Ryan L. Blake

10 August 2021

IT FDN 100

Assignment 05

https://github.com/rblake50/IntroToProg-Python

**Lists, and Dictionaries, and GitHub, oh my!**

# Introduction

Building on Assignment 04, this assignment explores further applications of Python lists, specifically where elements of the list are dictionary objects. The script also builds on the previous topics of text file input and output and manipulation. Finally, this document explores the basic capabilities of GitHub as a way to save coding artifacts for community participation.

# Topics

* More about lists
* Dictionaries
* GitHub

# Summary

Lists and dictionaries, along with tuples, have several fundamental features in common. At a high level, each object contains items that can be referred to and adjusted. However, the technical difference involves topics like mutability and sequences. Chapter 5 of the textbook (p. 140) does a good job explaining how dictionaries use a *key-value pair* instead of a sequence to store information. Practically, this means a dictionary element has a “key” (like an identifier) and a “value” (information associated with the key). Keys must be entirely unique inside a dictionary.

Outside the textbook definition, I like to think of keys as “headers” for information. While a list would only contain information, dictionaries would pair the information with a header or subject for the information. The example below illustrates a few of the differences between lists and dictionaries with the same information.

Text

Description automatically generated

Figure : Illustration of differences between lists and dictionaries.

Most notably, accessing values within the collections looks different depending on the type of collection. Lists use listName[index] as the access method with an integer as the index value. Dictionaries use dictName[key] with a string as the key value.

The assignment exercises another interesting concept: a list where the elements are… dictionaries. This technique offers some neat organization of data. Python is efficient at both reading and storing data from lines of text.

To read data from the text file, the script reads each line as a separate list item and assumes a comma separated list. That assumption allows the split(“,”) function to be used (with a comma as the delimiter) such that the two pieces of information—Task and Priority—can be accessed element-wise. Then, these elements are stored to a local dictionary variable with “Task” and “Priority” as the keys. Finally, the local dictionary variable is appended to a larger list that contains the dictionaries as elements. Since this description may be tough to follow with the transition of list to dictionary back into a different list, the code is copied below:

*# Step 1 - When the program starts, load the any data you have  
# in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)*objFileText = open(objFile,**"r"**)  
**for** strData **in** objFileText:  
 *# Assume comma separated values* strTask = strData.split(**","**)[0].strip()  
 strPriority = strData.split(**","**)[1].strip()  
 dicRow = {**"Task"**:strTask,**"Priority"**:strPriority}  
 lstTable.append(dicRow)  
objFileText.close()

At this point, the code has read the existing text file and stored the data locally in an organized way. From here, a friendly while loop can present a list of user options similar to Assignment 04. The menu tasks are fairly straightforward, though a fey key observations will be mentioned below, as they could appear in future scripts or assignments.

## Observation 1: Removing List Items

Unlike previous assignments, this assignment has an option for deleting data. While the code to accomplish this task is short, it took me a few tries to “get it right.” The reason for this was some confusion over using in versus the == comparison operator. I ran a few test cases while using two different tasks: one being “cleaning,” and another being “cleaning garage.” I soon noticed that using in on Python would recognize “cleaning” to be the same thing as the first element that simply contained the word “cleaning.” This prompted me to add a direct check of the dictionary value within the list using == before deleting. I used the same if statement to add a message if the user entered an item that was not found.

*# Step 5 - Remove a new item from the list/Table***elif** (strChoice.strip() == **'3'**):  
 strTask = input(**"Which item would you like to remove? Type the \"Task\": "**)  
 **for** item **in** lstTable:  
 **if** strTask == item[**"Task"**]:  
 boolFound = **True** lstTable.remove(item)  
 print(**"Item deleted"**)  
 **if** boolFound == **False**:  
 print(**"The Task you entered was not found..."**)  
 input(**"Press ENTER to continue"**)  
 boolFound = **False  
 continue**

## Observation 2: Writing Versus Appending

From the feedback on Assignment 03, I learned I incorrectly chose to (over)write data instead of append it. However, with this assignment, I recognized that writing truly was the best option. This is because the same text file was being used for initial input and final output. So, if a user decided to not save the output, the text file would remain untouched, even if several changes were made within the Python script. This presents a powerful example of using Python to read text data and then manipulate it inside of an IDE through local variables.

Because the changes were locally managed through a list of dictionaries, the list elements could be simply written (instead of appended) to the text file at the end. This, perhaps, simplifies the number of times where the file must be accessed. I can think of an alternatively successful code that would be more complicated to read and adjust the text file each time. By having the data in local, understood objects, it can simplify the I/O steps.

*# Step 6 - Save tasks to the ToDoToDoList.txt file***elif** (strChoice.strip() == **'4'**):  
 objFileText = open(objFile,**"w"**)  
 **for** item **in** lstTable:  
 objFileText.write(item[**"Task"**]+**", "**+item[**"Priority"**]+**"\n"**)  
 objFileText.close()  
 input(**"Data has been saved to file. Press ENTER to return to menu."**)  
 **continue**

The entire code can be tested in both PyCharm and the Python command shell. The text from the PyCharm console is copied below for some sample data:

|  |
| --- |
| C:\\_PythonClass\Assignment06\venv\Scripts\python.exe C:/\_PythonClass/Assignment05/Blake\_Assigment05.py  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  ==============  sweeping, 5  brushing teeth, 10  washing dishes, 11  Press enter to return to menu.  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  What task would you like to add? packing suitcase  What priority is this task? 6  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  ==============  sweeping, 5  brushing teeth, 10  washing dishes, 11  packing suitcase, 6  Press enter to return to menu.  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] -  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  Which item would you like to remove? Type the "Task": brushing teeth  Item deleted  Press ENTER to continue  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  ==============  sweeping, 5  washing dishes, 11  packing suitcase, 6  Press enter to return to menu.4  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 has been saved to file. Press ENTER to return to menu.  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  Good-bye! |

The text file is validated on Notepad:

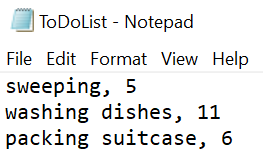


Figure : Text file output from script.

The command shell can also be used. An example is copied below.

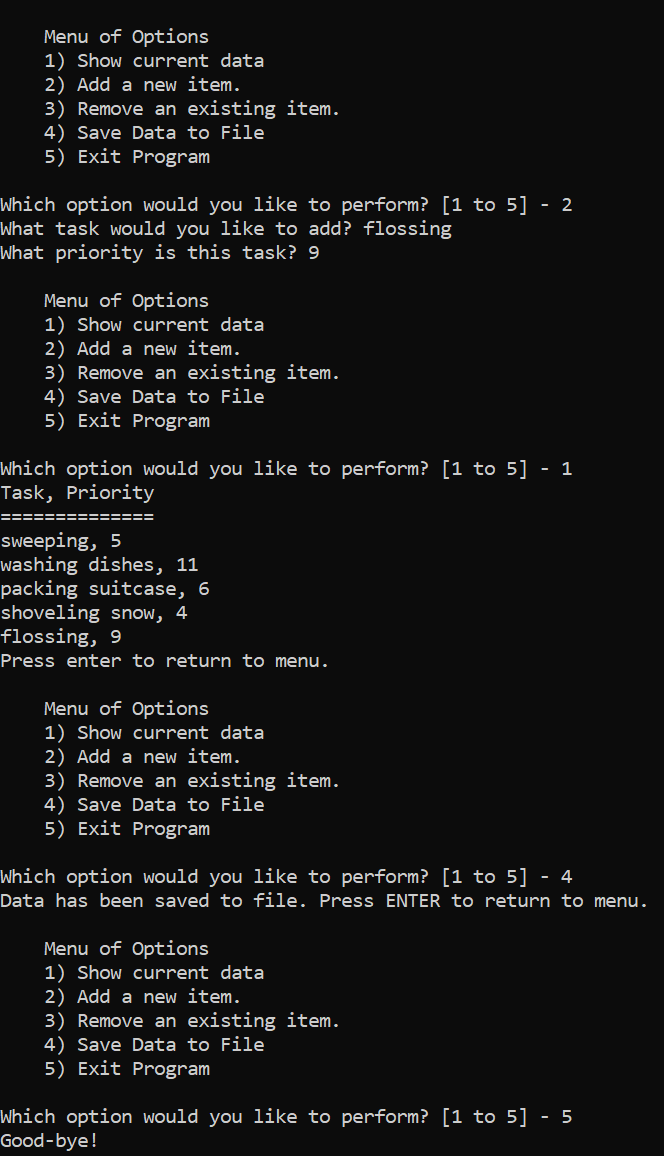


Figure : Command shell output.

Once finished with the code, I explored GitHub. GitHub acts as an online repository of code where developers can commit new code and changes for documentation and collaboration. GitHub offers a simple way to register and organize code repositories. I am excited to use GitHub in future exercises.

All told, this assignment stretched my understanding of Python collections and how they can be used together for efficient, effective code. Further, the Assignment introduced the utility of GitHub as a way to manage and document code.

# Appendix: Python Script

*# ------------------------------------------------------------------------ #  
# 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 the each dictionary "row" to a python list "table"  
# ChangeLog (Who,When,What):  
# RRoot,1.1.2030,Created started script  
# rblake50,7/31/21,Added code to complete assignment 5  
# GitHub link: https://github.com/rblake50/IntroToProg-Python  
# ------------------------------------------------------------------------ #  
  
# -- Data -- #  
# declare variables and constants*objFile = **"ToDoList.txt"** *# An object that represents a file*strData = **""** *# A row of text data from the file*strTask = **""** *# First element of strData (separated by comma)*strPriority = **""** *# Second element of strData (separated by comma)*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*boolFound = **False** *# Indicator if task is found in the list  
  
# Initialize*strMenu = **"""  
 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  
 """***# -- Processing -- #  
# Step 1 - When the program starts, load the any data you have  
# in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)*objFileText = open(objFile,**"r"**)  
**for** strData **in** objFileText:  
 *# Assume comma separated values* strTask = strData.split(**","**)[0].strip()  
 strPriority = strData.split(**","**)[1].strip()  
 dicRow = {**"Task"**:strTask,**"Priority"**:strPriority}  
 lstTable.append(dicRow)  
objFileText.close()  
  
*# -- Input/Output -- #  
# Step 2 - Display a menu of choices to the user***while** (**True**):  
 print(strMenu)  
 strChoice = str(input(**"Which option would you like to perform? [1 to 5] - "**))  
 *#print() # adding a new line for looks  
 # Step 3 - Show the current items in the table* **if** (strChoice.strip() == **'1'**):  
 print(**"Task, Priority\n=============="**)  
 **for** item **in** lstTable:  
 print(item[**"Task"**]+**", "**+item[**"Priority"**])  
 input(**"Press enter to return to menu."**)  
 **continue** *# Step 4 - Add a new item to the list/Table* **elif** (strChoice.strip() == **'2'**):  
 strTask = input(**"What task would you like to add? "**)  
 strPriority = input(**"What priority is this task? "**)  
 dicRow = {**"Task"**:strTask,**"Priority"**:strPriority}  
 lstTable.append(dicRow)  
 **continue** *# Step 5 - Remove a new item from the list/Table* **elif** (strChoice.strip() == **'3'**):  
 strTask = input(**"Which item would you like to remove? Type the \"Task\": "**)  
 **for** item **in** lstTable:  
 **if** strTask == item[**"Task"**]:  
 boolFound = **True** lstTable.remove(item)  
 print(**"Item deleted"**)  
 **if** boolFound == **False**:  
 print(**"The Task you entered was not found..."**)  
 input(**"Press ENTER to continue"**)  
 boolFound = **False  
 continue** *# Step 6 - Save tasks to the ToDoToDoList.txt file* **elif** (strChoice.strip() == **'4'**):  
 objFileText = open(objFile,**"w"**)  
 **for** item **in** lstTable:  
 objFileText.write(item[**"Task"**]+**", "**+item[**"Priority"**]+**"\n"**)  
 objFileText.close()  
 input(**"Data has been saved to file. Press ENTER to return to menu."**)  
 **continue** *# Step 7 - Exit program* **elif** (strChoice.strip() == **'5'**):  
 input(**"Good-bye!"**)  
 **break** *# and Exit the program*