Matthew Tsao

November 22, 2022

Foundations Of Programming: Python

Assignment06

GitHub URL: https://github.com/mrtsao1/IntroToProg-Python-Mod06

Data Processing of Text Files using Functions and Classes

Introduction

This week, I learned about functions and classes (declaration, calling within your script). I created a program read in data from a text file, format the data into a list of dictionary items, add/remove/display the items in this list, and export this list to a text file. The following information is a breakdown on how I wrote this program step-by-step.

Creating and Running the Program

Refer to the following subsections below.

Script Header

Create the script header. A script header should contain the title of the script file, a description, and a change log. A script header is created using comments, which consists of either starting the line with "#" (single line comment) or multi-line comments (Figure 1).

```
1    '''
2    Title: Assignment 06
3    Description: Working with functions in a class,
4                 When the program starts, load each "row" of data
5                 in "ToDoList.txt" into a python Dictionary.
6                 Add each dictionary "row" to a python list "table"
7    Change Log: (Who, When, What)
8    MTsao, 2022-11-18, Created File
9    MTsao, 2022-11-19, Completed add new task step
10   MTsao, 2022-11-21, Completed remove existing task, save file steps
11   '''
```

Figure 1. Script Header

Initialize Variables

Create variables with initialized values that will be used throughout the program (Figure 2). The key variables and their descriptions are shown in figure 2 below.

```
13   # Data ------------------------------------------------------------- #
14   # Declare variables and constants
15   file_name_str = "ToDoFile.txt"   # The name of the data file
16   file_obj = None                   # An object that represents a file
17   row_dic = {}                      # A row of data separated into elements of a dictionary {Task,Priority}
18   table_lst = []                    # A list that acts as a 'table' of rows
19   choice_str = ""                   # Captures the user option selection
```

Figure 2. Initialize Variables

Classes

Classes are a way of grouping functions, variables, and constants. In this program, Processor and IO classes will be created and used. See subsections below for more information.

Processor Class

The Processor class is created and will be used to perform processing tasks (Figure 3).

```
23   class Processor:
24       """ Performs Processing tasks """
```

Figure 3. Processor Class

(i)   Read Data from File

The "read_data_from_file" function is created and requires two input arguments "file_name" and "list_of_rows" (Figure 4). This function clears any data that was in "list_of_rows", reads in the data from "file_name" line by line, adds this data to "list_of_rows", closes the file, and returns the "list_of_rows".

```
26          @staticmethod
27          def read_data_from_file(file_name, list_of_rows):
28              """ Reads data from a file into a list of dictionary rows
29
30              :param file_name: (string) with name of file:
31              :param list_of_rows: (list) you want filled with file data:
32              :return: (list) of dictionary rows
33              """
34              list_of_rows.clear()  # clear current data
35              file = open(file_name, "r")
36              for line in file:
37                  task, priority = line.split(",")
38                  row = {"Task": task.strip(), "Priority": priority.strip()}
39                  list_of_rows.append(row)
40              file.close()
41              return list_of_rows
```

Figure 4. read_data_from_file function

(ii)   Add Data to List

The "add_data_to_list" function is created and requires three input arguments "task", "priority", and "list_of_rows" (Figure 5). This function formats "task" and "priority" into a dictionary row, appends this row to "list_of_rows", and returns the "list_of_rows".

```
43          @staticmethod
44          def add_data_to_list(task, priority, list_of_rows):
45              """ Adds data to a list of dictionary rows
46
47              :param task: (string) with name of task:
48              :param priority: (string) with name of priority:
49              :param list_of_rows: (list) you want filled with file data:
50              :return: (list) of dictionary rows
51              """
52              row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
53              list_of_rows.append(row)  # add task/priority to existing list
54              return list_of_rows
```

Figure 5. add_data_to_list function

(iii)   Remove Data from List

The "remove_data_from_list" function is created and requires two input arguments "task" and "list_of_rows" (Figure 6). This function compares each "task" item in "list_of_rows" with input argument "task". If a match is found, the dictionary row with the match will be deleted from "list_of_rows". If a

match is NOT found, the user will be notified that the task could not be deleted because it could not be found. The function will then return the "list_of_rows".

```python
@staticmethod
def remove_data_from_list(task, list_of_rows):
    """ Removes data from a list of dictionary rows

    :param task: (string) with name of task:
    :param list_of_rows: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """
    boolMatch = False  # create a bool to track status of search for task to delete
    for objRow in list_of_rows:  # search each row
        if objRow['Task'] == task:  # task to delete was found
            list_of_rows.remove(objRow)  # delete task/priority from list
            print("Deleted task \"", task, "\" from list\n")  # notify user of status
            print(task)
            boolMatch = True
    if not boolMatch:  # task to delete was NOT found, notify user of status
        print("Unable to delete task. \"", task, "\" was not found.\n")
    return list_of_rows
```

Figure 6. remove_data_from_list function

(iv)   Write Data to File

The "write_data_to_file" function is created and requires two input arguments "file_name" and "list_of_rows" (Figure 7). This function opens the file "file_name", writes each dictionary row in "list_of_rows" to the file, closes the file, and returns "list_of_rows". Any data that previously existed in the file "file_name" is overwritten.

```python
@staticmethod
def write_data_to_file(file_name, list_of_rows):
    """ Writes data from a list of dictionary rows to a File

    :param file_name: (string) with name of file:
    :param list_of_rows: (list) you want filled with file data:
    :return: (list) of dictionary rows
    """
    objFile = open(file_name, "w")  # create/open output file
    for objRow in list_of_rows:  # write data to output file
        objFile.write(objRow["Task"] + ',' + objRow["Priority"] + '\n')
    objFile.close()  # close output file
    return list_of_rows
```

Figure 7. write_data_to_file function

IO (Input/Output) Class

The IO class is created and will be used to perform input and output tasks (Figure 8).

```
94      class IO:
95          """ Performs Input and Output tasks """
```

Figure 8. IO (Input/Output) Class

(i)   Output Menu Tasks

The "output_menu_tasks" function is created and contains no input or output arguments (Figure 9). This function prints the menu.

```
97          @staticmethod
98          def output_menu_tasks():
99              """  Display a menu of choices to the user
100
101             :return: nothing
102             """
103             print('''
104         Menu of Options
105         1) Add a new Task
106         2) Remove an existing Task
107         3) Save Data to File
108         4) Exit Program
109         ''')
110             print()  # Add an extra line for looks
```

Figure 9. output_menu_tasks function

(ii)   Input Menu Choice

The "input_menu_choice" function is created and contains no input arguments (Figure 10). This function prompts the user to select an option (1 – 4) from menu and returns the user's input.

```
112         @staticmethod
113         def input_menu_choice():
114             """ Gets the menu choice from a user
115
116             :return: string
117             """
118             choice = str(input("Which option would you like to perform? [1 to 4] - ")).strip()
119             print()  # Add an extra line for looks
120             return choice
```

Figure 10. input_menu_choice function

(iii) Output Current Tasks in List

The "output_current_tasks_in_list" function is created and requires one input "list_of_rows" (Figure 11). This function prints the tasks and priority of the items in "list_of_rows". No output arguments for this function.

```python
122        @staticmethod
123        def output_current_tasks_in_list(list_of_rows):
124            """ Shows the current Tasks in the list of dictionaries rows
125
126            :param list_of_rows: (list) of rows you want to display
127            :return: nothing
128            """
129            rowCount = 1     # numbering system for printing out current tasks
130            print("******* The current tasks ToDo are: *******")
131            for row in list_of_rows:
132                print(str(rowCount) + ".\t" + row["Task"] + " (" + row["Priority"] + ")")
133                rowCount += 1
134            print("****************************************")
135            print()  # Add an extra line for looks
```

Figure 11. output_current_tasks_in_list function

(iv) Input New Task and Priority

The "input_new_task_and_priority" function is created and requires no input arguments (Figure 12). This function prompts the user for the task and priority level and returns the user's inputs.

```python
137        @staticmethod
138        def input_new_task_and_priority():
139            """  Gets task and priority values to be added to the list
140
141            :return: (string, string) with task and priority
142            """
143            pass
144            print("\t\tAdd a new task")
145            strTask = input("\t\tState the task: ")  # prompt user for task
146            strPriority = input("\t\tState the Priority level: ")  # prompt user for priority level
147            print()  # add an extra line for looks
148            return strTask, strPriority
```

Figure 12. input_new_task_and_priority function

(v) Input Task to Remove

The "input_task_to_remove" function is created and requires no input arguments (Figure 13). This function prompts user for task that they want to delete and returns the user's input.

```
150         @staticmethod
151         def input_task_to_remove():
152             """  Gets the task name to be removed from the list
153
154             :return: (string) with task
155             """
156             pass
157             print("\t\tRemove an existing Task")
158             strFindTerm = input("\t\tWhat task do you want to delete?: ")  # prompt user for task to delete
159             print()  # add an extra line for looks
160             return strFindTerm
```

Figure 13. input_task_to_remove function

Main Body of Script

Load Data from Text File

This section imports the data from the data file by calling the "read_data_from_file" function from the Processor class (Figure 14).

```
163     # Main Body of Script  --------------------------------------------- #
164
165     # Step 1 - When the program starts, Load data from ToDoFile.txt.
166     Processor.read_data_from_file(_file_name=file_name_str, list_of_rows=table_lst)  # read file data
```

Figure 14. Read in Data from File

Select Option From Menu

The program will call the "output_current_tasks_in_list" and "output_menu_tasks" functions from the IO class to display the current data followed by the menu. Then, the "input_menu_choice" function from the IO class will be called, prompting the user to choose an option from the menu (Figure 15). This process will be repeated until the user exits the program.

```
168     # Step 2 - Display a menu of choices to the user
169     while (True):
170         # Step 3 Show current data
171         IO.output_current_tasks_in_list(list_of_rows=table_lst)  # Show current data in the list/table
172         IO.output_menu_tasks()  # Shows menu
173         choice_str = IO.input_menu_choice()  # Get menu option
```

Figure 15. Select Option from Menu

Evaluate Selected Option

The menu choices are:

1) Add a New Item
2) Remove an Existing Item
3) Save Data to File
4) Exit Program

The program uses an if/elif…/else statement to evaluate the user's choice. These menu options are explored in the subsections below.

(i)   Add a New Item

The user entered "1" to add a new item to the list. The program will call the
"input_new_task_and_priority" function from the IO class to prompt the user to enter the "task" and
"priority". The program will take the inputs, format them into a dictionary item, and append it to the
existing list using "add_data_to_list" function from the Processor class (Figure 16).

```
175         # Step 4 - Process user's menu choice
176    ▽    if choice_str.strip() == '1':  # Add a new Task
177             task, priority = IO.input_new_task_and_priority()
178             table_lst = Processor.add_data_to_list(task=task, priority=priority, list_of_rows=table_lst)
179    △         continue  # to show the menu
```

Figure 16. Add a New Item

(ii)   Remove an Existing Item

The user entered "2" to remove an item from the list (Figure 17). The program will call the
"input_task_to_remove" function from the IO class to prompt the user to enter the "task" name. Then,
the program will call the "remove_data_from_list" function from the Processor class using the user
input and existing list "table_list" as input arguments.

```
181    ▽        elif choice_str == '2':  # Remove an existing Task
182             task = IO.input_task_to_remove()
183             table_lst = Processor.remove_data_from_list(task=task, list_of_rows=table_lst)
184    △         continue  # to show the menu
```

Figure 17. Remove an Existing Item

(iii)   Save Data to File

The user entered "3" to save the data to output file (Figure 18). The program calls the
"write_data_to_file" function from the Processor class to save the task list to the data file. Any data that
previously existed in the output file will be overwritten. The user will be notified that the data has been
saved to the output file.

```
186    ▽        elif choice_str == '3':  # Save Data to File
187             table_lst = Processor.write_data_to_file(file_name=file_name_str, list_of_rows=table_lst)
188             print("\t\tSave Data to File")
189             print("\t\tData Saved to \"" + file_name_str + "\"!\n")
190    △         continue  # to show the menu
```

Figure 18. Save Data to File

(iv)   Exit Program

The user entered "4" to exit the program (Figure 19). The program will display "Exit Program" and
"Goodbye!" to the user before terminating.

```
192          elif choice_str == '4':  # Exit Program
193              print("\t\tExit Program")
194              print("\t\tGoodbye!")
195          break  # by exiting loop
```

Figure 19. Exit Program

(v)   Invalid Choice

If the user input was not "1" through "4", the user will see the message "Invalid choice entered. Please try again." and will be prompted to make another selection (Figure 20).

```
197          else:   # Invalid choice
198              print("\t\tInvalid choice entered. Please try again.\n")
```

Figure 20. Invalid Choice

Verify Program Worked

My sample input file "ToDoList.txt" contains 4 items of data (Figure 21). I used this sample file to test my program in PyCharm and OS command/shell window.
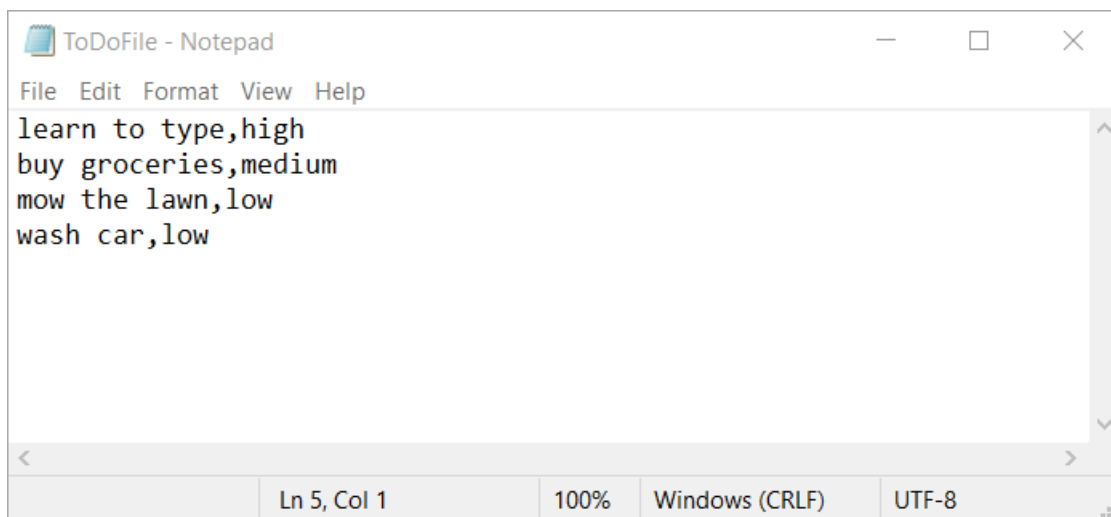

Figure 21. ToDoList.txt

Run Program Using PyCharm

(i)   Add a New Item

Run script in PyCharm and add item to list, showing that the script worked as expected (Figure 23).

```
C:\_PythonClass\Assignment06\venv\Scripts\python.exe C:/_PythonClass/Assignment06/Assignment06.py
******* The current tasks ToDo are: *******
1.  learn to type (high)
2.  buy groceries (medium)
3.  mow the lawn (low)
4.  wash car (low)
****************************************


        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program


Which option would you like to perform? [1 to 4] - 1

        Add a new task
        State the task: pay my taxes
        State the Priority level: high

******* The current tasks ToDo are: *******
1.  learn to type (high)
2.  buy groceries (medium)
3.  mow the lawn (low)
4.  wash car (low)
5.  pay my taxes (high)
****************************************
```

Figure 22. PyCharm - Add new item to list

(ii)   Remove an existing Item

The user attempts to delete a task that is not in the list (Figure 24). The user deletes a task that exists in the list (Figure 25).

```
Which option would you like to perform? [1 to 4] - 2


        Remove an existing Task
        What task do you want to delete?: task that doesn't exist


Unable to delete task. " task that doesn't exist " was not found.


******* The current tasks ToDo are: *******
1.  learn to type (high)
2.  buy groceries (medium)
3.  mow the lawn (low)
4.  wash car (low)
5.  pay my taxes (high)
*****************************************



        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program
```

Figure 23. Delete a task that DOES NOT exist

```
Which option would you like to perform? [1 to 4] - 2


        Remove an existing Task
        What task do you want to delete?: mow the lawn


Deleted task " mow the lawn " from list


mow the lawn
******* The current tasks ToDo are: *******
1.  learn to type (high)
2.  buy groceries (medium)
3.  wash car (low)
4.  pay my taxes (high)
*****************************************



        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program
```

Figure 24. Delete a task that DOES exist

(iii)   Save Data to File
The user saves the data to output file "ToDoToDoList.txt" (Figure 26).

```
Which option would you like to perform? [1 to 4] - 3

        Save Data to File
        Data Saved to "ToDoFile.txt"!

******* The current tasks ToDo are: *******
1.  learn to type (high)
2.  buy groceries (medium)
3.  wash car (low)
4.  pay my taxes (high)
*****************************************



        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program
```

Figure 25. Save Data to File

(iv)   Exit Program

The user exits the program(Figure 27). The program prompts the user to press enter on the line stating "Press the enter key to exit.".

```
Which option would you like to perform? [1 to 4] - 4

        Exit Program
        Goodbye!

Process finished with exit code 0
```

Figure 26. Exit Program

(v)   Verify Contents of Output File

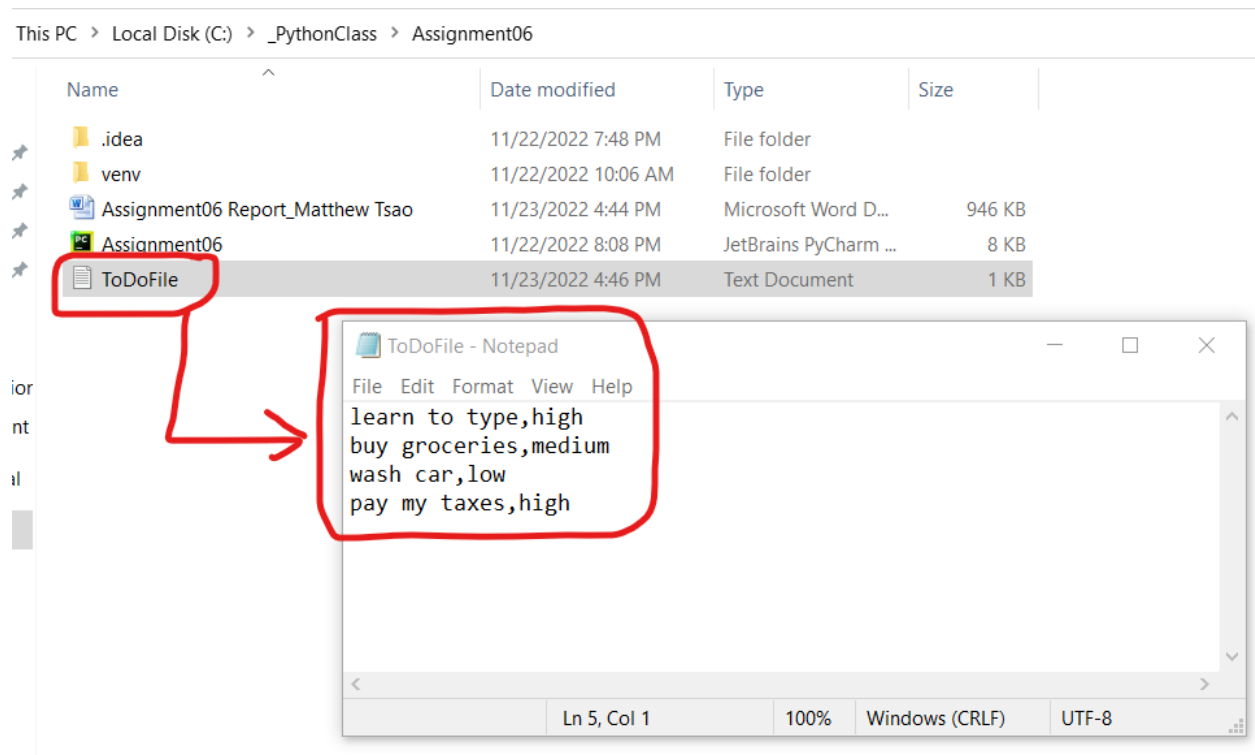Check to verify that the output file was created with data as expected (Figure 28).

Figure 27. Verification of Output File

Run Program Using OS command/shell window

Snip of Assignment06.py running using shell window (Figure 29).

Figure 28. Run Assignment06.py in shell window

Summary

The Assignment06.py program demonstrated the following concepts:

- Data manipulation using dictionaries (Add/remove dictionary entries)
- TODO comment to keep track of status when developing scripts
- Comparisons of user inputs to stored data entries to determine if an item exists within a list