Kevin Scales

Feb 15, 2021

Introduction to Programming (Python)

Assignment 5

To Do List

# Introduction

This assignment introduced working with dictionaries, combining them with lists to add, delete, list, and load and save dictionary items of tasks and priorities. The data is string, with no check made that the priorities are in any particular format (1 to 10, high or low, etc.) using now-familiar loops and conditionals.

# The Python Script

Using a combination of lists and dictionaries brought out the differences, though it complicated the syntax and left some of the book examples not helpful. The initial starter document, besides helping define properly some of the variables, also emphasized the separation of concerns by partitioning the code into exclusive blocks that dealt with each function that could apply to the common set of changing data.

The first part of the code read in the file, if one exists. Here the try-except came in handy, allowing us to start with no data if the file was missing without trying to open a non-existing file. This is not a serious problem, and simply moving on without the file is easily done with the try method.

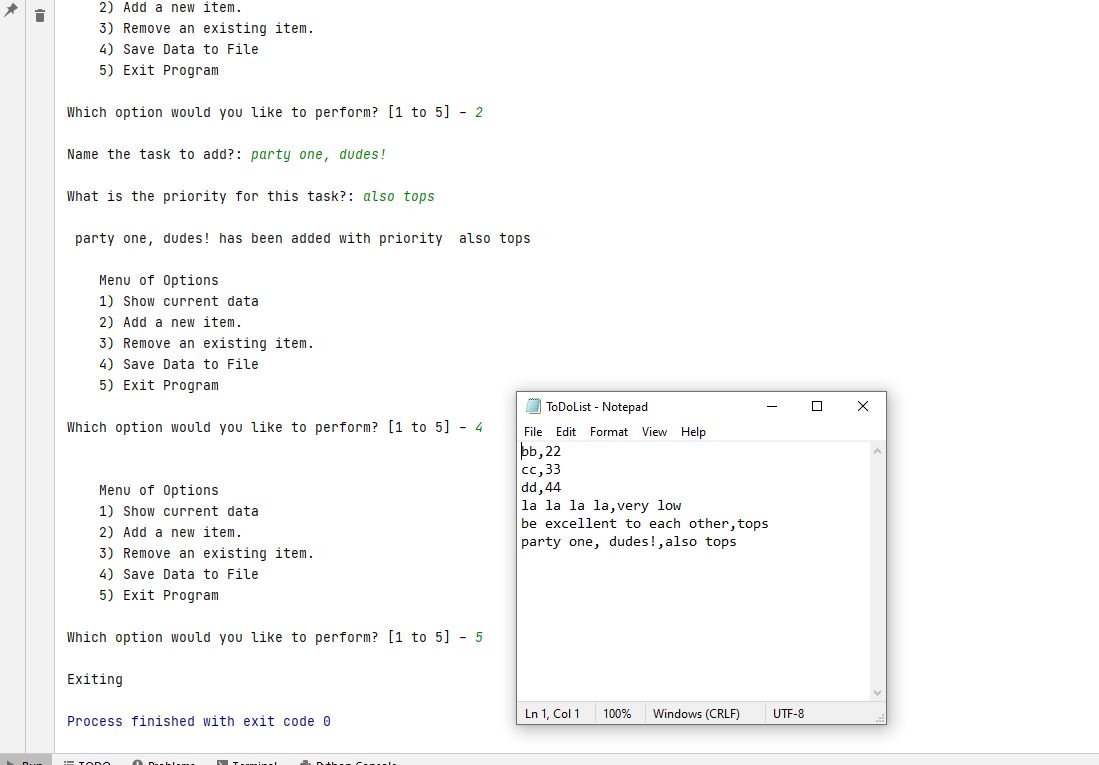
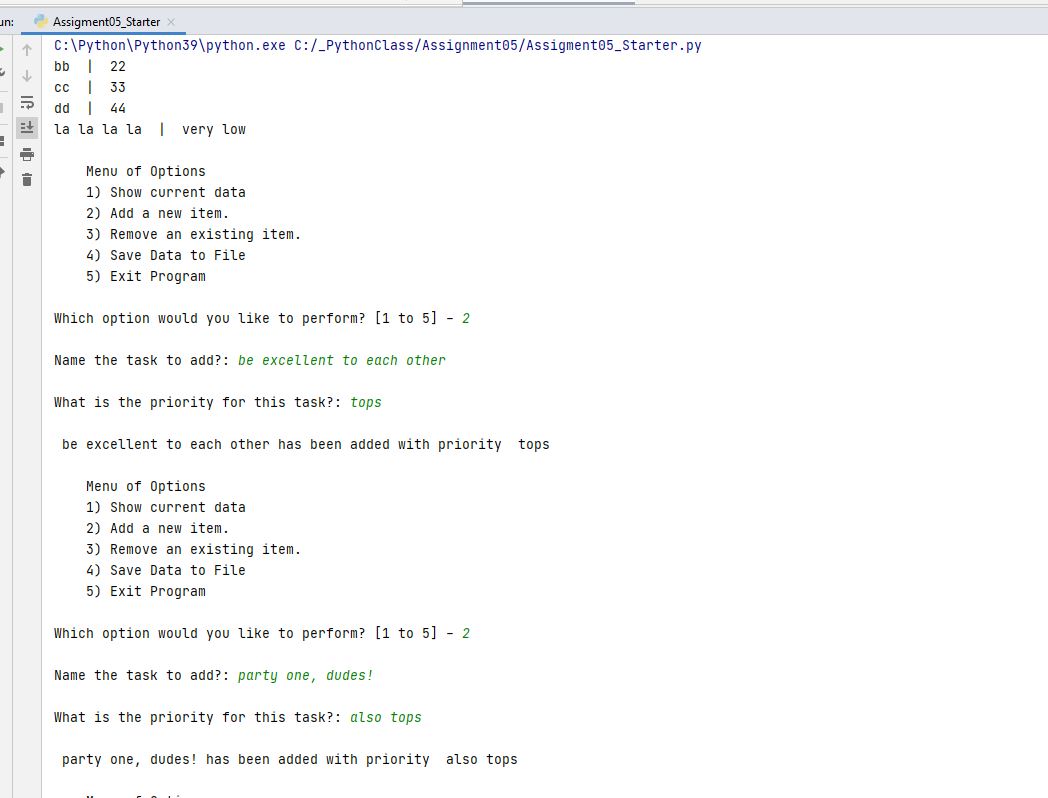
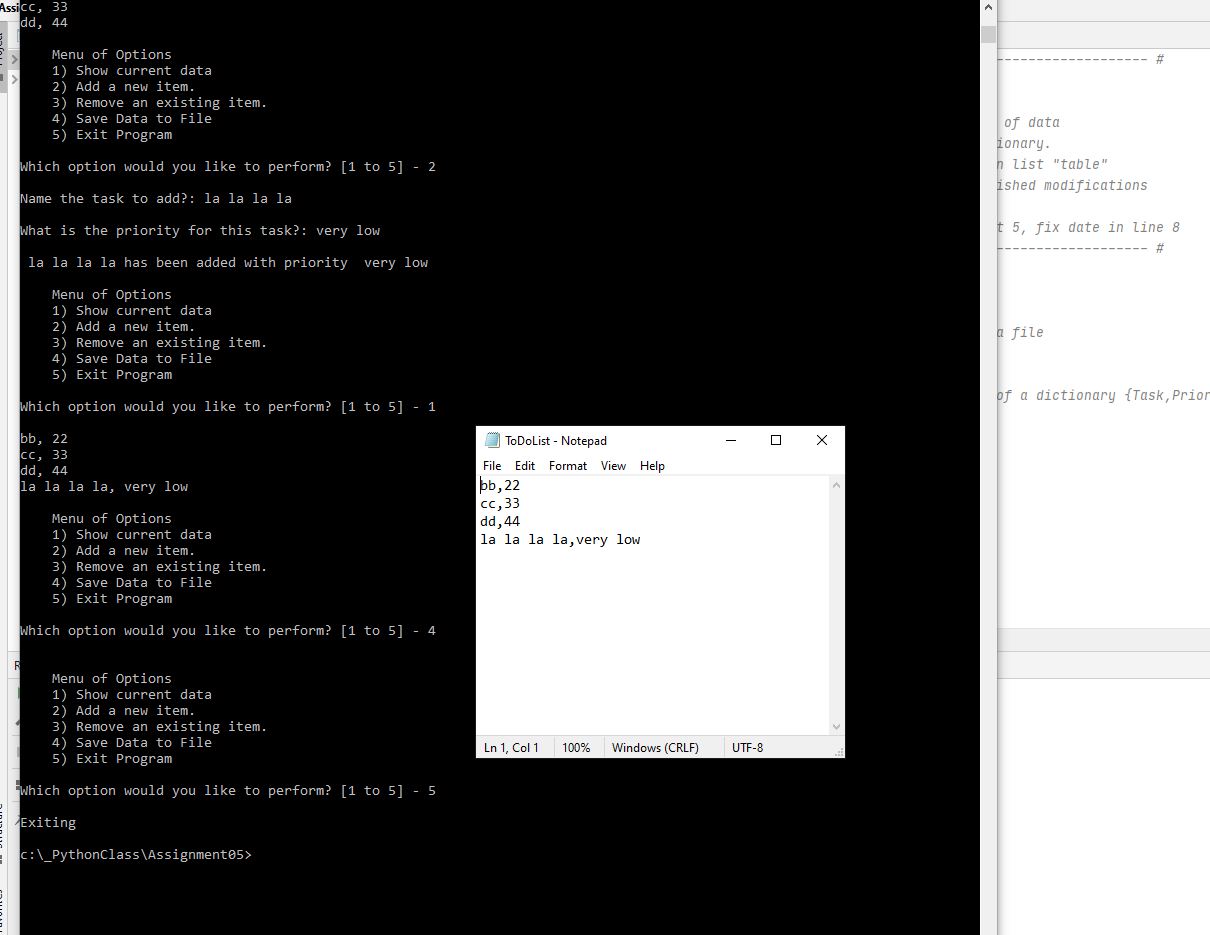
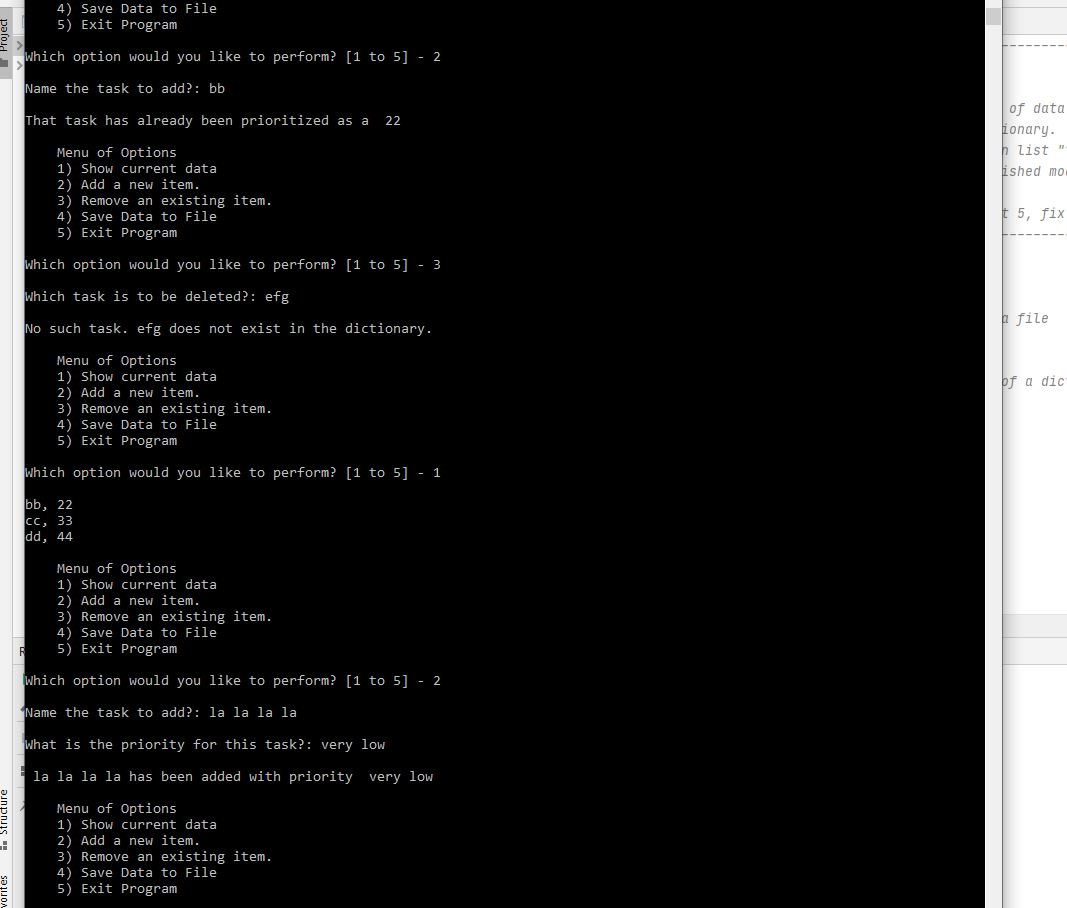
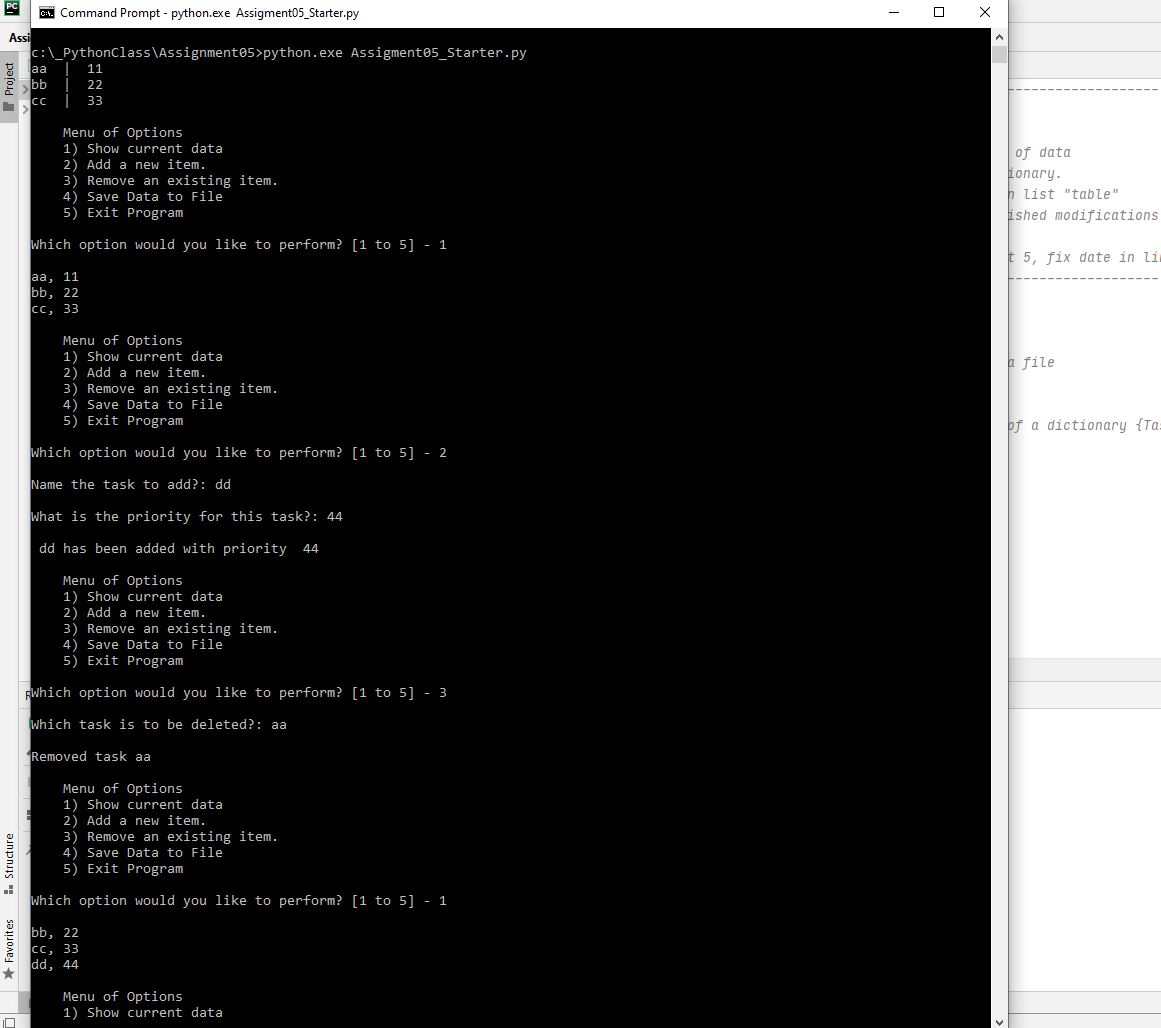
The overall assignment’s main trickiness was in separating list from dictionary syntax. At various points during development, all of my sections except 5 – exiting the program – had to be re-done after seemingly nearing completion only to find that some section didn’t quite blend well with the others. For example, I worked on reading in the data from a file last (just typing in fresh data for a while), only to find that I was not really working with dictionaries per se for most of it and all subsequent portions had to be re-done. I was essentially trying to work just a single line of a dictionary, not a list of them.

# The code and output

The code is given below:

*# ------------------------------------------------------------------------ #  
# 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): Kevin Scales, 2/15/21, finished modifications  
# RRoot,1.1.2020,Created started script  
# Kevin Scales,2.15.21,Added code to complete assignment 5, fix date in line 8  
# ------------------------------------------------------------------------ #  
  
# -- Data -- #  
# declare variables and constants*strFile = **"ToDoList.txt"** *# An object that represents a file*strData = **""** *# A row of text data from the file  
#objFile = None*dicRow = {} *# A row of data separated into elements of a dictionary {Task,Priority}*lstTable = [] *# A list that acts as a 'table' of rows*strChoice = **""** *# A Capture the user option selection*discon = 0  
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  
"""** *# A menu of user options  
  
# -- 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)  
#* ***TODO: Add Code Here, Done*try**:  
  
 objFile = open(strFile, **"r"**)  
 **for** row **in** objFile:  
 lstRow = row.split(**","**)  
 dicRow = {**"task"**: lstRow[0], **"priority"**: lstRow[1].strip()}  
 print(dicRow[**"task"**], **" | "**, dicRow[**"priority"**])  
 lstTable += [dicRow]  
  
 objFile.close()  
  
**except**:  
 print(**"File does not exist, no data loaded.\n"**)  
  
  
*# -- 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'**):  
 *#* ***TODO: Add Code Here****# for task, priority in dicRow.items():  
# print(task, ", ", priority)  
# print(lstTable, '\n')* **for** objRow **in** lstTable:  
 print(objRow[**"task"**], **'\b,'**,objRow[**"priority"**])  
  
 **continue** *# Step 4 - Add a new item to the list/Table* **elif** (strChoice.strip() == **'2'**):  
 *#* ***TODO: Add Code Here*** discon = 0  
 task = input(**"Name the task to add?: "**)  
 **for** objRow **in** lstTable:  
 **if** task == objRow[**"task"**]:  
 print(**"\nThat task has already been prioritized as a "**, objRow[**"priority"**])  
 discon = 1  
 **continue  
 if** discon != 1:  
 priority = input(**"\nWhat is the priority for this task?: "**)  
 dicRow = {**"task"**: task, **"priority"**: priority}  
 lstTable += [dicRow]  
 print(**"\n"**, task, **"has been added with priority "**, priority)  
 discon = 0  
  
 **continue** *# Step 5 - Remove an item from the list/Table based on its name* **elif** (strChoice.strip() == **'3'**):  
 *#* ***TODO: Add Code Here*** discon = 0  
 task = input(**"Which task is to be deleted?: "**)  
 i=0 *# counter for list position* **for** objRow **in** lstTable:  
 **if** task == objRow[**"task"**]:  
 **del** lstTable[i]  
 print(**"\nRemoved task"**, task)  
 discon = 1  
 **continue** i += 1  
  
 **if** discon != 1:  
 print(**"\nNo such task."**, task, **"does not exist in the dictionary."**)  
 discon = 0  
  
 **continue** *# Step 6 - Save tasks to the ToDoToDoList.txt file* **elif** (strChoice.strip() == **'4'**):  
 *#* ***TODO: Add Code Here*** objFile = open(strFile, **"w"**)  
 **for** objRow **in** lstTable:  
 objFile.write(objRow[**"task"**] + **','** + objRow[**"priority"**] + **'\n'**)  
 objFile.close()  
  
 **continue** *# Step 7 - Exit program* **elif** (strChoice.strip() == **'5'**):  
 *#* ***TODO: Add Code Here*** print(**"Exiting"**)  
 **break** *# and Exit the program*

Graphical output for the console window and the PyCharm IDE is found below, first for the console and then for the IDE:



# Conclusion

The list and dictionary types make up the predominant pedagogical topics for this assignment in terms of syntax. Try-Except is introduced in a simple manner. The principle of separations of concerns is formally defined, but speaks to concepts that we have already looked at in designing software based on inputs from menu-based interfaces.