Kegan Sanchez

November 16, 2020

Foundations of Programming (Python)

Assignment 05

The To Do List Program

# Introduction

This week, I modified a Python script that manages a “To Do list” which contains two columns of data, “Task” and “Priority,” and loads the columns into a Python dictionary object. Each dictionary object represents one row of data, and the rows were added to a Python list object to create a table of data. I added functionality to process the data based on the user’s choice. This program incorporates new components including working with dictionaries, deleting and appending to a list, and more complex file writing processes.

# Creating the Program

I began my program by creating a new project in PyCharm and adding the starter file to my project. I updated the changelog in the script’s header and started adding code to perform the assignment’s tasks.

## Loading Data from a Text File

My first task was to ensure that once the program starts, any data stored in a text file named ToDoList.txt was loaded into a list of dictionaries rows. I began by creating a text file, writing a couple pieces of data in it, and opening the text file using variables that were initialized in the beginning of the script. I then processed the data by splitting and stripping each row and added each processed row to the list table before closing the file (Figure 1).

strFile = "ToDoList.txt" # A data storage file

objFile = None # An object that represents a file

-------------------------------------------------

objFile = open(strFile, "r")

for row in objFile:

strData = row.split(",")

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

lstTable.append(dicRow)

objFile.close()

***Figure 1: Loading Data***

## Displaying the Current Data

Once the data was loaded into the list table, I moved on to the first menu option. If the user chose option 1, the current data would need to be displayed. I decided to show the user a header with the “Task” and “Priority” titles, followed by a dotted line divider, and then each task and priority pair on separate lines below it (Figure 2).

print("Task\tPriority")

print("---------------")

for row in lstTable:

print(row["Task"] + "\t\t" + row["Priority"])

***Figure 2: Displaying Data***

## Adding a New Item

Moving on, if the user chose option 2, they would be prompted for the name of a new task and its priority. I stored the user’s responses in two separate variables which I then inserted into a dictionary row. I appended this row to the current list table and displayed an “Item added.” message to the user (Figure 3).

strTask = str(input("Task: "))

strPriority = str(input("Priority: "))

dicRow = {"Task":strTask.lower(), "Priority":strPriority}

lstTable.append(dicRow)

print("Item added.")

***Figure 3: Adding Data***

## Removing an Item

If the user chose option 3, they could pick an item to remove. I retrieved the name of the task the user wanted to delete, stored it in a variable, and then searched the table for the matching task by comparing each item to this variable. If the task was found, it was deleted based on its position within the list table. I had assigned a 0 to a variable named taskFound, which was then updated to a 1 if the task was found, and I used this variable as a condition to send the user a message based on the result of the search. (Figure 4).

strTask = str(input("Which task would you like to delete? "))

taskFound = 0

for i in range(len(lstTable)):

if lstTable[i]["Task"] == strTask.lower():

del lstTable[i]

taskFound = 1

break

if taskFound:

print("Item deleted.")

else:

print("Item not found.")

***Figure 4: Removing Data***

## Writing and Saving the Data to a Text File

Besides exiting the program, the last option for the user was to save the current tasks to the ToDoList.txt file. If the user chose option 4, I opened and/or created a text file and wrote each row to the file with a comma delimiter and a new line. I then closed the file and sent a status message to the user (Figure 5).

objFile = open(strFile, "w")

for row in lstTable:

objFile.write(row["Task"] + "," + row["Priority"] + "\n")

objFile.close()

print("File saved.")

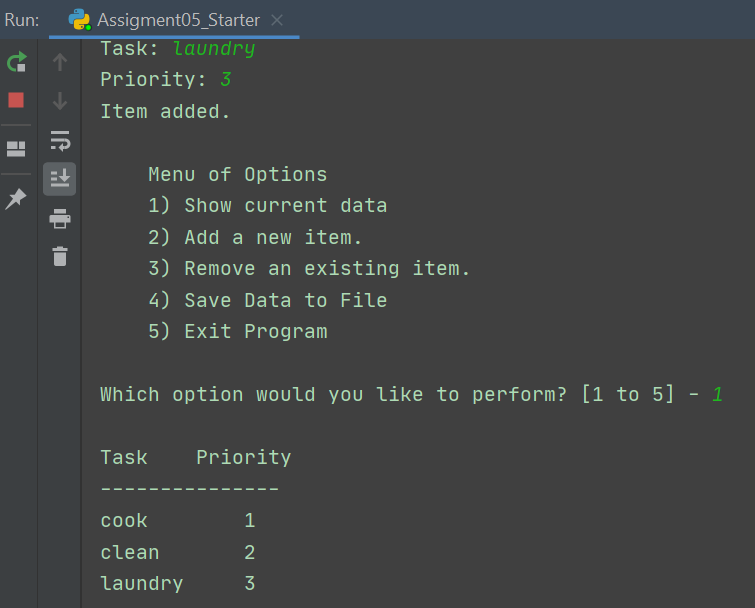
***Figure 5: Saving Data***

# Testing

When I began creating this program, I completed all coding and testing primarily in the PyCharm IDE. As part of this assignment, I also tested it in the command terminal.

## PyCharm

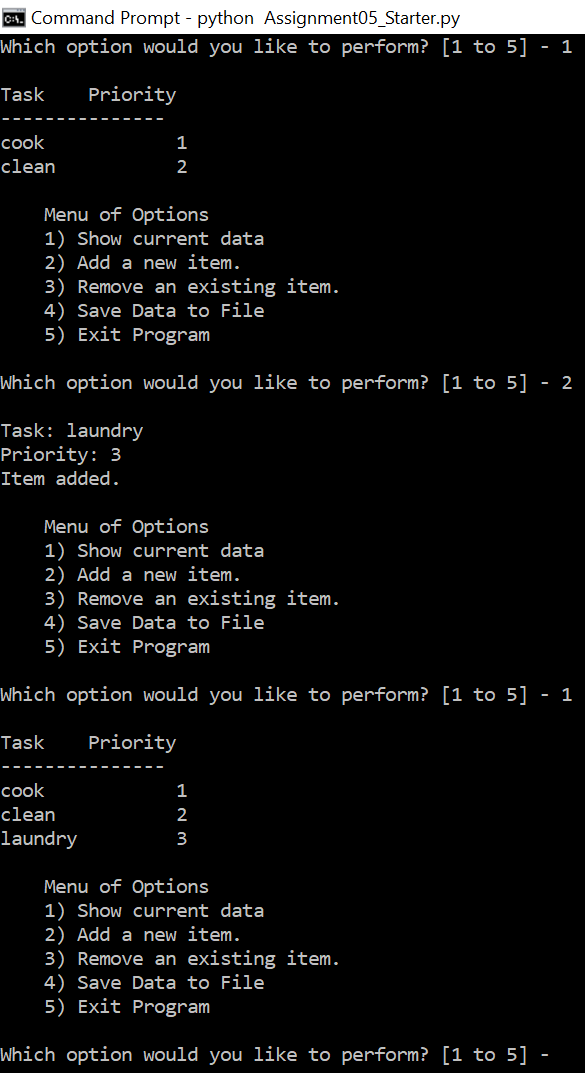
I first ran my To Do List program in PyCharm and received the desired output, including showing the current data, adding a new item, removing an item, and saving the data to a text file (Figure 6).



***Figure 6: PyCharm Testing***

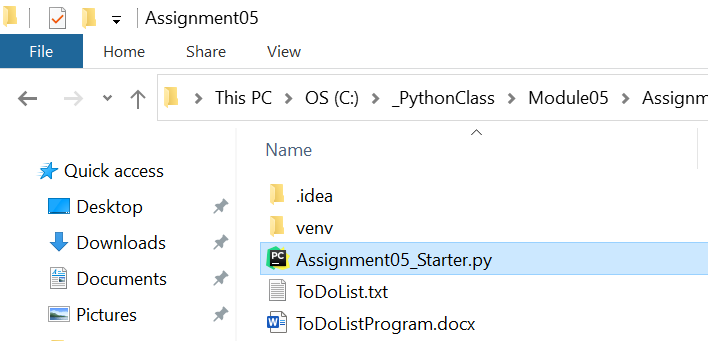
## Command Terminal

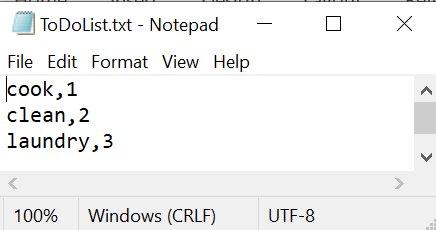
I also opened a command terminal, changed the directory to my assignment folder, and ran my Assignment05\_Starter.py script file. I received the desired output here, as well (Figure 7).



***Figure 7: Command Terminal Testing***

Finally, I checked that the text document was saved in my assignment folder, opened it, and confirmed that the data I entered was written into the file correctly (Figure 8).





***Figure 8: Verifying Data Saved to Text File***

# Summary

After completing the code for each menu option and seeing my code run successfully, I reflected on what I learned and what I could have done differently. I not only learned about how to create and use a list table with dictionary objects, but also how to delete and append to such a complex list structure, and then process the data and write it to a file.

There are many other ways that I could have created this program, including the use of custom functions and different types of ways to display the data and write it to a file. I also ran into a couple of cases that were not required for this course, but had me wondering how to address them in future programs. One such case was handling an empty file and another was an extra tab or spaces appearing after task names of a certain length. I will keep these in mind for future programs and look for ways to improve and account for them.