Mike Hamilton

Foundations of Programming: Python

Assignment 6

11/22/21

Extending an Interactive Python Script for Tracking To-Do Items Using Functions

Summary

- In this assignment, we took the basic outline of a Python script we had worked on in previous assignments and re-factored the code using some newly-defined classes and their associated functions.
- The script initially reads in any existing to-do items from a file, and then asks the user to display, add, remove, or write the new results back to that same file.
- We were required to run this script in both PyCharm and in a command window and show a screenshot of the output. We were also required to post the files to a new Github repository.
- In the sections below, I provide more detail on this process, and provide screenshots to document key steps and outputs.

Part 1: Creating the Script

In PyCharm, I created a new project in the 'Assignment_06' directory and then opened a new python script file. I then pasted in the starter syntax and modified it as follows:

```
# ------------------------------------------------------------------------ #
# Title: Assignment 06
# Description: Working with functions in a class,
#              When the program starts, load each "row" of data
#              in "ToDoList.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
# MHamilton,11.21.21,Modified code to complete assignment 06
# ------------------------------------------------------------------------ #

# Import os for relative path abilities.
import os

# Data ------------------------------------------------------------------ #
# Declare variables and constants
dirname = os.path.dirname(__file__)                     # The current directory where the script is located
file_name_str = os.path.join(dirname, "ToDoList.txt")   # The name of the data file (with relative path)
file_obj = None                                         # An object that represents a file
row_dic = {}            # A row of data separated into elements of a dictionary {Task,Priority}
table_lst = []          # A list that acts as a 'table' of rows
choice_str = ""         # Captures the user option selection


# Processing  ----------------------------------------------------------- #
class Processor:
    """  Performs Processing tasks """

    @staticmethod
    def read_data_from_file(file_name, list_of_rows):
        """ Reads data from a file into a list of dictionary rows

        :param file_name: (string) with name of file:
        :param list_of_rows: (list) you want filled with file data:
```

```
        :return: (list) of dictionary rows
        """
        list_of_rows.clear()  # clear current data
        try:
            file = open(file_name, "r")
            for line in file:
                task, priority = line.split(",")
                # Here we make sure to convert all text to lower case to match with later
                # functionality.
                row_dic = {"Task": task.strip().lower(), "Priority": priority.strip().lower()}
                list_of_rows.append(row_dic)
            file.close()
            return list_of_rows
        except:
            print("No file found. You may add tasks and a new file will be created.")

    @staticmethod
    def add_data_to_list(task, priority, list_of_rows):
        """ Adds user-defined task/priority items into a list of dictionary rows

        :param task: (string) user-defined task name:
        :param priority: (string) user-defined priority for task:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        # insert into data table
        list_of_rows.append({"Task": task, "Priority": priority})
        return list_of_rows

    @staticmethod
    def remove_data_from_list(task, list_of_rows):
        """ Removes user-defined task/priority items from a list of dictionary rows

        :param task: (string) user-defined task name:
        :param list_of_rows: (list) you want filled with file data:
        :return: (list) of dictionary rows
        """
        # first set removed parameter as false, then loop through table and identify item to be removed
        itemRemoved = False
        for row_dic in list_of_rows:
            if row_dic["Task"] == task.strip():
                list_of_rows.remove(row_dic)
                print("Item removed!")
                # Here we must set itemRemoved to True to not print the following block
                itemRemoved = True
        # if item not found, print error message to user
        if itemRemoved == False:
            print("Item not found.")

        return list_of_rows

    @staticmethod
    def write_data_to_file(file_name, list_of_rows):
        """ Write list of user-defined task/priority items to file

        :param file_name: (string) user-defined filename for output:
        :param list_of_rows: (list) you want filled with file data:
        :return: nothing
        """
        # Process the data into a file
        outfile = open(file_name, "w")
        for row_dic in list_of_rows:
            outfile.write(row_dic["Task"] + "," + row_dic["Priority"] + "\n")
        outfile.close()
        # Display a message to the user
        print("Data saved to file!")
        # Print extra line for space
        print()


# Presentation (Input/Output)  ------------------------------------------- #


class IO:
    """ Performs Input and Output tasks """

    @staticmethod
    def output_menu_tasks():
        """  Display a menu of choices to the user
```

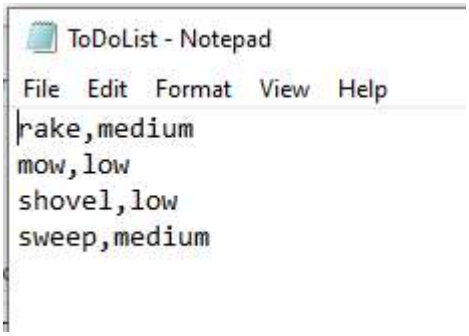
```
        :return: nothing
        """
        print('''
        Menu of Options
        1) Add a new Task
        2) Remove an existing Task
        3) Save Data to File
        4) Exit Program
        ''')
        print()  # Add an extra line for looks

    @staticmethod
    def input_menu_choice():
        """ Gets the menu choice from a user

        :return: string
        """
        try:
            choice = str(input("Which option would you like to perform? [1 to 4] - ")).strip()
            print()  # Add an extra line for looks
            return choice
        except:
            print("Please enter a numeric value between 1 and 4.")
            print()  # Add an extra line for looks

    @staticmethod
    def output_current_tasks_in_list(list_of_rows):
        """ Shows the current Tasks in the list of dictionaries rows

        :param list_of_rows: (list) of rows you want to display
        :return: nothing
        """
        print("******* The current tasks ToDo are: *******")
        for row_dic in list_of_rows:
            print(row_dic["Task"] + " (" + row_dic["Priority"] + ")")
        print("*******************************************")
        print()  # Add an extra line for looks

    @staticmethod
    def input_new_task_and_priority():
        """ Prompts user for new input (task and priority) to add to list

        :return: (tuple) task and string as a 2-element tuple:
        """
        print("Please enter a to-do item and its priority: ")
        task = input("Enter a to-do item: ").strip().lower()
        priority = input("Enter a priority: ").strip().lower()
        # return items as tuple
        return (task, priority)

    @staticmethod
    def input_task_to_remove():
        """ Prompts user for new input (task and priority) to add to list

        :return: (string) task:
        """
        # prompt user for task to remove
        print("Please enter the item you would like to remove: ")
        task = input("Enter name here: ")
        # return task
        return task.strip().lower()


# Main Body of Script  ------------------------------------------------------ #

# Step 1 - When the program starts, Load data from ToDoFile.txt.
Processor.read_data_from_file(file_name_str, table_lst)  # read file data

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

    # Step 4 - Process user's menu choice
    if choice_str.strip() == '1':  # Add a new Task
        # First get new input from user (a task and a priority)
```

```
        (task, priority) = IO.input_new_task_and_priority()
        # Now add that data to the list
        Processor.add_data_to_list(task, priority, table_lst)
        continue  # to show the menu

    elif choice_str == '2':  # Remove an existing Task
        # First ask user which item to remove
        task = IO.input_task_to_remove()
        # Now remove that item
        Processor.remove_data_from_list(task, table_lst)
        continue  # to show the menu

    elif choice_str == '3':  # Save Data to File
        # Here we write to file using pre-defined filename
        Processor.write_data_to_file(file_name_str, table_lst)
        continue  # to show the menu

    elif choice_str == '4':  # Exit Program
        print("Goodbye!")
        break  # and Exit
```

The script first tries to read in a list of to-do items based on a relative path. If no list is found, it prints an error message to the user, but then continues. It next prompts the user with a menu and waits for input. The user may choose to add items to the list, to remove an item, to write to an output (text) file, or to exit.

Part 2: Running the Script

The next part of this assignment was to run the script I had just created. I first ran it directly in the PyCharm IDE, starting with a list file with the following items:



During the script, I removed the “rake” item and added a “wash dishes” item. The script looked like this.

Screenshot 1 of 2

```
C:\_PythonClass\Assignment_06\venv\Scripts\python.exe C:/_PythonClass/Assig
******* The current tasks ToDo are: *******
rake (medium)
mow (low)
shovel (low)
sweep (medium)
*******************************************


        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] - 2

Please enter the item you would like to remove:
Enter name here: rake
Item removed!
******* The current tasks ToDo are: *******
mow (low)
shovel (low)
sweep (medium)
*******************************************
```

Screenshot 2 of 2:

```
        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

Please enter a to-do item and its priority:
Enter a to-do item: wash dishes
Enter a priority: high
******* The current tasks ToDo are: *******
mow (low)
shovel (low)
sweep (medium)
wash dishes (high)
*****************************************


        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] - 3

Data saved to file!
```

After running the script, the text file now looked like this:

ToDoList - Notepad

File Edit Format View Help

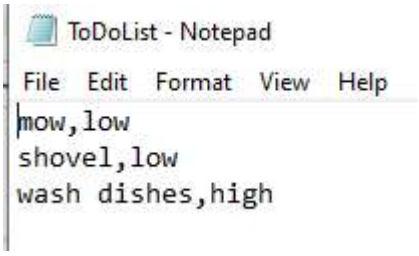
```
mow,low
shovel,low
sweep,medium
wash dishes,high
```

I then opened a command terminal window by typing “cmd” in the file directory bar. I then typed in “python Assignment06.py” to execute the script located within this folder. The results of the script are shown below:

Screenshot 1 of 1:

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

C:\_PythonClass\Assignment_06>python Assignment_06.py
******* The current tasks ToDo are: *******
mow (low)
shovel (low)
sweep (medium)
wash dishes (high)
*******************************************


        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] - 2

Please enter the item you would like to remove:
Enter name here: sweep
Item removed!
******* The current tasks ToDo are: *******
mow (low)
shovel (low)
wash dishes (high)
*******************************************


        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] - 3

Data saved to file!

******* The current tasks ToDo are: *******
mow (low)
shovel (low)
wash dishes (high)
*******************************************


        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] - 4

Goodbye!
```

Lastly, I checked to make sure that the output file had been processed correctly. The list now correctly contained the remaining items that had not been removed from the file during the command prompt execution (“sweep” was removed).



Conclusions and Observations

In this assignment we continued our work with dictionaries, but re-wrote the syntax to use functions (organized into classes). Using functions allows us to make the code both more flexible and easier to follow (when done correctly). Using functions seems to be an important part of the rest of this course.