Navdeep Sandhu
ID: 1228819
Spring 2022 - IT FDN 110 – Python
May 23rd, 2022
https://github.com/navisandhu7/IntroToProg-Python-Mod06/

Assignment 06
To-Do List Manager Using Functions

INTRODUCTION

For the sixth assignment for IT FDN 110, students were tasked with modifying and utilizing existing code to create a program that would take user input for a task and its priority and save it to a text file for later use and display, creating a "to-do list." The objective was to create this program using functions where possible to perform repetitive tasks, as well as split the code as neatly as able into "data," "processing," and "input/output" sections. This document will describe the process I used to successfully create this code. Note that some of the code was written by R. Root. His contributions are given in the header portion of the code itself.

CODE WALKTHROUGH & REASONING

Figure 1 shows the code written to perform the specified function. The first several lines encompass the header of the code, which is self-explanatory and will not be covered in detail in this document. The rest of the code will be explained by section: Data, Processing, and I/O.

DATA

The first section of code encompasses initialization of any variables that will be used later in the code. Most items are initialized as empty variables, such as *row_dic* for an empty dictionary and *table_lst* for an empty table. The only initialized variable that has a value is *file_name_str*, which is a string that defines the name of the input and output text file for the stored to-do list. Of special note is the variable *file_obj*, which is an "empty" object for the file that will be opened later in the code. It is initialized via assignment of "None." One important note is that for this code to work properly, a completely blank "ToDoFile.txt" text file must be supplied. This file must also be devoid of newlines, tabs, spaces, and other "invisible" characters, else a "ValueError" will be thrown. This file is included in the zip package.

PROCESSING

The second section of code defines the "Processor" class and several functions that perform repetitive tasks. Note that all code written by N. Sandhu is almost a direct copy from Assignment 05, as the goal of the code has not changed between assignments. Thus, I will not cover the explanation of the code itself in detail in this document. For more information, please refer to N. Sandhu's submission for Assignment 05.

The first function in this section reads data from the file *file_name*. It reads in any existing data and appends it to the local variable *list_of_rows*. The function returns *list_of_rows* back to the main code. Both *file_name* and *list_of_rows* must be supplied to this function. Generally, *file_name* will be “ToDoFile.txt” (*file_name_str*) and *list_of_rows* will be an empty list (*table_lst*).

The next function adds data to the existing list. It must be supplied a task, a priority, and the list to which the data will be appended. The function returns the modified list back to the main code. Generally, these parameters will be defined by user inputs for adding a new task and its priority, which should both be stripped strings. The *list_of_rows* parameter will generally be the *table_lst* variable such that data will be appended to the main list of tasks and priorities.

The third function in this section will remove an existing task if it can be found in the supplied list. The function requires a string input for a task as well as the main list of data (generally, *table_lst*). The function will search in the parameter *list_of_rows* for the supplied *task* string. If found, the *list_of_rows* parameter is modified and the supplied *task* is removed. If the task cannot be found, a flag is set and the code returns a print statement to the user stating “Task not found.” Note that this is the only instance in which there is “presentation” code in the non-presentation or main-body sections. This was done for simplicity. It would be possible to return the flag parameter from the function and apply logic in the main body of code to determine if the task was found. However, to minimize the changes necessary for this code to run properly, this was not done. This function otherwise returns only the *list_of_rows* parameter, regardless of it is modified.

The final function in this section writes the list of data to a text file with comma-separated formatting. This function requires a stripped string for the file name (including the extension) and will save it to the local directory that the code is running from, in the case of code execution via IDE. If the code is run from a command console, the file would attempt to save to the active directory, though an empty “ToDoFile.txt” file must be supplied in this directory as well. After finishing the file-write process, the code closes the file object and returns the *list_of_rows* parameter to the main code.

INPUT/OUTPUT

This section of code defines all of the I/O necessary to achieve the functionality of the code, mostly consisting of print and input statements. These functions tend to be simpler than the previous section.

The first function is simply a menu display option. Whenever this function is called, the 4-option menu will print to the screen for the user to view. This is done repetitively such that the user can use multiple menu options in quick succession. This function has no return parameters.

The second function grabs the user input for the menu option. This is stored in the *choice* parameter, which is returned to the main body of the script.

The third function displays the current tasks and priorities stored in the *list_of_rows* parameter. This is the only function in this section that takes an argument from the main script. The function loops through the *list_of_rows* parameter and neatly prints the lines (consisting of tasks and priorities) to the display window for the user. This function has no return parameters.

The fourth function grabs user inputs for a new *task* and *priority*. These are stripped strings that are returned to the main body of the code.

The final function in the script grabs user input for a task to remove from the existing list. This is returned to the main body of the code as a stripped string via the *remove_task* parameter.

MAIN BODY

The main body of the script starts by calling the *read_data_from_file* function in the *Processor* class, supplying the named arguments *file_name_str* and *table_lst* (“ToDoFile.txt” and an empty list, respectively). The return list is saved in *table_lst*.

The rest of the main body of the code is wrapped in an infinite while loop, such that the user may choose to perform multiple menu actions until they decide to purposefully exit the program via menu option 4. The top of this loop displays the current task list, displays the menu, and grabs the user’s input for menu choice via three I/O functions (*output_current_tasks_in_list*, *output_menu_tasks*, *input_menu_choice*). The user’s choice is stored in the *choice_str* variable.

From here, logic begins to access the menu options. If the user selects option 1, *IO.input_new_task_and_priority* is called to grab user inputs for a new task and priority, which are conveniently stored in the variables *task* and *priority*. These, along with the existing *table_lst* variable, are supplied to *Processor.add_data_to_list*, which returns an updated list that is then stored in *table_lst*. The code then returns to the menu.

Selecting option 2 allows the user to remove an existing task. User input for the task to remove is grabbed via *IO.input_task_to_remove*, which is then stored in *task*. This and the existing list of data are supplied to *Processor.remove_data_from_list*, which performs its task and returns an updated list of data that is stored in *table_lst*. The code then returns to the main menu.

Option 3 allows the user to save the current data stored in the *table_lst* variable. This list of data and the *file_name_str* variable are supplied to *Processor.write_data_to_file* to write the output to the named file. The function returns a *list_of_rows* parameter, which is stored in *table_lst*, though there should not be any updates to this variable by calling this function.

Option 4 prints a “Goodbye!” message to the user and then breaks from the infinite loop, ending the program. Figures 2 through 7 show the code operating in both PyCharm and the Windows Command console.

SUMMARY

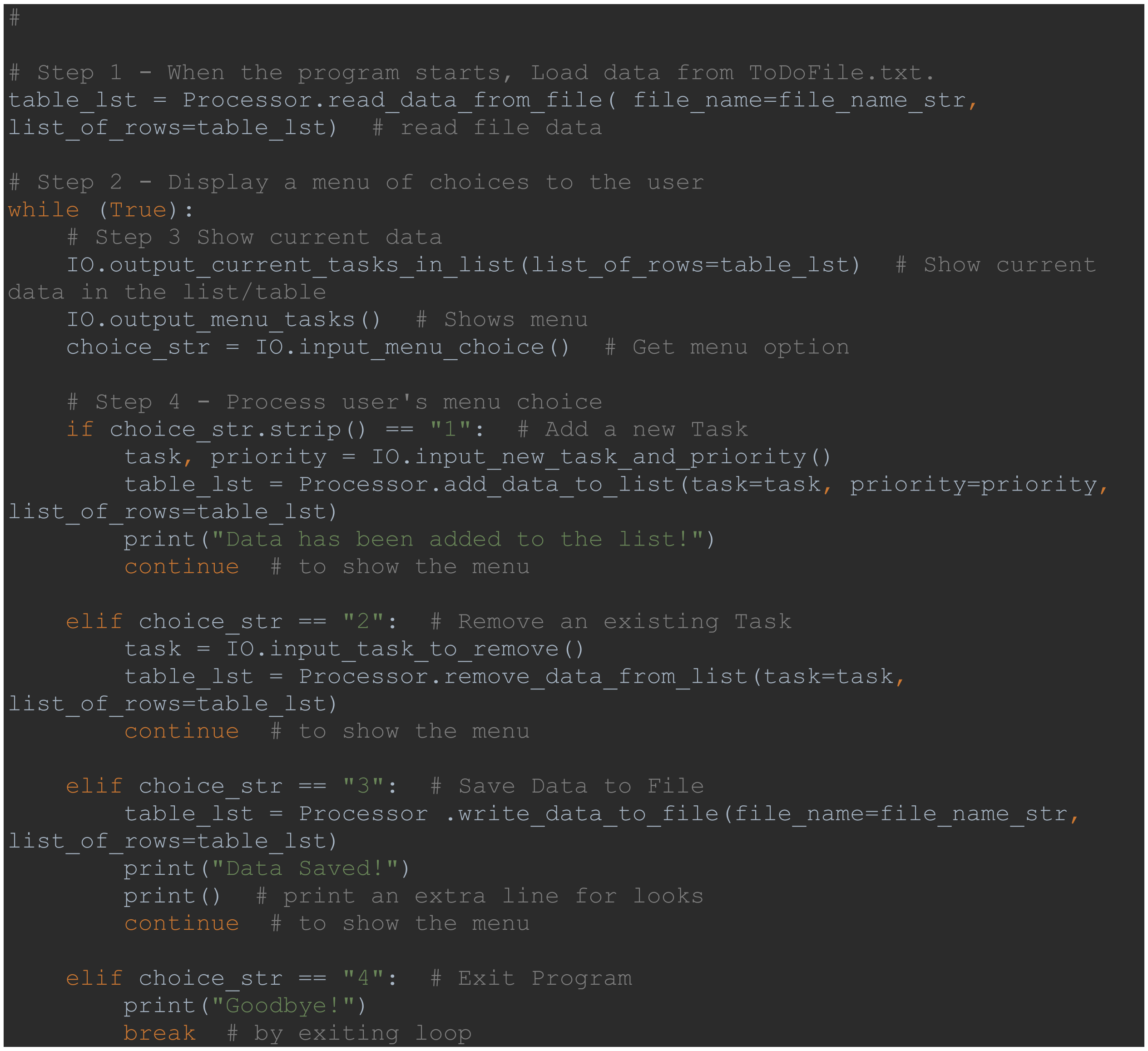
In summary, this document describes the code behind a successful “to-do list” manager program that takes user inputs for a to-do list and each item’s priority and saves it to a “ToDoFile.txt” file. This is done via functions instead of only a main body of script, which was done in Assignment 05.

Functions are split into two categories (“Processing” and “I/O”) based on their general functionality. These functions are then called in a main body of code to perform the desired task (in this case, adding to, removing from, or saving tasks to a text file in a comma-separated format).

The main body of this script is much cleaner than the previous week and shows the utility of creating functions for code blocks that will be run repetitively.

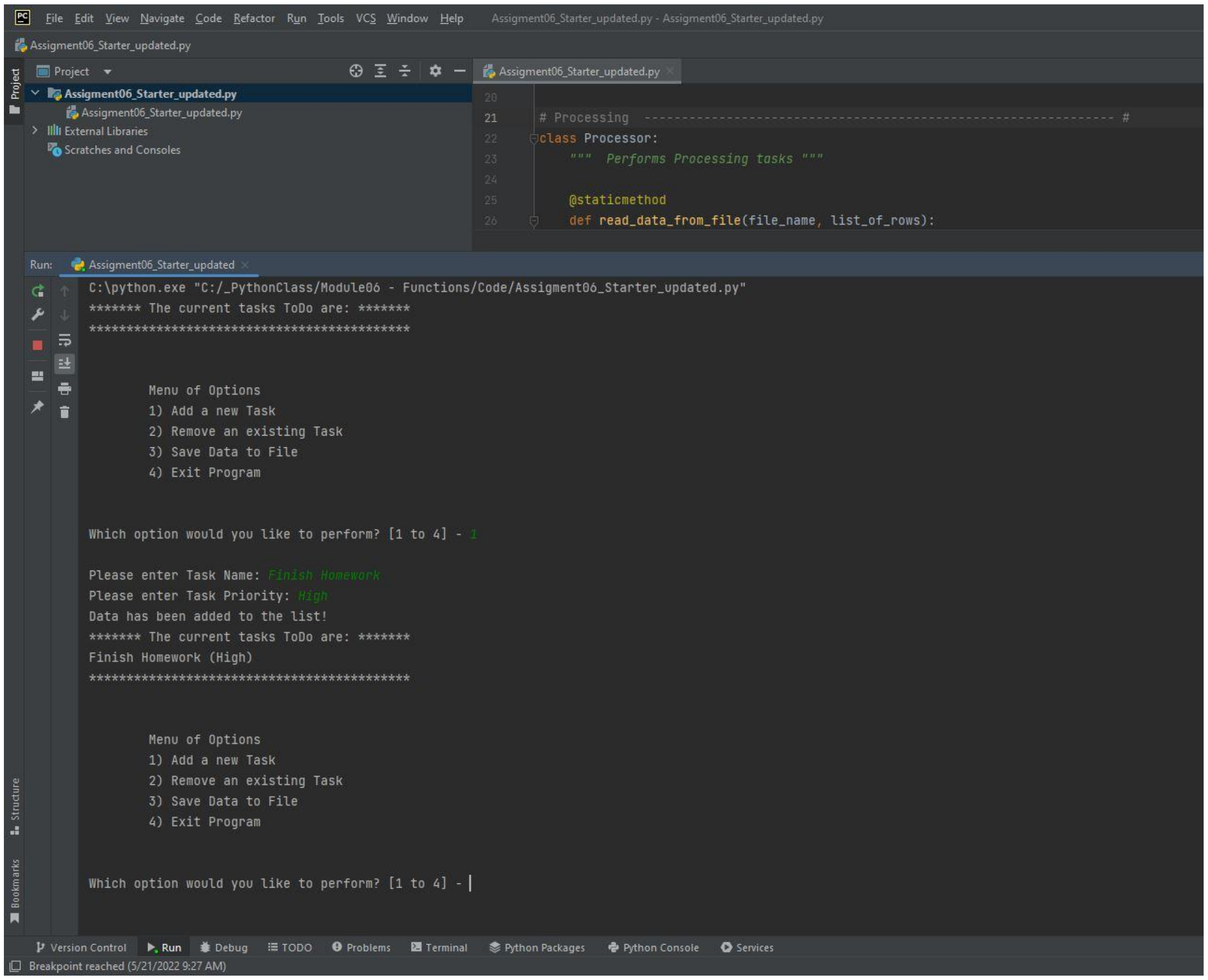
```
# ---------------------------------------------------------------------------
- #
# Title: Assignment 06
# Description: Working with functions in a class,
#              When the program starts, load each "row" of data
#              in "ToDoToDoList.txt" into a python Dictionary.
#              Add the each dictionary "row" to a python list "table"
# ChangeLog (Who,When,What):
# RRoot,1.1.2030,Created started script
# NSandhu,5.21.2022,Modified code to complete assignment 06
# ---------------------------------------------------------------------------
- #

# Data ---------------------------------------------------------------------
#
# Declare variables and constants
file_name_str = "ToDoFile.txt"  # The name of the data file - NOTE THAT A
BLANK "TODOFILE.TXT" MUST BE SUPPLIED
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
        """
        i = 0
        flag = 0
        for data in list_of_rows:
            if data["Task"].lower() == task.lower().strip():
                print(f"The following Task has been removed: {data['Task']}")
                print()  # print new line for looks
                flag = 1
                i += 1
                list_of_rows.remove(data)
            else:
                i += 1
                if i == len(list_of_rows) and flag == 0:
                    print("Task not found.")
                    print()  # print new line for looks
        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
        """
        toDoList = open(file_name, "w+")
        for data in list_of_rows:
            toDoList.write(str(data["Task"]) + "," + str(data["Priority"]) +
"\n")
        toDoList.close()
        return list_of_rows


# 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 = input("Please enter Task Name: ").strip()
        priority = input("Please enter Task Priority: ").strip()
        return task, priority

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

        :return: (string) with task
        """
        remove_task = input("Please enter name of Task to remove from list:
").strip()
        return remove_task


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



```
#

# Step 1 - When the program starts, Load data from ToDoFile.txt.
table_lst = 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)
        print("Data has been added to the list!")
        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()  # print an extra line for looks
        continue  # to show the menu

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

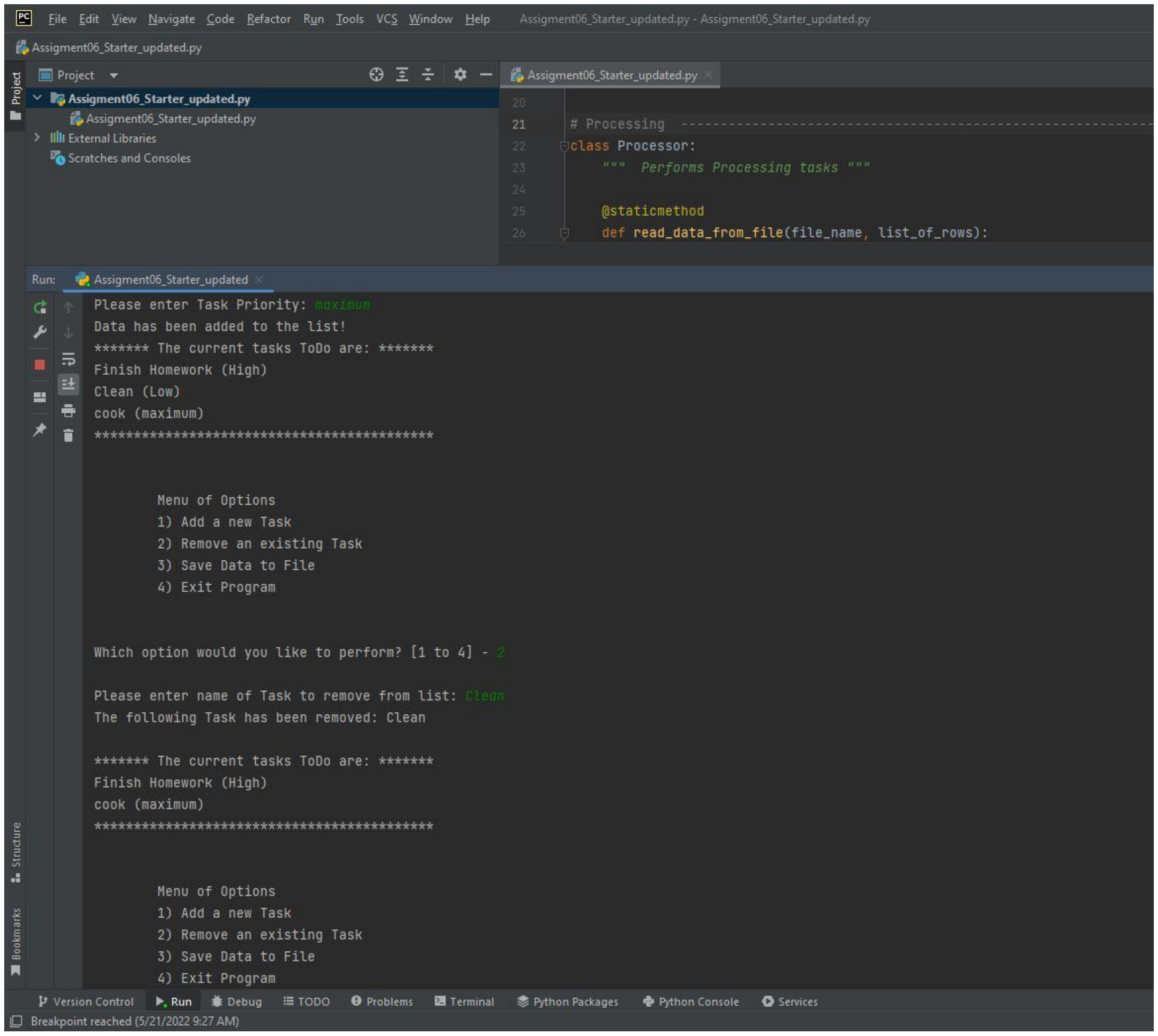
Fig. 1 – Code written by both R. Root and N. Sandhu for creation of the to-do list manager program.


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Assigment06_Starter_updated.py - Assigment06_Starter_updated.py
Project
Assigment06_Starter_updated.py
External Libraries
Scratches and Consoles
Run: Assigment06_Starter_updated
Version Control
Run
Debug
TODO
Problems
Terminal
Python Packages
Python Console
Services
Breakpoint reached (5/21/2022 9:27 AM)

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

    @staticmethod
    def read_data_from_file(file_name, list_of_rows):
```

```
C:\python.exe "C:/_PythonClass/Module06 - Functions/Code/Assigment06_Starter_updated.py"
******* 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

Please enter Task Name: Finish Homework
Please enter Task Priority: High
Data has been added to the list!
******* The current tasks ToDo are: *******
Finish Homework (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] -
```

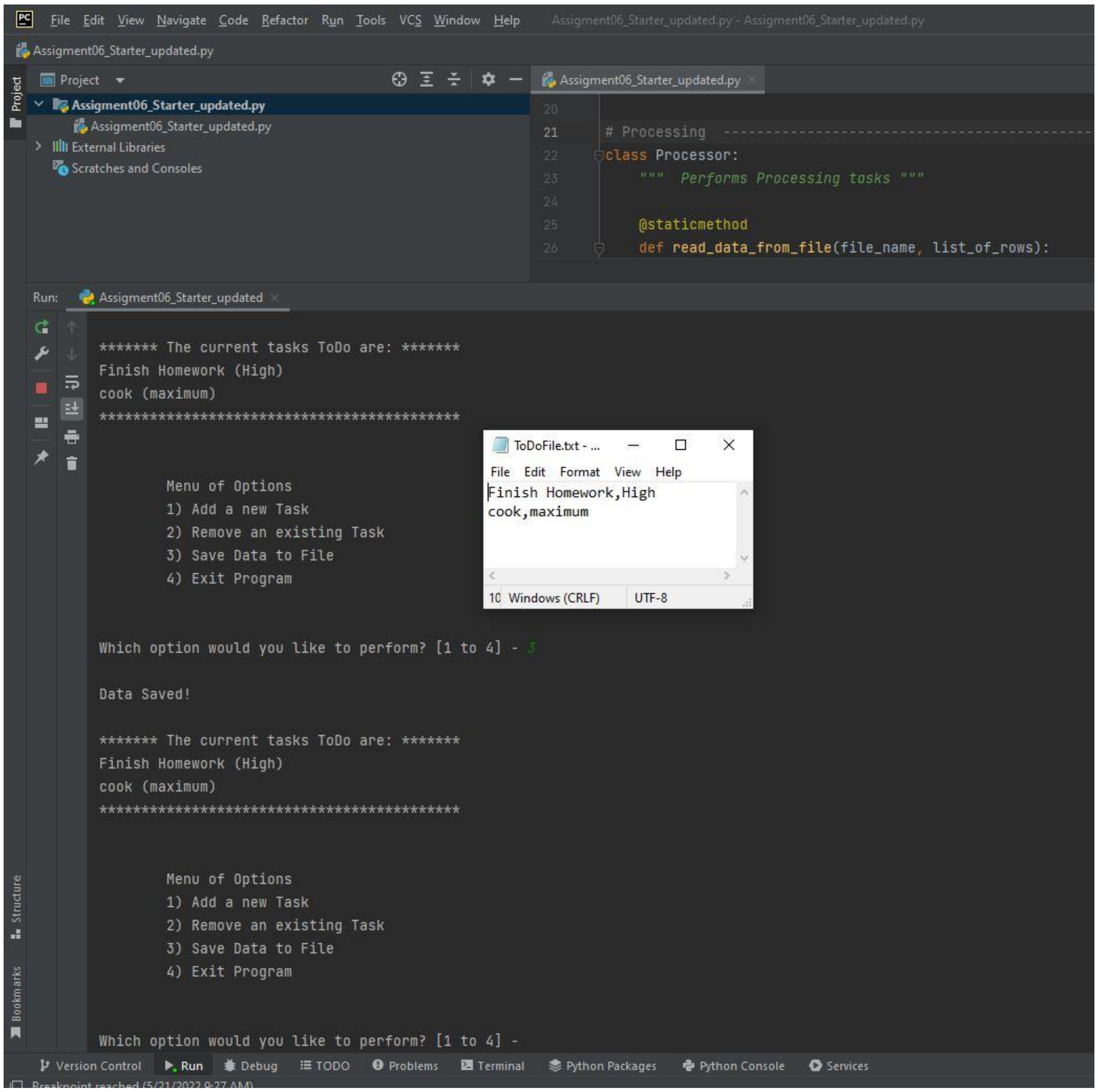
Fig. 2. – A task being added using PyCharm.


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Assigment06_Starter_updated.py - Assigment06_Starter_updated.py
Project
Assigment06_Starter_updated.py
External Libraries
Scratches and Consoles
Run: Assigment06_Starter_updated
Version Control Run Debug TODO Problems Terminal Python Packages Python Console Services
Breakpoint reached (5/21/2022 9:27 AM)

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

    @staticmethod
    def read_data_from_file(file_name, list_of_rows):
```

```
Please enter Task Priority: maximum
Data has been added to the list!
******* The current tasks ToDo are: *******
Finish Homework (High)
Clean (Low)
cook (maximum)
*******************************************



        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 name of Task to remove from list: Clean
The following Task has been removed: Clean

******* The current tasks ToDo are: *******
Finish Homework (High)
cook (maximum)
*******************************************



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

Fig. 3. – A continuation of Fig. 2. A task is removed using PyCharm. Note that several other tasks were added prior to removing a task.

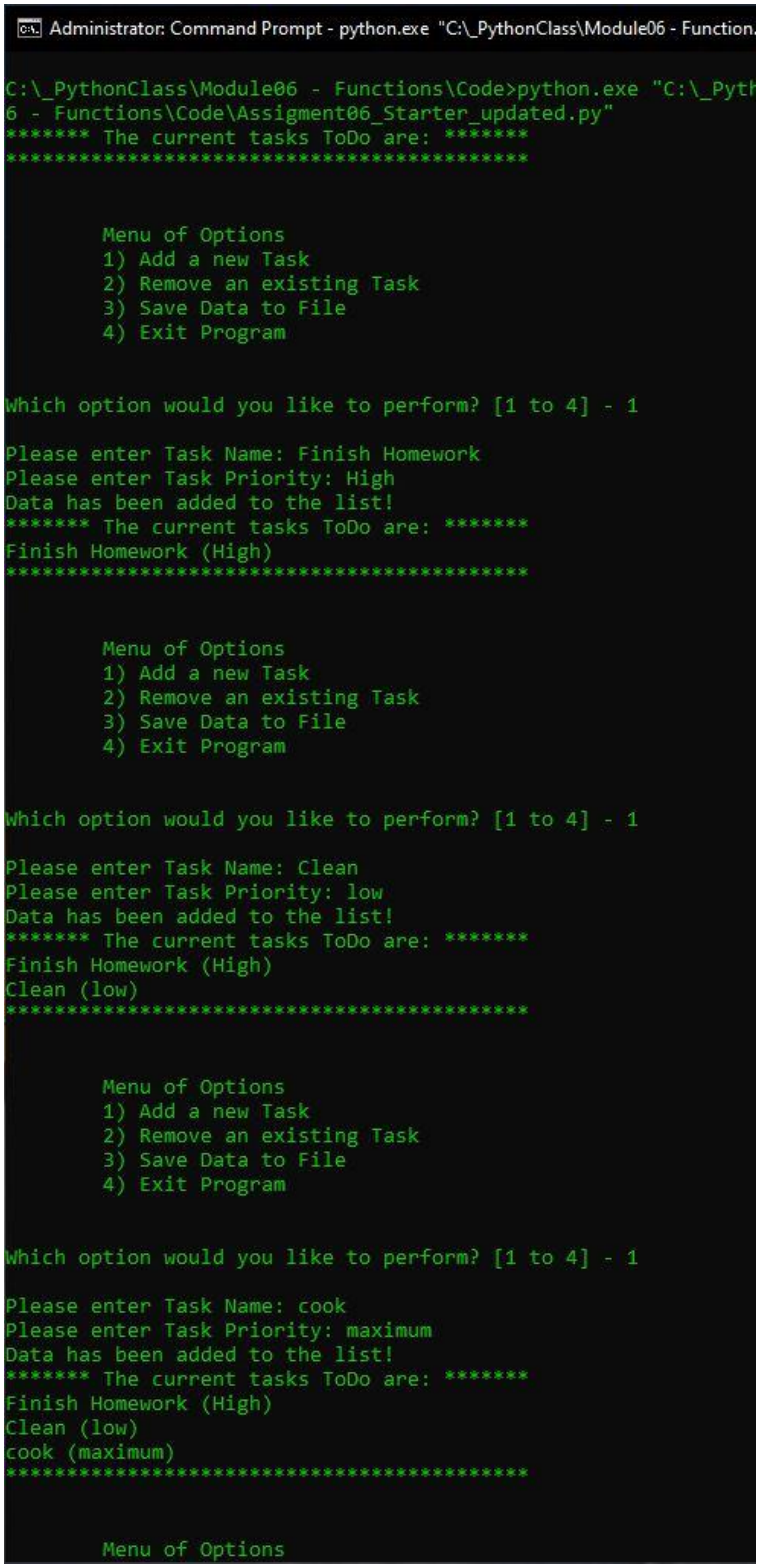

File Edit View Navigate Code Refactor Run Tools VCS Window Help
Assigment06_Starter_updated.py - Assigment06_Starter_updated.py
Project
Assigment06_Starter_updated.py
External Libraries
Scratches and Consoles
Run: Assigment06_Starter_updated
ToDoFile.txt - ...
File Edit Format View Help
Windows (CRLF) UTF-8
Version Control Run Debug TODO Problems Terminal Python Packages Python Console Services
Breakpoint reached (5/21/2022 9:27 AM)

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

    @staticmethod
    def read_data_from_file(file_name, list_of_rows):
```

```
******* The current tasks ToDo are: *******
Finish Homework (High)
cook (maximum)
*******************************************



        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: *******
Finish Homework (High)
cook (maximum)
*******************************************



        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] -
```

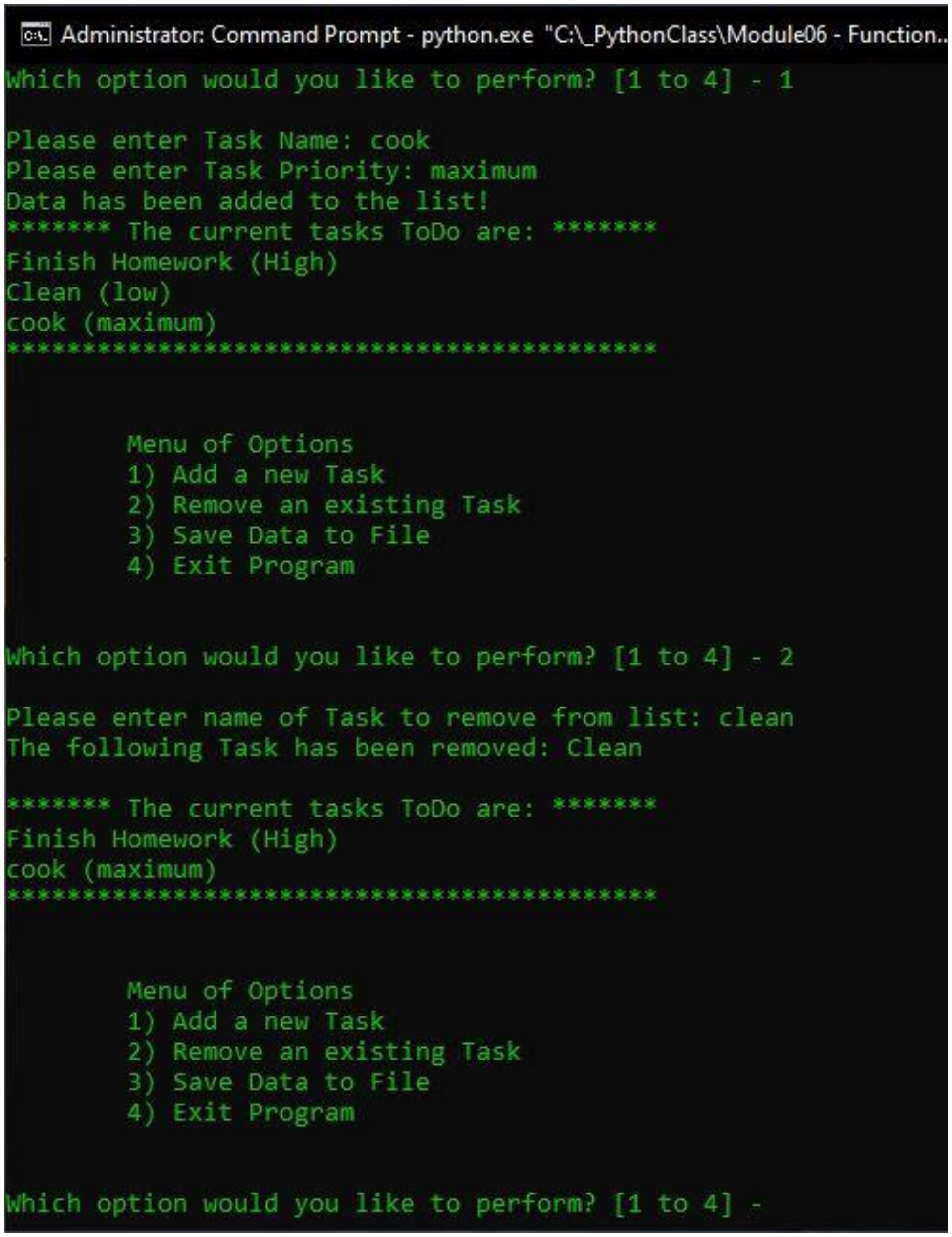
```
Finish Homework,High
cook,maximum
```

Fig. 4. – A continuation of Fig. 3. The to-do list is saved using PyCharm. The resulting text file is also shown.



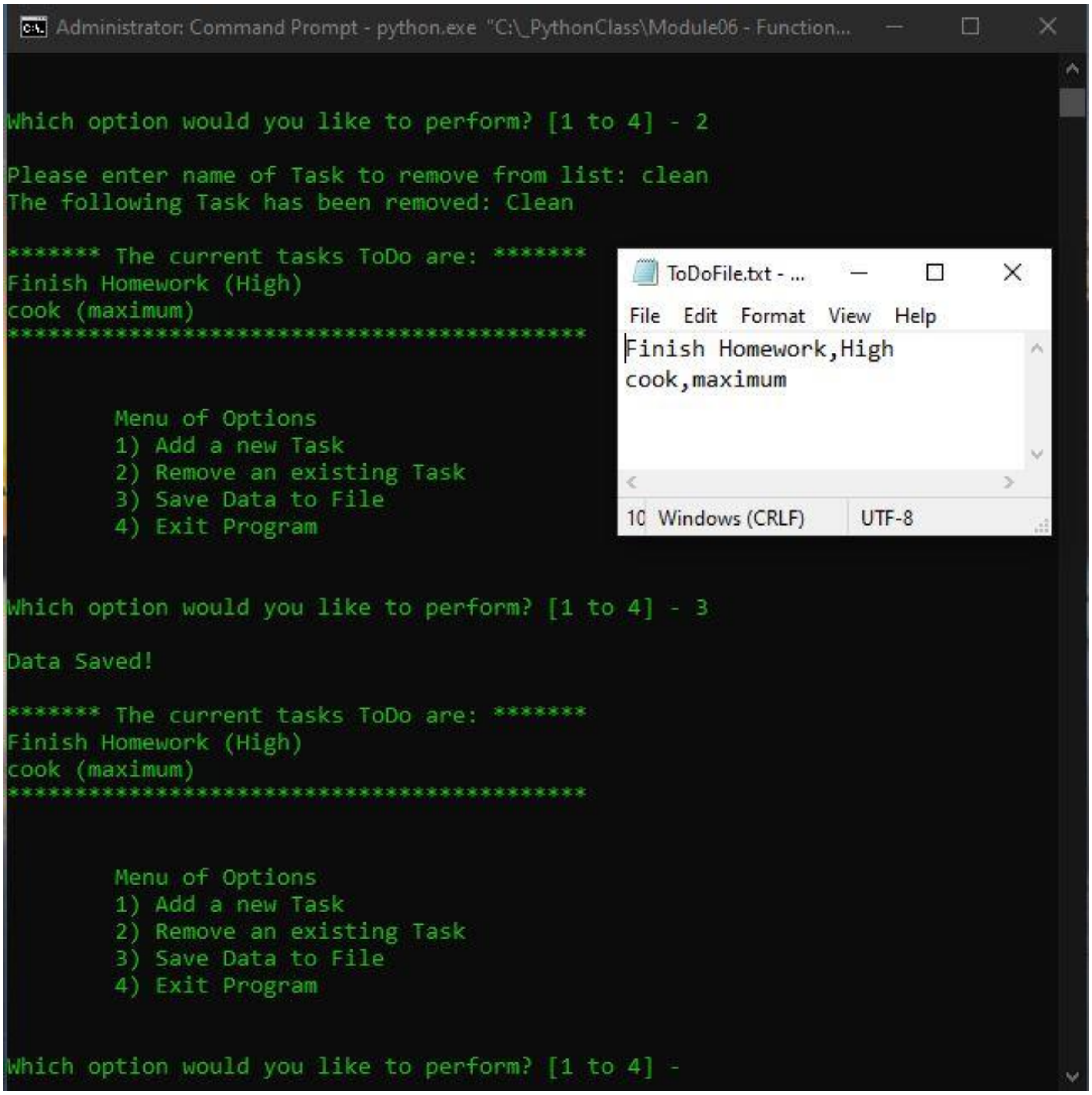
```
Administrator: Command Prompt - python.exe  "C:\_PythonClass\Module06 - Function

C:\_PythonClass\Module06 - Functions\Code>python.exe "C:\_Pyth
6 - Functions\Code\Assigment06_Starter_updated.py"
******* 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

Please enter Task Name: Finish Homework
Please enter Task Priority: High
Data has been added to the list!
******* The current tasks ToDo are: *******
Finish Homework (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

Please enter Task Name: Clean
Please enter Task Priority: low
Data has been added to the list!
******* The current tasks ToDo are: *******
Finish Homework (High)
Clean (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

Please enter Task Name: cook
Please enter Task Priority: maximum
Data has been added to the list!
******* The current tasks ToDo are: *******
Finish Homework (High)
Clean (low)
cook (maximum)
*******************************************


        Menu of Options
```

Fig. 5. – Several tasks being added via the Windows Command Console.



```
Administrator: Command Prompt - python.exe  "C:\_PythonClass\Module06 - Function..
Which option would you like to perform? [1 to 4] - 1

Please enter Task Name: cook
Please enter Task Priority: maximum
Data has been added to the list!
******* The current tasks ToDo are: *******
Finish Homework (High)
Clean (low)
cook (maximum)
*******************************************



        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 name of Task to remove from list: clean
The following Task has been removed: Clean

******* The current tasks ToDo are: *******
Finish Homework (High)
cook (maximum)
*******************************************



        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] -
```

Fig. 6. – A continuation of Fig. 5. A task being removed via the Windows Command Console.


Administrator: Command Prompt - python.exe "C:_PythonClass\Module06 - Function...
ToDoFile.txt - ...
File Edit Format View Help
10 Windows (CRLF) UTF-8

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

Please enter name of Task to remove from list: clean
The following Task has been removed: Clean

******* The current tasks ToDo are: *******
Finish Homework (High)
cook (maximum)
*******************************************


        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: *******
Finish Homework (High)
cook (maximum)
*******************************************


        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] -
```

```
Finish Homework,High
cook,maximum
```

Fig. 7. - A continuation of Fig. 6. The to-do list is saved using the Windows Command Console. The resulting text file is also shown.