Jessica Trinh
May 15th 2020
IT FDN 100 A
Assignment 05 Documentation
https://github.com/trinh-j/IntroToProg_Python

Writing a Basic Program using Collections of Data

Introduction

For this assignment, we will be writing a basic program that gives the end-user options to write, display, and/or save data to a file. This program will be very similar to Assignment 04, but will include the use of different (and more) types of data, i.e. dictionaries, lists/tables, functions such as .split() and .strip(), to name a few. To begin, a starter .py file has been provided to us by professor Randal Root, to which we will add code in the appropriate sections. In addition to learning how to use various collections of data, the objective of this project is to learn how to modify code from another developer. This will be a helpful exercise for future projects that require collaboration on a single coding project.

Collections

Before we start manipulating our starter file, we will discuss the new types of data being introduced in this assignment.

Data is often grouped into a collection. There are several types of collections in python, such as dictionaries, lists, tuples, and sets.

<u>Dictionaries</u>: Uses curly braces; a single element is a key-value assignment; i.e. dict = {key1: value1, key2, value2, etc.}. Think of the keys as columns and the assigned values as a row.

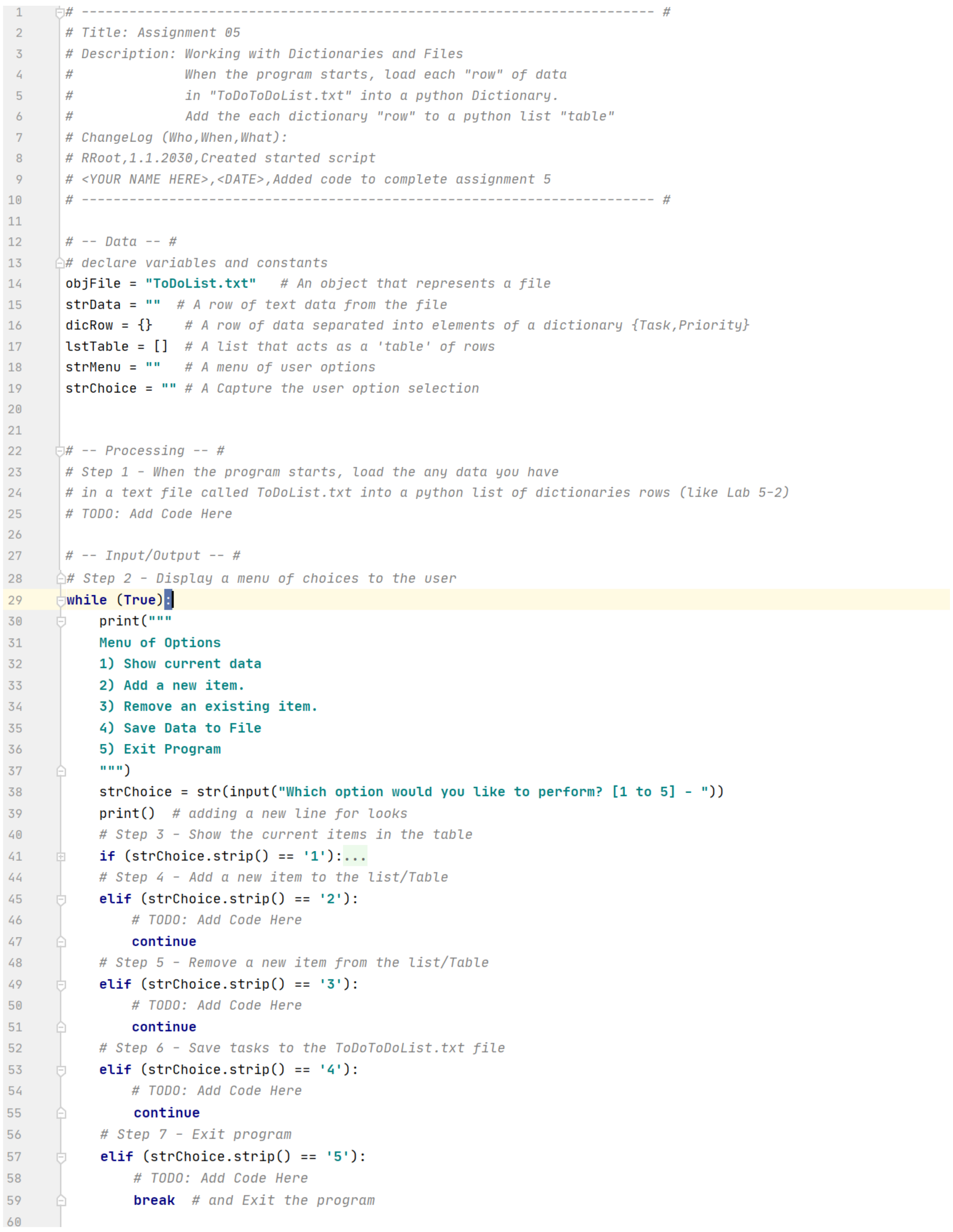
<u>Lists</u>: Uses brackets and can contain values of multiple types. E.g. list = [‘a’, 5, 10]; mutable collections

<u>Tuples</u>: Like lists but use parentheses. Object is immutable, meaning elements are preserved. E.g. tuple = (2, 1, 3, 5).

<u>Sets</u>: Uses curly braces; removes duplicate elements; sorts data e.g. set = {2,3,1,5} → {1,2,3,5}

Writing the Script

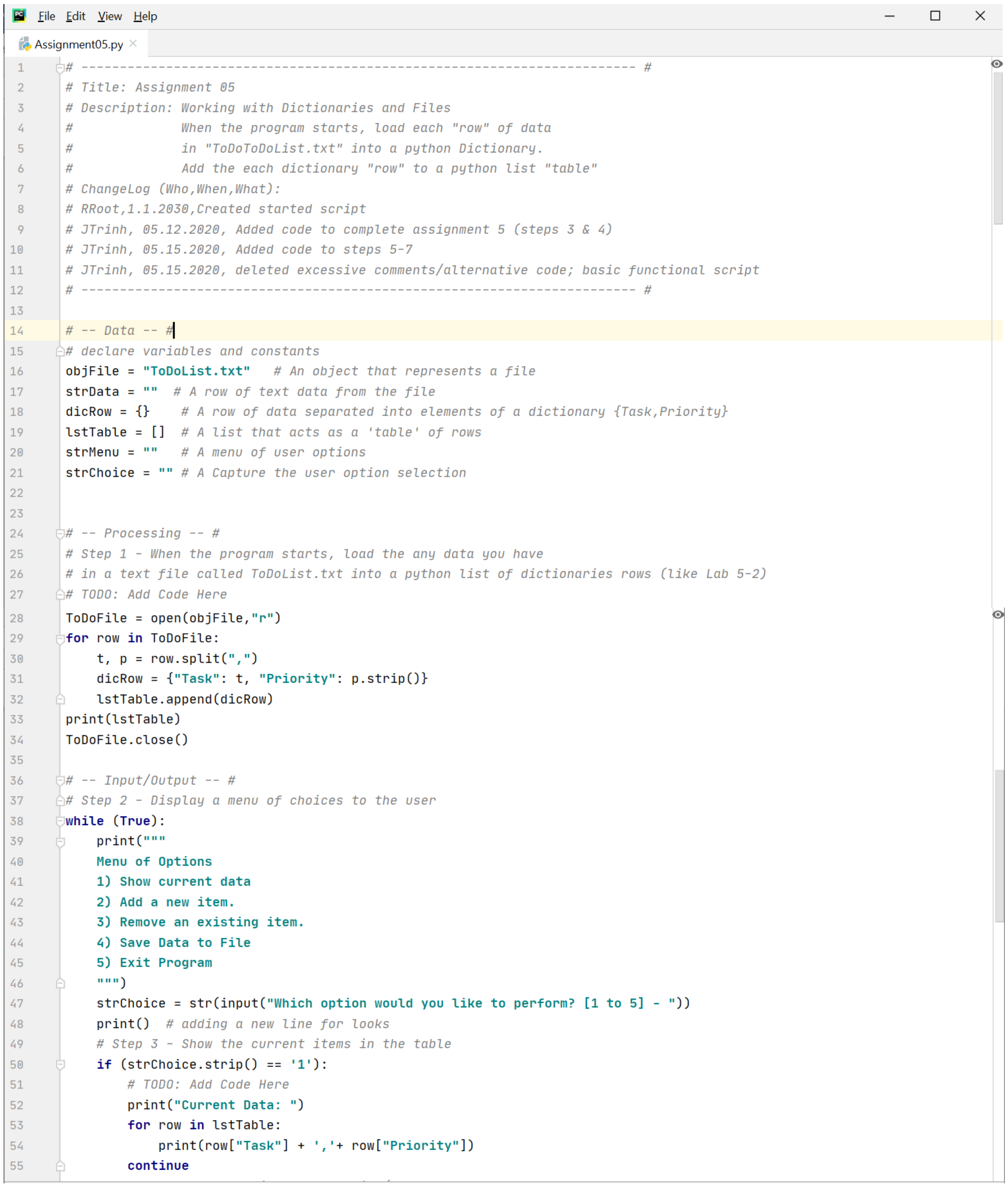
In the starter code, we see that the types of data collection used in our assignment are dictionaries and lists, as seen in the “Data” section (Figure 1). Professor root has sectioned off the while loop into steps 3 through; each section will require me, along with my classmates, to modify the script by adding code below the “#TODO: Add Code Here” lines.



```python
# ------------------------------------------------------------------------- #
# 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
# <YOUR NAME HERE>,<DATE>,Added code to complete assignment 5
# ------------------------------------------------------------------------- #

# -- Data -- #
# declare variables and constants
objFile = "ToDoList.txt"   # An object that represents a file
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 the user option selection


# -- 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

# -- Input/Output -- #
# Step 2 - Display a menu of choices to the user
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'):...
    # Step 4 - Add a new item to the list/Table
    elif (strChoice.strip() == '2'):
        # TODO: Add Code Here
        continue
    # Step 5 - Remove a new item from the list/Table
    elif (strChoice.strip() == '3'):
        # TODO: Add Code Here
        continue
    # Step 6 - Save tasks to the ToDoToDoList.txt file
    elif (strChoice.strip() == '4'):
        # TODO: Add Code Here
        continue
    # Step 7 - Exit program
    elif (strChoice.strip() == '5'):
        # TODO: Add Code Here
        break  # and Exit the program

```

Figure 1. Starter Script from Professor Randal Root.

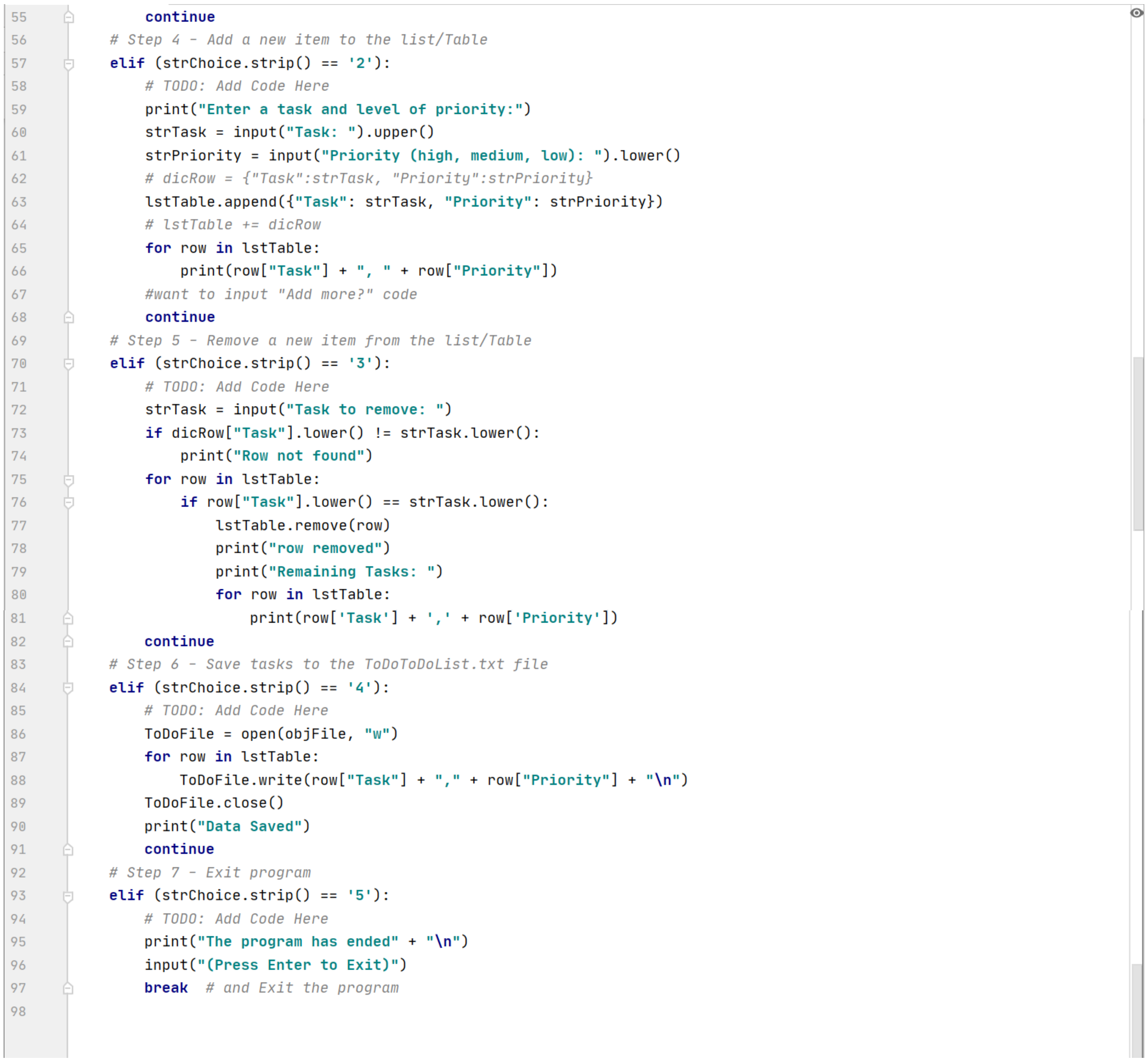
I began writing my code under the processing section (Figure 2, lines 28-34). I created a variable "ToDoFile" to open and access the text file assigned to variable "objFile." The goal of this section is to read out any rows of data in the text file and properly reformat the values into a dictionary, and then append each dictionary to a table, so that it can be read out in the program. Using resources and lecture notes, I create a for-loop that allowed me to iterate through my text file, unpackage my values into variables "t" and "p" (for "task" and "priority", respectively; assign "t" and "p" to keys "Task" and "Priority", and then add each dictionary as a row to the table "lstTable." Once all the dictionary-rows are appended to the table, the table is printed out to show the initial contents of the text file before the program begins, and the "ToDoFile" text file is closed.



```
1   # ---------------------------------------------------------------------- #
2   # Title: Assignment 05
3   # Description: Working with Dictionaries and Files
4   #              When the program starts, load each "row" of data
5   #              in "ToDoToDoList.txt" into a python Dictionary.
6   #              Add the each dictionary "row" to a python list "table"
7   # ChangeLog (Who,When,What):
8   # RRoot,1.1.2030,Created started script
9   # JTrinh, 05.12.2020, Added code to complete assignment 5 (steps 3 & 4)
10  # JTrinh, 05.15.2020, Added code to steps 5-7
11  # JTrinh, 05.15.2020, deleted excessive comments/alternative code; basic functional script
12  # ---------------------------------------------------------------------- #
13
14  # -- Data -- #
15  # declare variables and constants
16  objFile = "ToDoList.txt"   # An object that represents a file
17  strData = ""  # A row of text data from the file
18  dicRow = {}    # A row of data separated into elements of a dictionary {Task,Priority}
19  lstTable = []  # A list that acts as a 'table' of rows
20  strMenu = ""   # A menu of user options
21  strChoice = "" # A Capture the user option selection
22
23
24  # -- Processing -- #
25  # Step 1 - When the program starts, load the any data you have
26  # in a text file called ToDoList.txt into a python list of dictionaries rows (like Lab 5-2)
27  # TODO: Add Code Here
28  ToDoFile = open(objFile,"r")
29  for row in ToDoFile:
30      t, p = row.split(",")
31      dicRow = {"Task": t, "Priority": p.strip()}
32      lstTable.append(dicRow)
33  print(lstTable)
34  ToDoFile.close()
35
36  # -- Input/Output -- #
37  # Step 2 - Display a menu of choices to the user
38  while (True):
39      print("""
40      Menu of Options
41      1) Show current data
42      2) Add a new item.
43      3) Remove an existing item.
44      4) Save Data to File
45      5) Exit Program
46      """)
47      strChoice = str(input("Which option would you like to perform? [1 to 5] - "))
48      print()  # adding a new line for looks
49      # Step 3 - Show the current items in the table
50      if (strChoice.strip() == '1'):
51          # TODO: Add Code Here
52          print("Current Data: ")
53          for row in lstTable:
54              print(row["Task"] + ','+ row["Priority"])
55          continue
```

Figure 2. (Above) Added code starter script, changed file name form “Assigment05_Starter.py” to “Assignment05.py”; script shown (above) from beginning to line 55.

After writing code for the processing section, I started adding code within the while loop, starting at line 52 (Figure 2). Here, if the end-user’s input, based on menu options, is “1” I want the rows in “lstTable” to be displayed without brackets or curly braces. Using a for-loop, I iterated through “lstTable” and printed the values associated with “Task” and “Priority” keys, in a comma separated format. Once the program gets through all the rows in the table, it continues to the beginning of the while-loop, where the user is given the menu and asked for another input based on the menu.



```
55          continue
56      # Step 4 - Add a new item to the list/Table
57      elif (strChoice.strip() == '2'):
58          # TODO: Add Code Here
59          print("Enter a task and level of priority:")
60          strTask = input("Task: ").upper()
61          strPriority = input("Priority (high, medium, low): ").lower()
62          # dicRow = {"Task":strTask, "Priority":strPriority}
63          lstTable.append({"Task": strTask, "Priority": strPriority})
64          # lstTable += dicRow
65          for row in lstTable:
66              print(row["Task"] + ", " + row["Priority"])
67          #want to input "Add more?" code
68          continue
69      # Step 5 - Remove a new item from the list/Table
70      elif (strChoice.strip() == '3'):
71          # TODO: Add Code Here
72          strTask = input("Task to remove: ")
73          if dicRow["Task"].lower() != strTask.lower():
74              print("Row not found")
75          for row in lstTable:
76              if row["Task"].lower() == strTask.lower():
77                  lstTable.remove(row)
78                  print("row removed")
79                  print("Remaining Tasks: ")
80                  for row in lstTable:
81                      print(row['Task'] + ',' + row['Priority'])
82          continue
83      # Step 6 - Save tasks to the ToDoToDoList.txt file
84      elif (strChoice.strip() == '4'):
85          # TODO: Add Code Here
86          ToDoFile = open(objFile, "w")
87          for row in lstTable:
88              ToDoFile.write(row["Task"] + "," + row["Priority"] + "\n")
89          ToDoFile.close()
90          print("Data Saved")
91          continue
92      # Step 7 - Exit program
93      elif (strChoice.strip() == '5'):
94          # TODO: Add Code Here
95          print("The program has ended" + "\n")
96          input("(Press Enter to Exit)")
97          break  # and Exit the program
98
```

Figure 2. (Above) (continued) script from line 55 to end

Next, I wrote code for steps 4 through 7, allowing the user to add tasks, remove tasks, save tasks, and exit the program, respectively. To give a more detailed break-down:

Step 4: If user input is “2,” then add following user inputs associated with task and priority into a dictionary, then append the dictionary to the table. I tried to do this multiple ways with declaring a separate variable for my dictionary, but experienced some errors so I reverted to the code written in its current state.

Step 5: If user input is “3,” then user wants to remove a task from the list. If their task, in lowercase, does not match an existing task, also in lowercase, in lstTable, then notify the user that the task does not exist. If the input matches any task found in the table, remove the task and let the user know that it has been removed and what the current list is after row-removal.

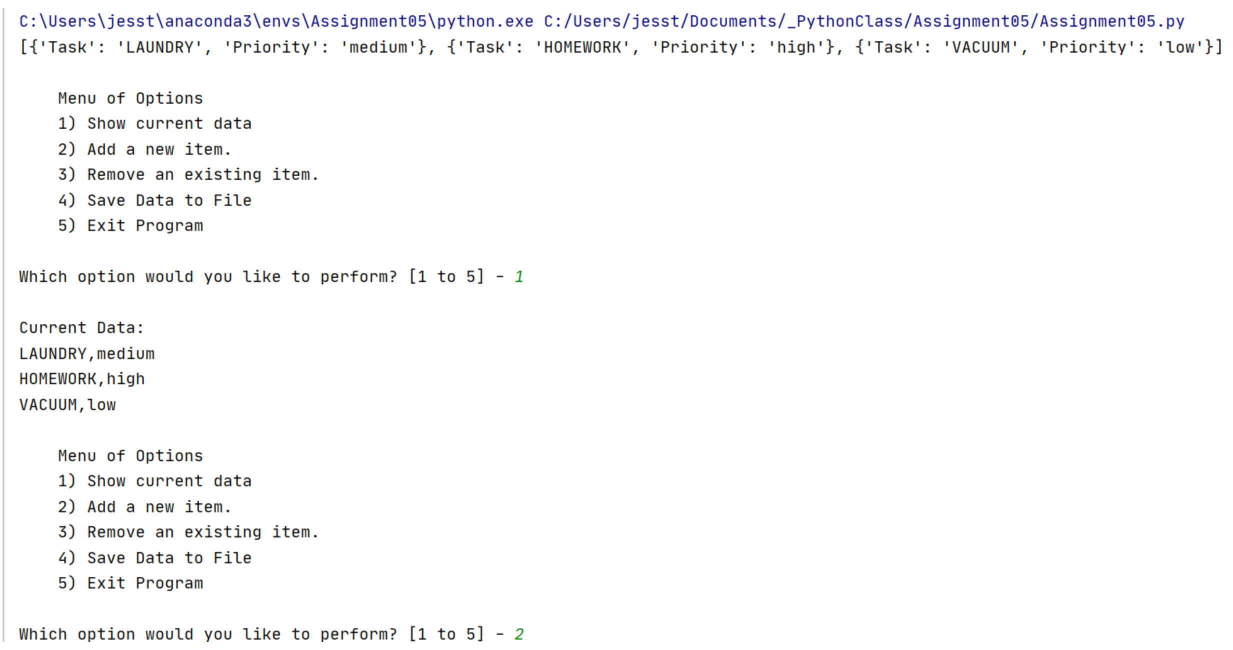
Step 6: If user input is “4,” then the user wants to save their data in memory queue. Here, I wrote code to open the text file, format and write all dictionary elements to the text file, then close the text file. The user is also notified that their data has been saved.

Step 7: Finally, if the user input is “5,” then the user wants to end the program. The program ends with a print statement and simple input function and breaks from the while loop.

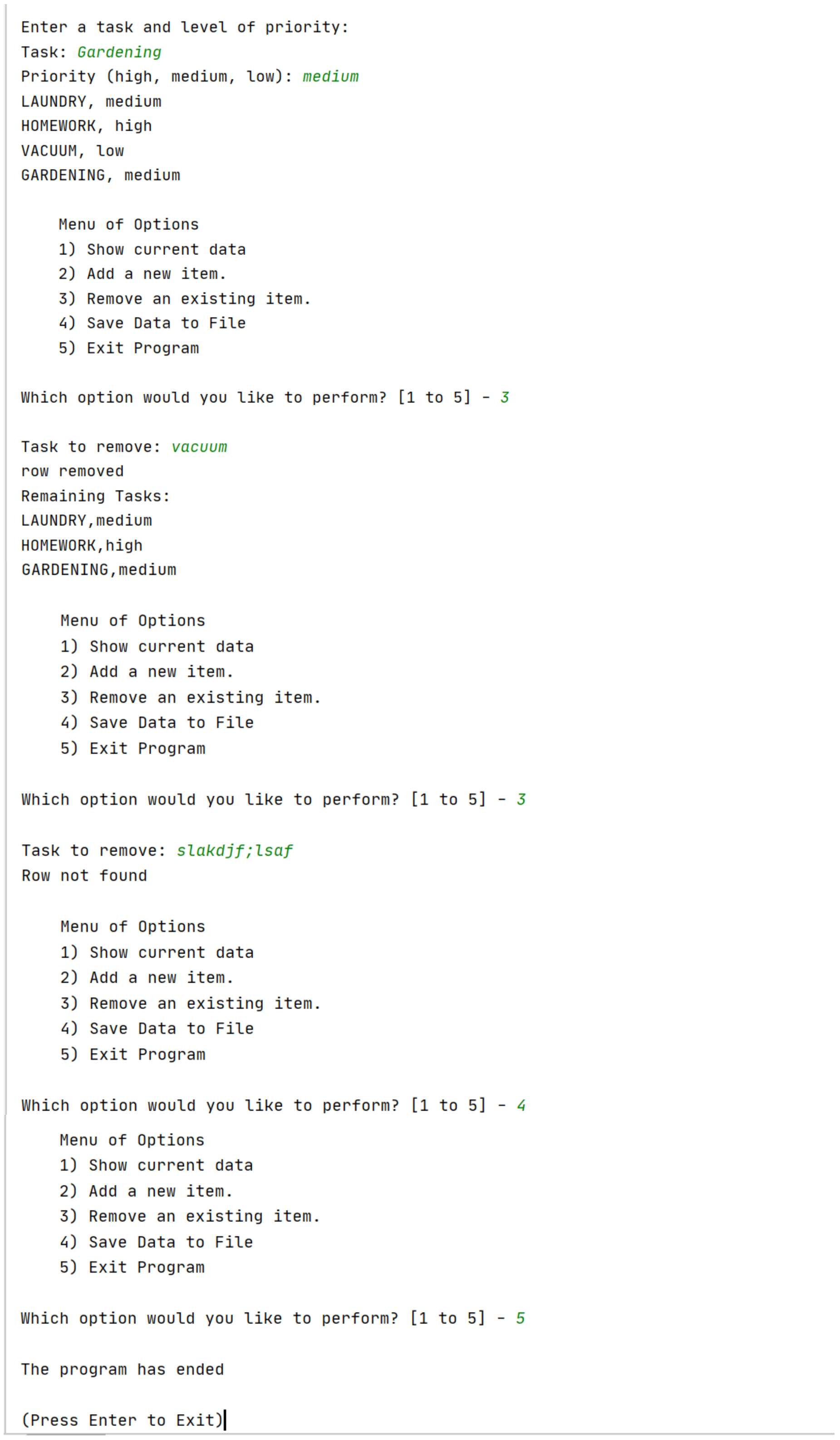
Output

To ensure that the program is functional, I ran the script many times. Below is a screenshot of the program ran with all possible inputs the user is allowed to choose from the Menu of Options (Figure 3).

At the beginning of this output, from a previous run, existing data in the text file “ToDoFile” is displayed from the “Processing” section of the script. Initially, the existing or “current” tasks displayed are “LAUNDRY”, “HOMEWORK”, and “VACUUM” (Figure 3), but after the program runs through and the user makes changes, the resulting tasks saved to the text file are “LAUNDRY”, “HOMEWORK”, and “GARDENING,” as seen in Figure 4.

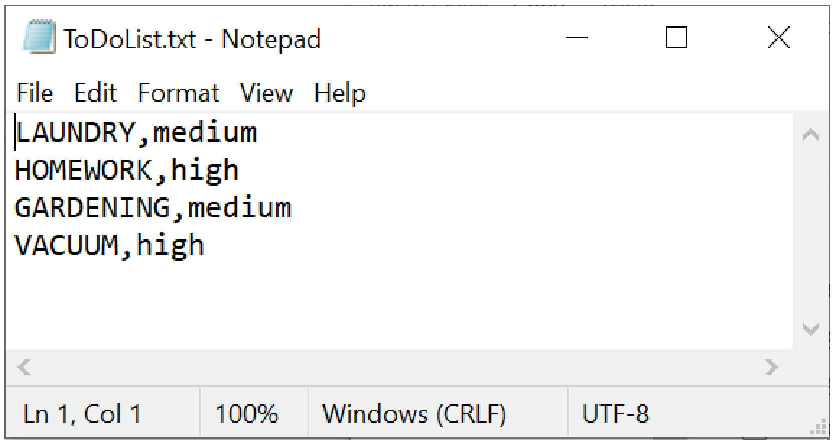


```
C:\Users\jesst\anaconda3\envs\Assignment05\python.exe C:/Users/jesst/Documents/_PythonClass/Assignment05/Assignment05.py
[{'Task': 'LAUNDRY', 'Priority': 'medium'}, {'Task': 'HOMEWORK', 'Priority': 'high'}, {'Task': 'VACUUM', 'Priority': 'low'}]

    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

Current Data:
LAUNDRY,medium
HOMEWORK,high
VACUUM,low

    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
```



```
Enter a task and level of priority:
Task: Gardening
Priority (high, medium, low): medium
LAUNDRY, medium
HOMEWORK, high
VACUUM, low
GARDENING, medium

    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

Task to remove: vacuum
row removed
Remaining Tasks:
LAUNDRY,medium
HOMEWORK,high
GARDENING,medium

    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

Task to remove: slakdjf;lsaf
Row not found

    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
    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

The program has ended

(Press Enter to Exit)
```

Figure 3. (Above) Script ran in PyCharm Console with all possible input functions from menu options (1 through 5).



```
LAUNDRY,medium
HOMEWORK,high
GARDENING,medium
VACUUM,high
```

Figure 4. “ToDoList” text file with tasks saved by user from using the program in Figure 3, and from the command line (see image captures “cmdline_1” and cmdline_2” PNG files in repository)

Summary/Discussion

This was the toughest assignment yet; I definitely got stuck almost as soon as I started and hit many walls when trying to understand the variables declared. While working my way through the program, I found that a couple variables, “strMenu” and “strData” weren’t even present in the actual program script. I was also hesitant about introducing new variables (i.e. “ToDoFile”), but found it hard to work with just the variables given. I feel like a big struggle that I had when creating this program is knowing how to handle each part, and each part took a lot longer than I allotted time for. I was able to work through one section successfully but have it cause errors in another part of my program. This was super frustrating because I soon learned that progress is far from linear when it comes to coding; and that I have a lot of room for improvement.

I was able to get this program to work to its most basic function, but I can’t say that I am super proud of it, or that it is my best work. As I continue to go through the program, I have ideas that would make this program more user friendly, such as implementing a nested loop in step 4 (to have the user add the tasks and priorities until they exit the loop), but for some reason can’t get my mind to clearly execute it. While I am still toggling and modifying things to test my script, I hope that future lectures and assignments will chip away these gaps.