
* A statement is displayed to the user "File has been saved."

STEP SEVEN

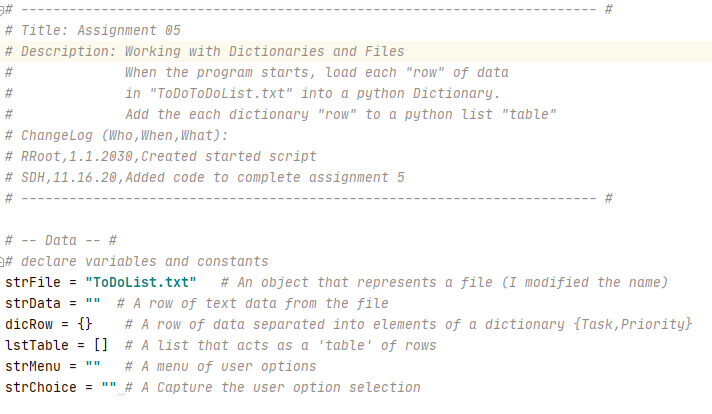
* The objective of step seven (menu item 5) is to exit Python
* Since the starter code includs the break statement, no additional code is required here

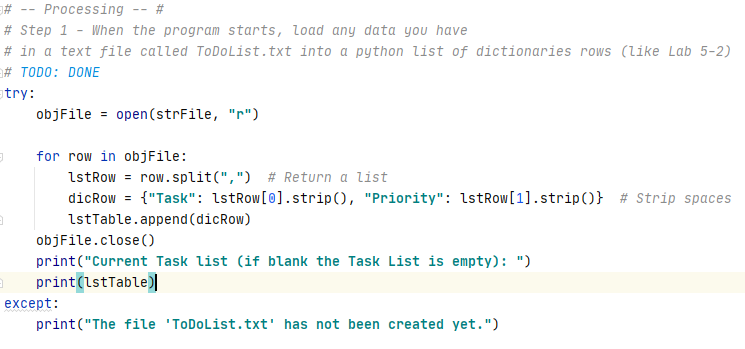
**Figure 2** displays the code after being ran in the PyCharm run window.

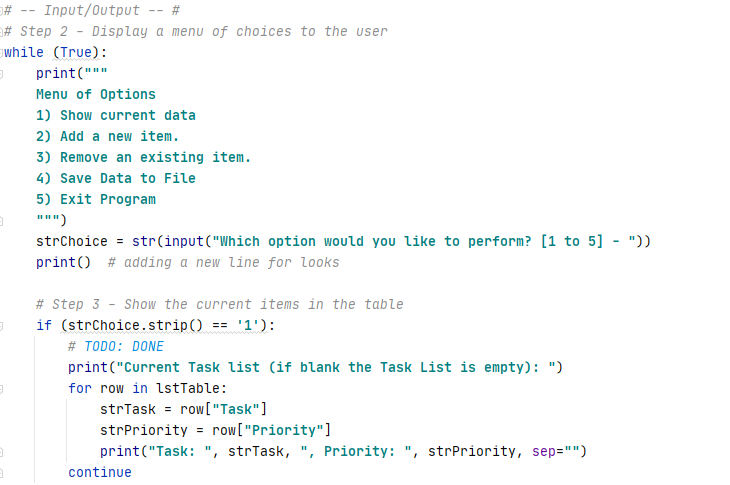
**Figure 3** displays the code after being ran at the command prompt.

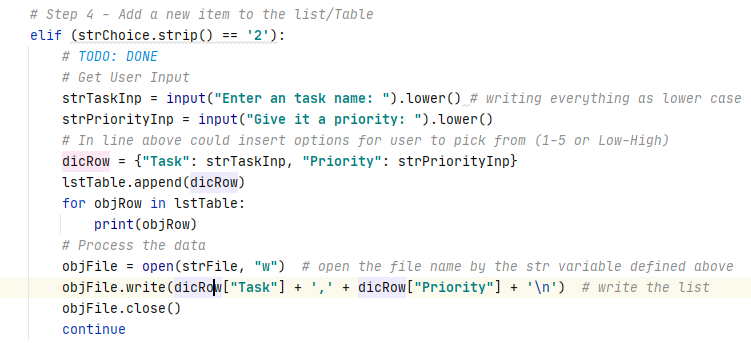
**Figure 4** displays the text file "*ToDoList.txt*" after I have completed running the code.

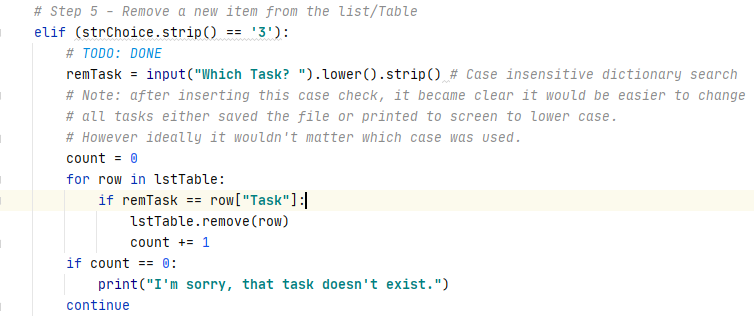
1. **Image Captures**





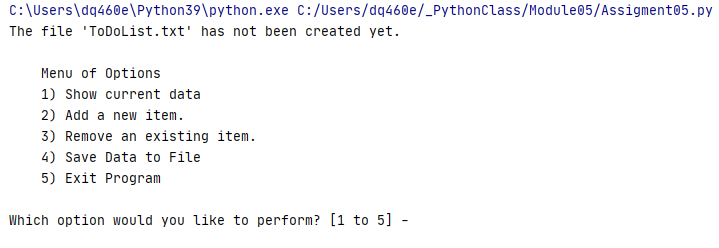


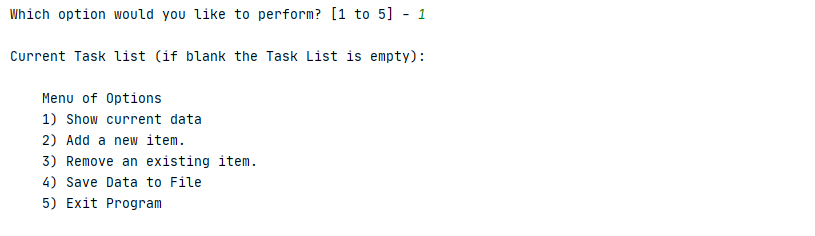


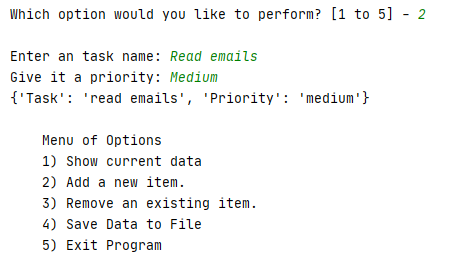


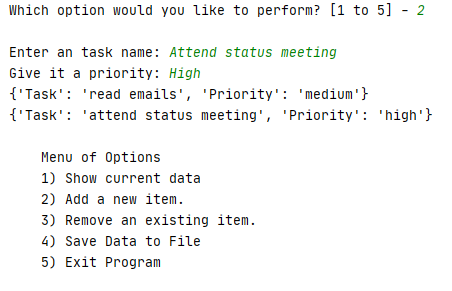


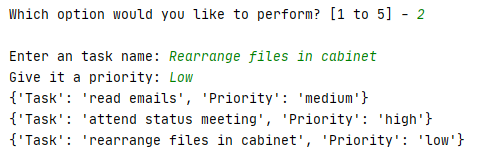
***Figure 1: Script file in PyCharm (HomeInventory.py)***

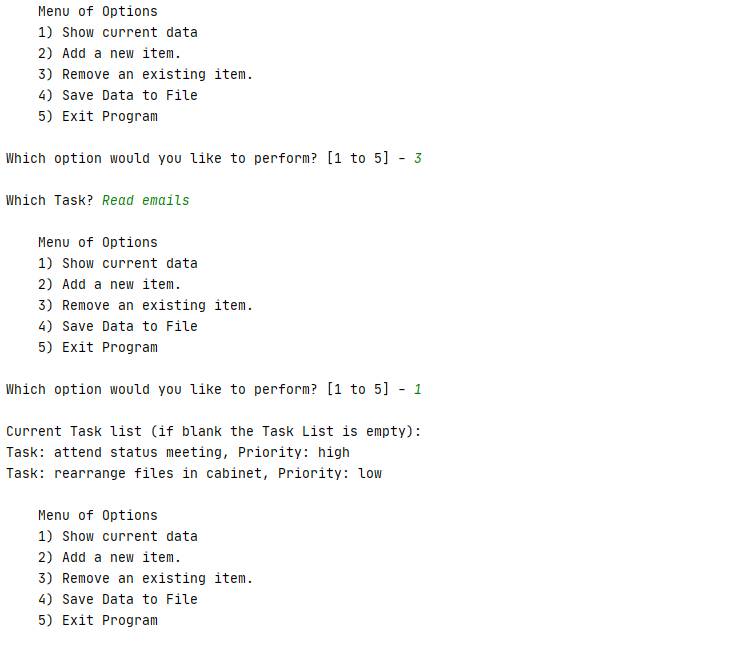


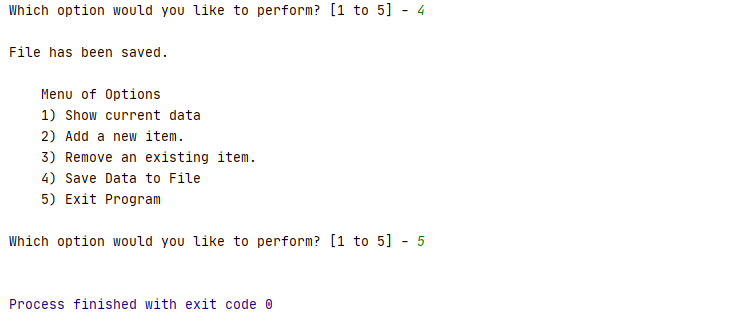




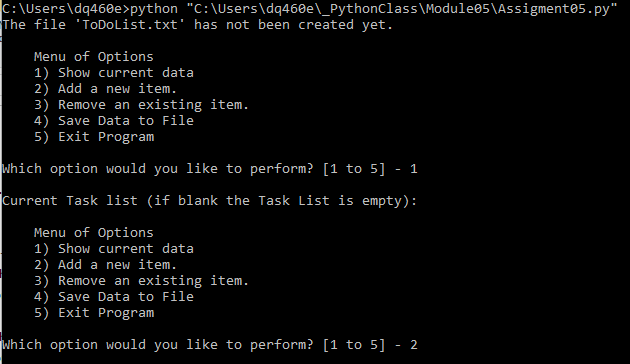


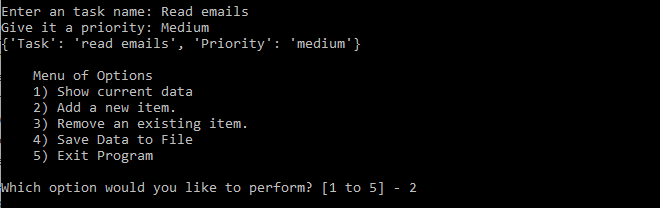


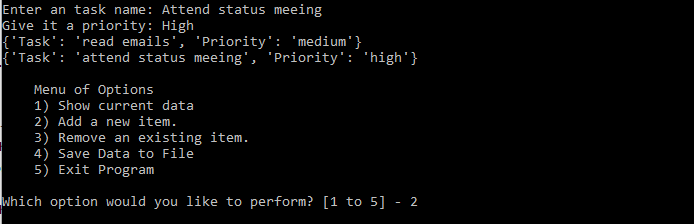


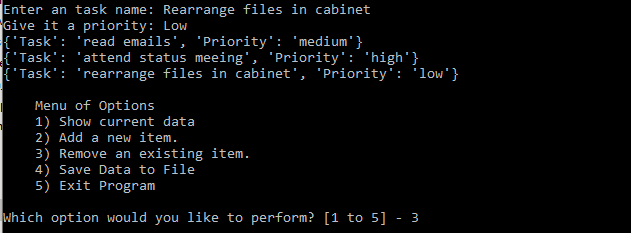


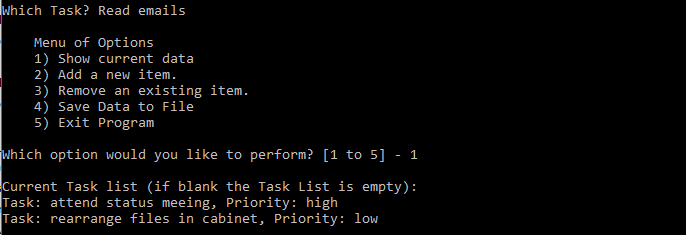
***Figure 2: Script run window in PyCharm (HomeInventory.py)***

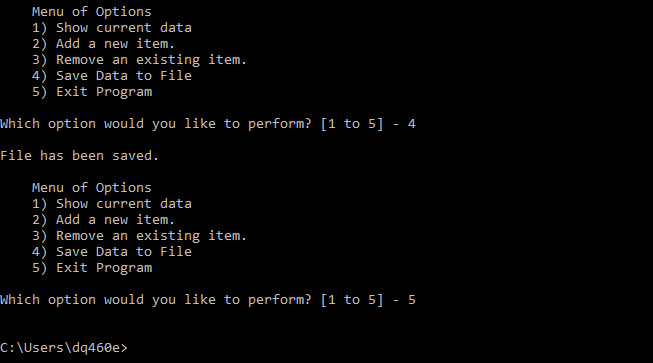




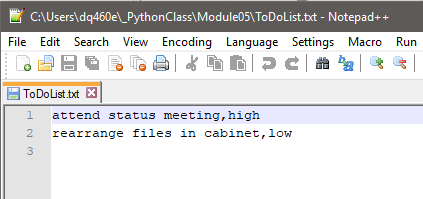








***Figure 3: Script running from a Command Shell***



***Figure 4: Output Text File (HomeInventory.txt)***

1. **Conclusion**

In completing Module 5 I have learned how to create scripts that work with rows of data in the form of Dictionaries. I have demonstrated this learning by modifying the given starter script in PyCharm to use the Python Dictionary object to enable a user to write/modify/read a To Do list stored as a text file. This is the first assignment/task that has given me the feeling that I've programmed a "useful" piece of code.