Erik Knighton

5/13/2020

IT FDN 100 A

Assignment05

<https://github.com/erikjkUW/IntroToProg-Python>

­Managing Task Manager Tasks

# Introduction

I have to admit, beginning with someone else's code was quite the trip. Obviously there was nothing wrong with the existing code, the comments, etc. However, I felt like I was stumbling through every line, t­rying not to disturb the foundations already established. I know that was the intention, but I truly did not anticipate how disorienting it would be for the majority of the assignment. With the module functioning properly, and time for writing, the mental fatigue is real. I will explain below what stumbling blocks I came across, but overall I find the results agreeable.

# Document Woes

Fundamentally speaking I had no trouble with the requested project, but I did get stuck a couple of times, primarily on turning the .txt document into dictionary entries. First, though, I tackled what was familiar: structuring the dictionary and list. Step 4 was where I chose to place the code I knew would work, and as I have been doing all along, I added print statements at every turn to make sure at least nothing was too broken to stop the program from running.

From there I jumped to Step 6, which was similar enough to the previous Inventory Helper that I felt confident here again. The one mistake I made was to assume I would be appending data to the document, where in order for items to be added *and* removed, it would be overwriting. Had I realized that right away, and I cannot but wonder why it skipped my mind, my subsequent troubles would not have taken shape.

Once I had written to the document, or at least appended tasks to it, I looked back at Step 1. I opened the textbook to Chapter 7 again, and could not find what I was seeking. Specifically, I found nothing in the assignment documents, nor the video, describing how exactly to pull lines from the text document back into the dictionary. I must have missed it, or had some other gap in my understanding, because I tried and tried and only got errors. And when I took to Google for answers, I mostly encountered with ... as statements that, while making enough sense to be copied over, were outside of what we have learned so far. Abandoning those, I realized part of my issue was how I had gone about Step 6, and that by opening and reading the document it would be necessary to overwrite when saving.

That revelation led me promptly to Step 5, where I was able to come up with a way to delete the internal list items I had added, without worrying about the reading and writing to the document. For a moment, the program hitched here, and in my frustration with earlier attempts I had forgotten to ensure I was using string formatting such as title and lower. Once those were fixed, and this part of the program was operating correctly, I looked at how I would store the values for Priority within the dictionary. I chose to leave them as strings, regardless of my settling on simple integer digits. Were I needing to sort by urgency or date or some other calculable metric, I would have added int to my inputs.

Finally the reading from the document and putting those values into the dictionary clicked. And I phrase it that way because it appears I put the horse before the cart and kept adding the Task column item to the key to the Priority column's value, so I ended up with dicRow = {taskName:1} and not {"Task":taskName} and {"Priority":taskPriority} as intended. Such turmoil, and I did it to myself. Good grief.

# The Code, It Works

I was ecstatic when the program performed as intended. With breaks for food and a cursory glance at GitHub, I took my screenshots and prepared to turn everything in. A couple of minor alterations after running the code and reviewing the captures and I ended up with every menu option executing as intended (Figure 1). After verifying that the text document updated accordingly, I ran the program through the command prompt, exploring again all of the options to affirm their correct operation (Figure 2). In the end, I present the list of four six-letter-so-that-I-maintain-some-semblance-of-elegance tasks, and their assigned priorities (Figure 3).

On the following pages is my code, copied from PyCharm.

#### # ------------------------------------------------------------------------ #

#### # 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,1.1.2020,Created started script

#### # erikjk,5.14.2020, Created text file and populated it with tasks

#### # erikjk,5.14.2020, Worked through Step 4 - Adding to the document

#### # erikjk,5.14.2020, Worked through Step 6 - Appending values to document

#### # erikjk,5.14.2020, Worked through Step 1 - Failed and worked on Step 3

#### # erikjk,5.14.2020, Realized I had done Step 6 wrong - Overwrite after read, not append

#### # erikjk,5.14.2020, Figured out Step 1 - Worked on formatting

#### # erikjk,5.14.2020, Worked through Step 5 - Deleting dictionary entries based on keys

#### # erikjk,5.14.2020, Added timer in Step 7 after printing goodbye message

#### # erikjk,5.14.2020, Organized Data section, variables, etc.

#### # ------------------------------------------------------------------------ #

#### # -- Data -- #

#### # declare variables and constants

#### objFile = "ToDoList.txt" # An object that represents a file

#### todoTxt = "" # For opening and closing the target document

#### 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 of the user's option selection

#### taskName = "" # A capture of the user's preferred task name

#### taskPriority = "" # A capture of the user's priority for the task

#### myTask = "" # A variable used throughout to reference dictionary values within the list

#### # -- 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)

#### import time # Using timer in place of "enter to exit" to avoid double input

#### todoTxt = open(objFile, "r")

#### for line in todoTxt:

#### strData = line.split(" | ")

#### dicRow = {"Task":strData[0].title(), "Priority":strData[1].strip()}

#### lstTable.append(dicRow)

#### todoTxt.close()

#### # -- Input/Output -- #

#### # Step 2 - Display a menu of choices to the user

#### print("Welcome to Task Manager!\n")

#### 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 5] - "))

#### print() # adding a new line for looks

#### # Step 3 - Show the current items in the table

#### if (strChoice.strip() == '1'):

#### print("Task | Priority")

#### for myTask in lstTable:

#### print(myTask["Task"], myTask["Priority"], sep=" | ")

#### continue

#### # Step 4 - Add a new item to the list/Table

#### elif (strChoice.strip() == '2'):

#### taskName = input("Task Name: ").title()

#### taskPriority = input("Task Priority: ")

#### dicRow = {"Task": taskName, "Priority": taskPriority}

#### lstTable.append(dicRow)

#### continue

#### # Step 5 - Remove a new item from the list/Table

#### elif (strChoice.strip() == '3'):

#### myTask = input("Which Task do you wish to remove: ")

#### for dicRow in lstTable:

#### if dicRow["Task"].lower() == myTask:

#### lstTable.remove(dicRow)

#### print("The requested Task has been deleted.")

#### continue

#### # Step 6 - Save tasks to the ToDoToDoList.txt file

#### elif (strChoice.strip() == '4'):

#### todoTxt = open(objFile, "w")

#### for myTask in lstTable:

#### todoTxt.write(myTask["Task"] + " | " + myTask["Priority"] + "\n")

#### todoTxt.close()

#### print("Your Tasks document has been successfully updated.")

#### continue

#### # Step 7 - Exit program

#### elif (strChoice.strip() == '5'):

#### print("Thank you for using Task Manager!")

#### time.sleep(3)

#### break # and Exit the program

### 

### Figure 1: Showcase of Every Choice Operating Correctly in PyCharm

### 

### Figure 2 : Showcase of Every Choice Operating Correctly in Command Prompt

### 

### Figure 3: Text Document After Successful Runs in PyCharm and Command Prompt

GitHub is a community I had intended on joining. I am writing, at this point only in pseudo-code, a way to roll for ability scores in Dungeons and Dragons to my DM's unique methodology. Not that this will be the same account I use for personal projects, but I am glad for the introduction.

Getting my account set up was a breeze, as was making my first repository. All according to plan. Now I find myself in the position where I need to finish this before I can upload it, but need to upload it before I can finish it, but need to finish before I can upload it, and upload when it's done.

# Summary

I said in the introduction that this assignment was the toughest yet, and that is to be expected, I suppose. Still, between tripping over the code I inherited, messing up my approach to the document writing, how that snowballed into my failing to read the document, which avalanched into an inability to progress with building the dictionaries within the list that was so simple in overview. Being that I am to blame for my own failures, I hope in struggling to overcome my own mental blocks that next week's assignment will be smoother. Good practices, better habits, and I appreciate the challenge as always.