Kyle Biondich

8/16/2023

IT FDN 110 A

Assignment 06

https://github.com/kbiondo/IntroToProg-Python-Mod06

ToDo Funs with Functions

Introduction

Week 6 of the course introduced classes and functions and how to use the return element when passing data between functions. The following paragraphs outline the methods that were used to read a text file into a python script, capture user input from a menu, and either add a new item to a list, read the total list, or write the list back to an external file.

Intended Outcome

The intended outcome of this assignment is to initially load a text file that consists of tasks and priorities if it exists and store the contents of that text file in a list and provide the ability to:

1. Add new tasks and their priorities to the list
2. Remove a task and its priority from the list
3. Save the contents of the list to a file
4. Exit the program.

```
PS D:\DEV\uw\assignments\_PythonClass\Assignment06> python .\Assignment06.py
******* The current tasks ToDo are: *******
cat (1)
dog (2)
jjj (5)
*******************************************


        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] - 
```

Figure 1: Intended Outcome: Assignment06.py Menu

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

Enter a task: new
Enter a priority: 4
******* The current tasks ToDo are: *******
cat (1)
dog (2)
jjj (5)
new (4)
*******************************************


        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] -
```

Figure 2: Intended Outcome: Assignment06.py Menu 1

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

Enter Task to Remove: cat
******* The current tasks ToDo are: *******
dog (2)
jjj (5)
new (4)
*******************************************


        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] -
```

Figure 3: Intended Outcome: Assignment06.py Menu 2

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

Data Saved!
******* The current tasks ToDo are: *******
dog (2)
jjj (5)
new (4)
*****************************************


        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] -
```

Figure 4: Intended Outcome: Assignment06.py Menu 3

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

Goodbye!
PS D:\DEV\uw\assignments\_PythonClass\Assignment06>
```

Figure 5: Intended Outcome: Assignment06.py Menu 4

Declare Variables and constants

The starter assignment python file contained a bunch of starting variables and constants, as seen in figure 6.

```
# Data ------------------------------------------------------------------- #
# Declare variables and constants
file_name_str = "ToDoFile.txt"  # The name of the data file
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
```

Figure 6: Variables and Constants

Step 1 – Processing Class – Read Data from file

In this step, the function read_data_from_file is defined. It starts with using the open function with the append argument to ensure a 'TodoFile.txt' exists, then uses the open function again with the 'read' argument to read the contents of the file, if any. It then loops through the file with a for loop and puts any contents of the file first into a dictionary row, then into a list of dictionary rows.

```python
# Processing  ------------------------------------------------------------ #
Kyle, 3 days ago | 1 author (Kyle)
class Processor:      Kyle, 3 days ago • finished hw 06 python
    """  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
        Args:
            file_name (string): name of the file
            list_of_rows (list): list of dictionary rows
        Returns:
            list: list of dictionary rows
        """
        # Reads data from a file into a list of dictionary rows

        file_obj = open(file_name, "a")

        list_of_rows.clear()  # clear current data

        file_obj = open(file_name, "r")

        for row in file_obj:
            file_data = row.split(",")
            list_of_rows.append({"Task": file_data[0].strip(), "Priority": file_data[1].strip()})
        return list_of_rows

```

Figure 7: Load a File into a dictionary list

Step 2 – Add data to list

Next, the starter file contained the 'add_data_to_list' function. In this function I added a counting variable along with a while loop to check if the data passed into the function was already in the list. Only if the task name wasn't found would it be added to the list.

```python
    @staticmethod
    def add_data_to_list(task, priority, list_of_rows):
        """ Adds data to a list of dictionary rows

        :param task: (string) with name of task:
        :param priority: (string) with name of priority:
        :param list_of_rows: (list) you want to add more data to:
        :return: (list) of dictionary rows
        """
        row_dic = {"Task": task.strip(), "Priority": priority.strip()}

        items = 0 # start counter at 0
        while items < len(list_of_rows):
            if list_of_rows[items]["Task"] != task:
                items += 1
            elif list_of_rows[items]["Task"] == task:
                return list_of_rows
        list_of_rows.append(row_dic)
        return list_of_rows
```

Figure 8: Add data to list

Step 3 – Remove Data from List

The next function is the 'remove_data_from_list' function. Similar to the previous function, I use a counting variable and a while loop to check each item in the list against the item that was captured from the user. If the item in the list is found, it is removed from the list.

```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
        """
        # TODO: Add Code Here!

        items = 0 # start counter at 0
        while items < len(list_of_rows):
            if list_of_rows[items]["Task"] != task:
                items += 1
            elif list_of_rows[items]["Task"] == task:
                list_of_rows.pop(items)
            else:
                break
        return list_of_rows

```

Figure 9: Remove data from list

Step 4 – Write data to file

The next function 'write_data_to_file' calls the open function with the 'write' argument and writes the contents of the list to the file.

```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
        """
        # TODO: Add Code Here!
        write_file = open(file_name, "w")
        for items in list_of_rows:
            write_file.write(items['Task'] + ',' + items['Priority'] + '\n')
        write_file.close()

        return list_of_rows
```

Figure 10: Write list to file

Step 5 – Class IO

The next part of the script that was provided contains the input and output processing items. Here are the functions for displaying the menu to the user, capturing the users' choice, outputting the current list items, inputting a new list item, and removing a list item.

```
Kyle, 3 days ago | 1 author (Kyle)
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

```

Figure 11: Menu Choices

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

        :return: string
        """

        choice = str(input("Which option would you like to perform? [1 to 4] - ")).strip()
        print()  # Add an extra line for looks
        return choice

```

Figure 12: Input Menu Choice

```python
    @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 in list_of_rows:
            print(row["Task"] + " (" + row["Priority"] + ")")
        print("*****************************************")
        print()  # Add an extra line for looks

```

Figure 13: Current tasks in list

```python
    @staticmethod
    def input_new_task_and_priority():
        """  Gets task and priority values to be added to the list

        :return: (string, string) with task and priority
        """
        pass  # TODO: Add Code Here!
        task = str(input("Enter a task: "))
        priority = str(input("Enter a priority: "))
        return task, priority

```

Figure 14: Input new Task and Priority Item

```python
    @staticmethod
    def input_task_to_remove():
        """  Gets the task name to be removed from the list

        :return: (string) with task
        """
        pass
    # TODO Add Code Here!
        findTask = input("Enter Task to Remove: ")
        return findTask

```

Figure 15: Input task to remove

Step 6 – Main Body of the Script

This next section that was provided is the main script section, where the logic for running the script is contained. The script uses a while loop to keep displaying a list of the options a user can run. If the user chooses 1, it will run the part of the script that pertains to adding a new task and priority. If the user chooses 2, it will run the part of the script that pertains to removing an item form the list. If the user chooses 3, it will save the contents of the list to the 'ToDoList.txt' file, and if the user chooses 4, the script will exit.

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

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

while (True):
    # Step 3 Show current data
    IO.output_current_tasks_in_list(list_of_rows=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
        task, priority = IO.input_new_task_and_priority()
        table_lst = Processor.add_data_to_list(task=task, priority=priority, list_of_rows=table_lst)
        continue  # to show the menu

    elif choice_str == '2':  # Remove an existing Task
        task = IO.input_task_to_remove()
        table_lst = Processor.remove_data_from_list(task=task, list_of_rows=table_lst)
        continue  # to show the menu

    elif choice_str == '3':  # Save Data to File
        table_lst = Processor .write_data_to_file(file_name=file_name_str, list_of_rows=table_lst)
        print("Data Saved!")
        continue  # to show the menu

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

Figure 16: Main body of the Script

Observations

This was a rather difficult assignment. It took me many attempts to get the logic right on the processing steps of adding and removing tasks to the list. I eventually got the while loops to work, but it took a long time going through the debug method to eventually get it right. I also had issues with returning the correct outputs from one function to another. Again, the debug process was instrumental here.

Summary

In summary, utilizing all the resources provided to the class and the online lecture, this paper outlines all the steps that were taken to create a python script that results in a successful execution of the intended outcome (Figure 1). Following the steps outlined above will allow for the audience to recreate the presented result.