David Lammers

Nov 17, 2022

ToDoList 3.0.py

Assignment 06

Github link

To Do List 3.0 Script

Introduction

In this assignment we were asked to complete a script that was already created. This script introduced the concept of 'separation of concerns' and required us to understand how the different parts of the code called to and interacted with other parts of the code.

Writing the Script

Since most of the script was already written, I spent a few minutes familiarizing myself with the layout and the different functions. The blocks are separated neatly and in a manner that makes sense. The *Processor* block handles the data, the *IO* block takes the user input when needed and returns any data back to the processor for processing and the main body of the script bring it all together and handles what data was being collected and where in the processor it needs to be sent all based on the user input to the main menu of options. It seemed the best place to start would be at the end, in the main body, so that I could trace where the program was looking at for each line of code.

The program was missing code starting at the *IO* function *input_new_task_and _priority()*. For user input == 1, the program is looking to the *IO* class and the method for the variables *task* and *priority* (figure 1-red). If you go to that part of the script, you can see that the function *input_new_task_and _priority()* should be returning new *task* and *priority* variables to the processor (fig 1-green). This requires the program to ask the user for some input. I created the *task* and *priority* variables to ask the user for this input. While in there, I thought that this would be an appropriate place to relay the inputs back to the user in a printed note back to them as well. An example of the what the end product looks like can be seen in figure 2.

Now that the first line of code is satisfied in the *if* statement, the second line is defining what will be added to the *table_lst.* From here, I went back to the *Processor* and to the function *add_data_to_list()* (fig 1-blue). By the description, I knew that this function was going to need to take those inputs that were just generated by the function *input_new_task_and _priority()* and place them into the *table_lst.* Since the table is a list of dictionaries, I knew that the dictionary variable would be defined similarly to what we had done in the previous assignments, so this part was pretty simple. Then I appended the *list_of_rows* with *.append* and added the row. With this new row now added on to the existing list, I returned this back to the main body of the script. Now I tested the script and typed in "1" when prompted, entered a test task and priority and check if it is functioning correctly. I was able to see the newly added task when I ran the script, selected 1, and typed in a new task (fig 2). Great!

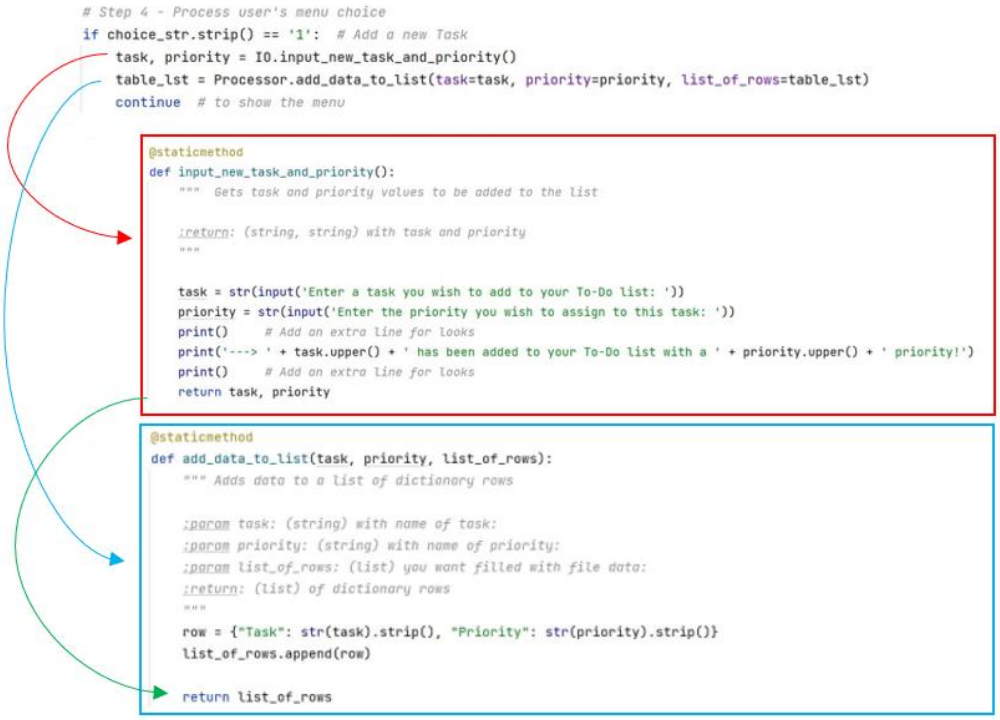


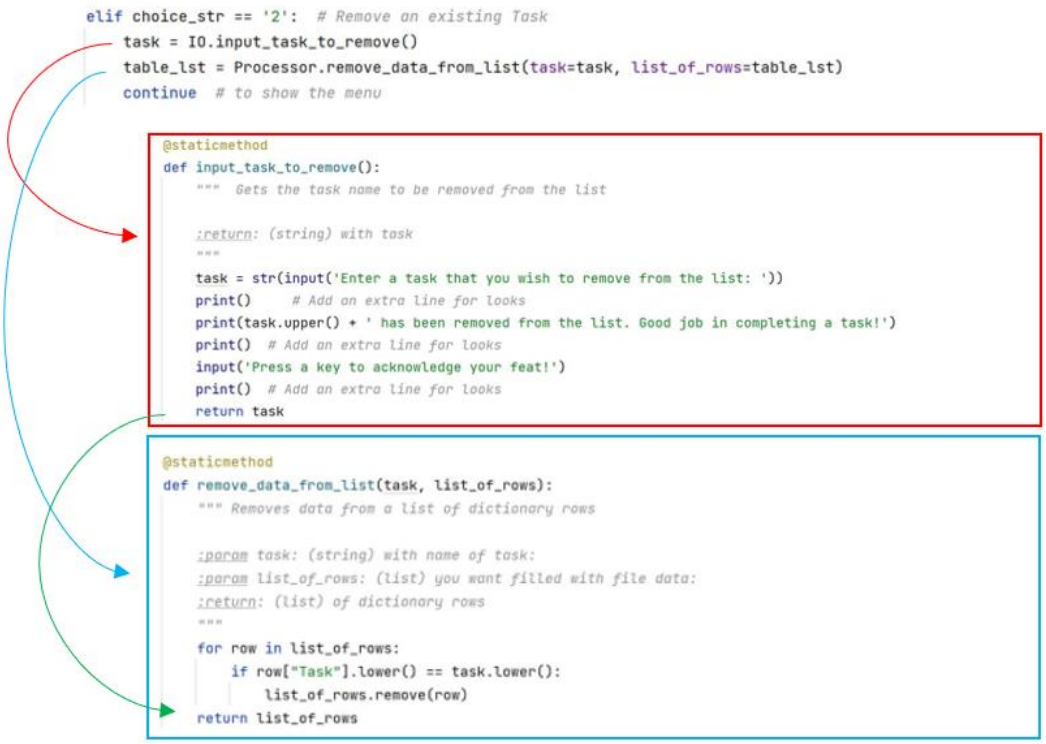
Figure 1 Menu Option "1" Code

```
Enter a task you wish to add to your To-Do list: eat turkey
Enter the priority you wish to assign to this task: high

---> EAT TURKEY has been added to your To-Do list with a HIGH priority!

******* The current tasks ToDo are: *******
drink water (high)
eat turkey (high)
*****************************************
```

Figure 2 Menu option 1 being tested

I then moved on to menu option “2” in a similar fashion. The main body of the script points to the *IO* class once more, but this time it looks at the *input_task_to_remove()* function (fig 3-red). Again, a user input would be required in order to select which one of the tasks the user desired to remove from the list so I wrote an input line redefining the local variable *task*. Once again, the output from this function would need to get sent to the processor for processing (fig 3- green). The function *remove_data_from_list()* has to look through each dictionary item and see if it matches the *task* variable that was just created in the previous step of the program. If it is found, it removes the

dictionary row from the list. Once this loop is satisfied, the *list_of_rows* is returned according to the program (fig 3- blue).



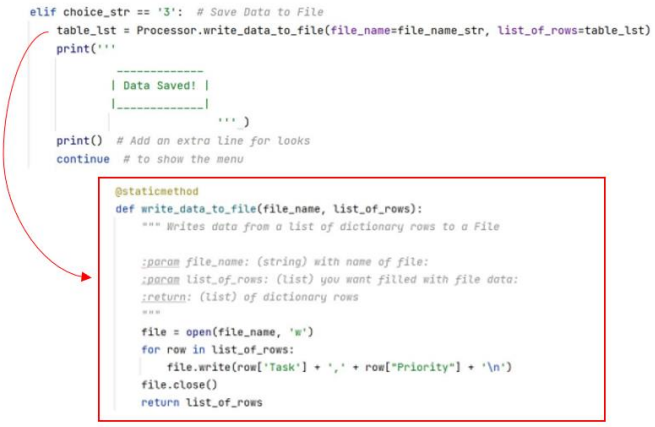
```
elif choice_str == '2':  # Remove an existing Task
    task = IO.input_task_to_remove()
    table_lst = Processor.remove_data_from_list(task=task, list_of_rows=table_lst)
    continue  # to show the menu
```

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

    :return: (string) with task
    """
    task = str(input('Enter a task that you wish to remove from the list: '))
    print()     # Add an extra line for looks
    print(task.upper() + ' has been removed from the list. Good job in completing a task!')
    print()  # Add an extra line for looks
    input('Press a key to acknowledge your feat!')
    print()  # Add an extra line for looks
    return task
```

```
@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
    """
    for row in list_of_rows:
        if row["Task"].lower() == task.lower():
            list_of_rows.remove(row)
    return list_of_rows
```

Figure 3 Menu Option "2" Code

The final code that needed to be completed concerned saving the data to the text file. First, the file needs to be opened with the write *'w'* string signifying that the file is being opened in order to write to it. Once the file is open, the function looks through all of the rows in the list that are currently being held in local memory, and writes these to a list in the text file. The format for this list is the standard comma separated format. Once written, the file is closed fig(4). After some testing I decided I wanted display the 'Data Saved' in a fun manner, so I played around with some string formatting. I think it came out pretty good (fig.5)



```
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! |
           |_____________|
                        ''' )
    print()  # Add an extra line for looks
    continue  # to show the menu
```

```
@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
    """
    file = open(file_name, 'w')
    for row in list_of_rows:
        file.write(row['Task'] + ',' + row["Priority"] + '\n')
    file.close()
    return list_of_rows
```

Figure 4 Menu Option "3" Code

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


 --------------
| Data Saved!  |
|______________|
```

Figure 5 'Data Saved!'

After all of the code was complete, I ran the code through the debugger. I set breakpoints at various lines of code just to play with the tools that are available. Since my code didn't have any bugs in it, it was uneventful as I just kept stepping through the code with no red flags. For the next assignment I plan to run the debugger more frequently, as I am developing the code. I could see that it can be useful by helping connect the different lines of code to one another.

Summary

This assignment made us understand how functions can be pulled out of the main body and called to only when needed. This is a really nice feature because the individual functions can be modified without touching the rest of the code. That is, as long as the inputs and outputs of the function are still satisfied. The complete code and some sample runs are located in the appendix.

Appendix

```
# ------------------------------------------------------------------------ #
# Title: ToDolist 3.0
# Description: Working with functions in a class,
#              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
# Dlammers, 11/22/2022, completed add data to list function in Processor and IO
# Dlammers, 11/23/2022, Completed delete data from list in Processor and IO
#                       Completed write data to file in Processor
# ------------------------------------------------------------------------ #

# 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


# 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
        file = open(file_name, "r")
        for line in file:
            task, priority = line.split(",")
            row = {"Task": task.strip(), "Priority": priority.strip()}
            list_of_rows.append(row)
        file.close()
        return list_of_rows

    @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 filled with file data:
        :return: (list) of dictionary rows
        """
        row = {"Task": str(task).strip(), "Priority": str(priority).strip()}
        list_of_rows.append(row)

        return list_of_rows

    @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
        """
        for row in list_of_rows:
            if row["Task"].lower() == task.lower():
                list_of_rows.remove(row)
        return list_of_rows

    @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
        """
        file = open(file_name, 'w')
        for row in list_of_rows:
            file.write(row['Task'] + ',' + row["Priority"] + '\n')
        file.close()
        return list_of_rows
```

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

    @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

    @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
        """

        task = str(input('Enter a task you wish to add to your To-Do list: '))
        priority = str(input('Enter the priority you wish to assign to this task: '))
        print()     # Add an extra line for looks
        print('---> ' + task.upper() + ' has been added to your To-Do list with a ' + priority.upper() + ' priority!')
        print()     # Add an extra line for looks
        return task, priority

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

        :return: (string) with task
        """
        task = str(input('Enter a task that you wish to remove from the list: '))
        print()     # Add an extra line for looks
        print(task.upper() + ' has been removed from the list. Good job in completing a task!')
        print()  # Add an extra line for looks
        input('Press a key to acknowledge your feat!')
        print()  # Add an extra line for looks
        return task
```

```python
# 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

# Step 2 - Display a menu of choices to the user
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! |
               |_____________|
                             ''')
        print()  # Add an extra line for looks
        continue  # to show the menu

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

Figure 6 Complete ToDoList 3.0.py Code

ToDoList 3.0

```
C:\Users\rp853d\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Python\Python_Class\Assignment06\ToDoList 3.0.py"
******* The current tasks ToDo are: *******
drink water (high)
eat turkey (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] - 1

Enter a task you wish to add to your To-Do list: eat pie
Enter the priority you wish to assign to this task: very high

---> EAT PIE has been added to your To-Do list with a VERY HIGH priority!

******* The current tasks ToDo are: *******
drink water (high)
eat turkey (high)
eat pie (very 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

Enter a task that you wish to remove from the list: eat turkey

EAT TURKEY has been removed from the list. Good job in completing a task!

Press a key to acknowledge your feat!

******* The current tasks ToDo are: *******
drink water (high)
eat pie (very 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! |
               |_____________|


******* The current tasks ToDo are: *******
drink water (high)
eat pie (very 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!

Process finished with exit code 0
```

Figure 7 Example code run in PyCharm IDLE

```
C:\Users\rp853d>python C:\Python\Python_Class\Assignment06\ToDoList3.py
There is currently no data saved to file.
******* The current tasks ToDo are: *******
****************************************


        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

Enter a task you wish to add to your To-Do list: eat turkey
Enter the priority you wish to assign to this task: high

---> EAT TURKEY has been added to your To-Do list with a HIGH priority!

******* The current tasks ToDo are: *******
eat turkey (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] - 1

Enter a task you wish to add to your To-Do list: eat pie
Enter the priority you wish to assign to this task: very high

---> EAT PIE has been added to your To-Do list with a VERY HIGH priority!

******* The current tasks ToDo are: *******
eat turkey (high)
eat pie (very 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

Enter a task that you wish to remove from the list: eat turkey

EAT TURKEY has been removed from the list. Good job in completing a task!

Press a key to acknowledge your feat!

******* The current tasks ToDo are: *******
eat pie (very high)
****************************************
```

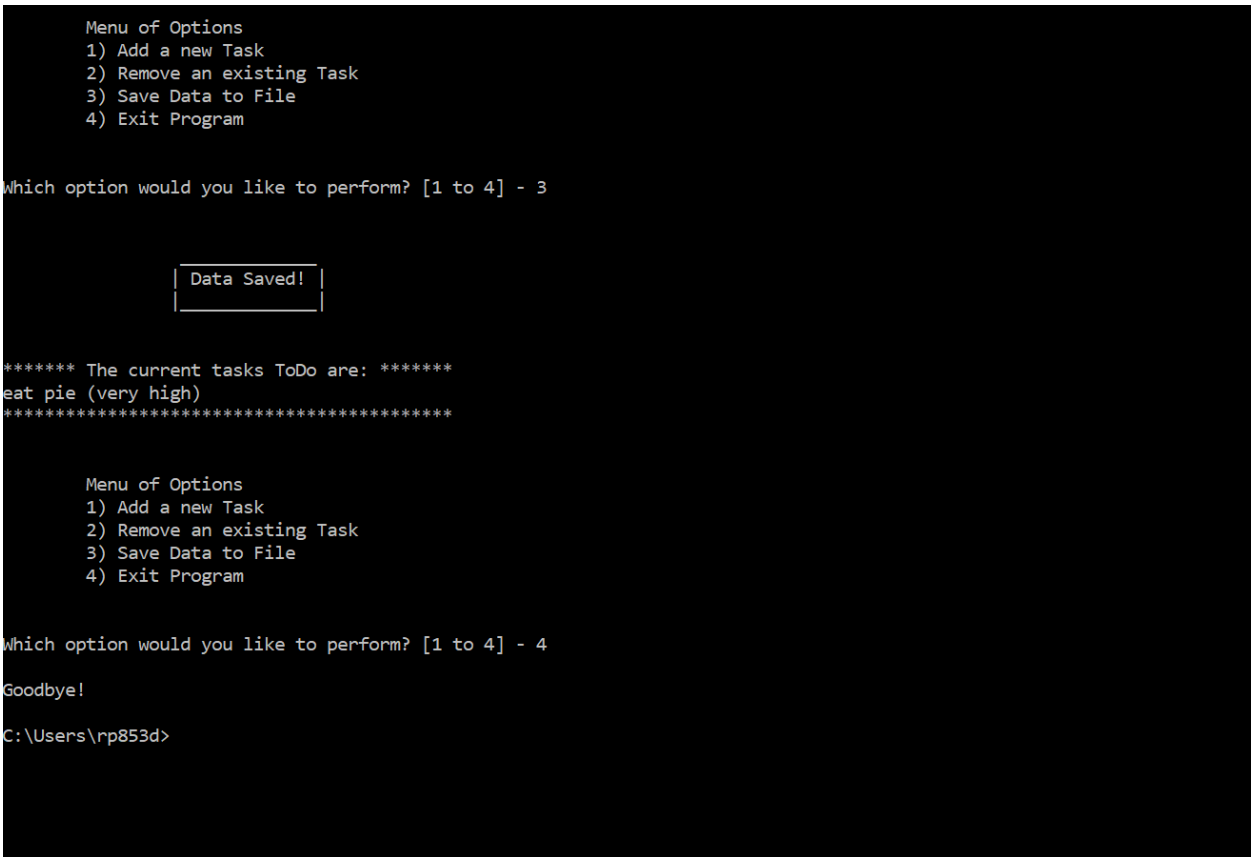


Figure 8 Example run in the Command Prompt