Hwai-Jiang Jong

May 17, 2021

Foundations of Programming: Python

Assignment 06

The To-Do List Script

# Introduction

In this report, I am documenting what I learned in Module 6 and the steps I took to create a script to perform Assignment06. It contains three main sections (excluding this introduction):

1. What I learned in Module 6;
2. Developing a script in PyCharm for Assignment06;
3. How I created a GitHub webpage
4. Summary

# What I Learned in Module 6

This section documents my learning from reading and watching the materials as below

1. Course video of module 6
2. Textbook: Chapter 6
3. <http://www.learnpython.org/en/Functions> (external site)
4. <https://youtu.be/qO4ZN5uZSVg> (external site)

The key learning is summarized in response to the questions listed in this assignment.

* A function is a collection of statements to perform a specific task that is often reusable for multiple programs or programmers.
* Not only very useful for organizing the large programs, functions can save development time when a task is performed very often and commonly used.
* A function can receive variables from another function (or main program) which calls it, perform some actions, and then return the results back to the calling function or main program.
* When a variable is listed in a pair of parentheses in the function definition, it is a parameter. When it is listed in a pair of parentheses of a function where it gets called, it is an argument.
* Local variables are encapsulated in a function and only visible to that function. On the contrary, global variables are outside of the functions and can be visible to (but not changeable by) all functions.
* Although a function can run without them, Docstrings in a function definition are very important in making the function understandable. Docstrings contain information of the function’s purpose, its parameters, and its return variables.
* A group of functions can be defined under a class, in which they are collectively for a specific “area of concern”. A class is a collector of functions that achieves a “Separation of Concerns” pattern in a program, especially when the code becomes very large.
* The debugging tools in PyCharm is very useful when the programmer needs insight to the internal working of the code. The programmer can choose the location in the code to pause in the middle of the run. The “step over”, “step into”, and “step out” functions in the debugger allow the programmer to analyze the behavior of the code and help spot the error.
* A GitHub webpage is a webpage designed to host my projects form my GitHub repository. It can be set up by creating an *index.md* (markdown) file from within the repository.

# Writing To-Do List Script

## Script Construct Planning

Like assignment05, the script would load the existing data from “ToDoList.txt” on the hard drive into a list table and show current data, then it asks users to choose between five different options concerning the data. The five options are

1. Add a new item of task and priority
2. Remove an existing item of task and priority
3. Save the updated data into the file
4. Reload the file
5. Exit the program

A pseudo code is written as the following.

*Open a file “ToDoList.txt” and read the data into a list table*

*Loop {*

*Show the current data to user*

*Print the menu of options for user to select*

*If option 1 is selected: Add a new item of task and priority to the table*

*If option 2 is selected: Remove an item of task and priority from the table*

*If option 3 is selected: Save the data into the file and close the file*

*If option 4 is selected: Reload the file data to the table*

*If option 5 is selected: Exit the program*

*}*

## Writing the Script in PyCharm

I created a new project folder “C:\\_PythonClass\Assignment06\”. I loaded the template “Assignment06\_starter.py” into PyCharm and modified it to fulfill the assignment. The final script was then saved as “Assignment06.py” for submission.

This script was completed at incremental steps. I started with a text file “ToDoList.txt” in the working folder, with two items of task and priority. In the code, I created the variable definition to include all variables needed in this code. Then the data in the “ToDoList.txt” were loaded into a list table. The five options with their actions were implemented in a while loop based on their specific actions on the data. The script with its various sections is described in detail as follows.

1. Change History: Figure 1 shows the steps I took to make this script perform as required.

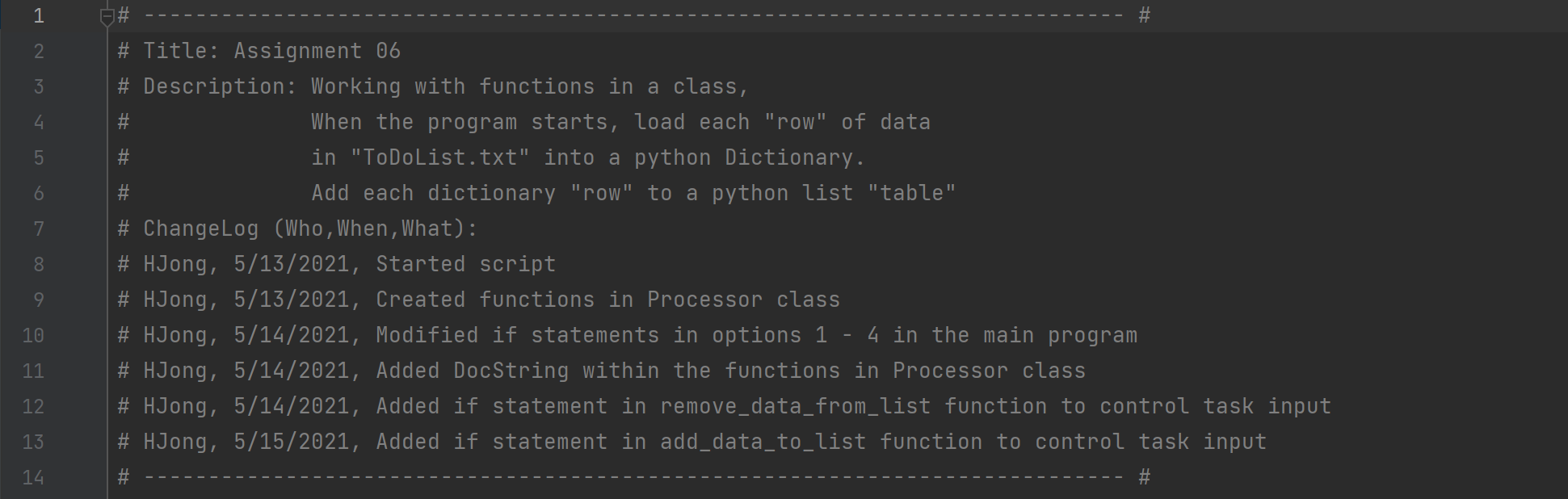


Figure Change Log of the Script

1. Data definition: Figure 2 shows the variables I used to perform tasks on the data.

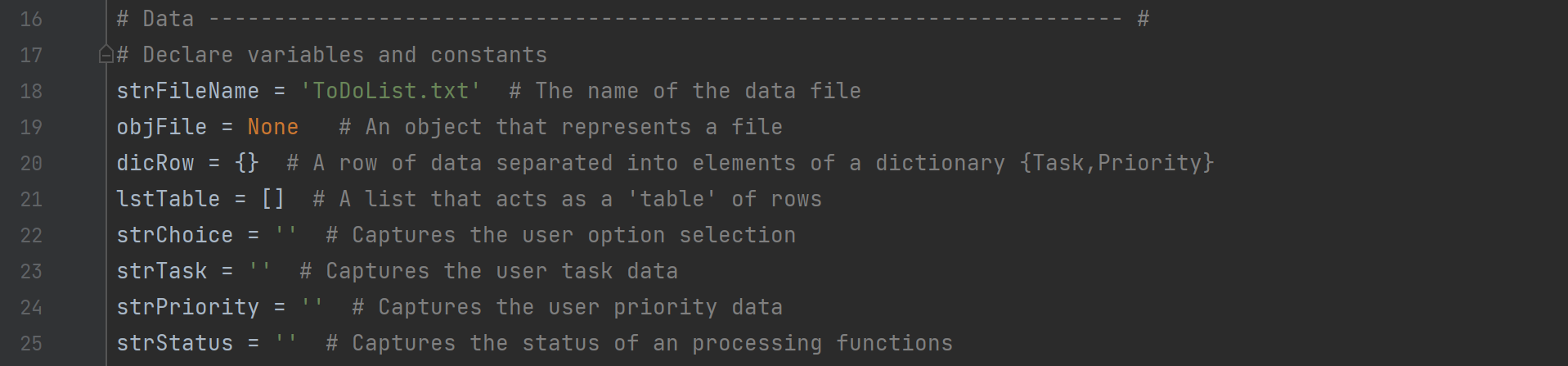


Figure Data Declaration Section of the Script

1. Data Processing – the *read\_data\_from\_file()* function: The data are loaded into a list table from the data file. See Figure 3.

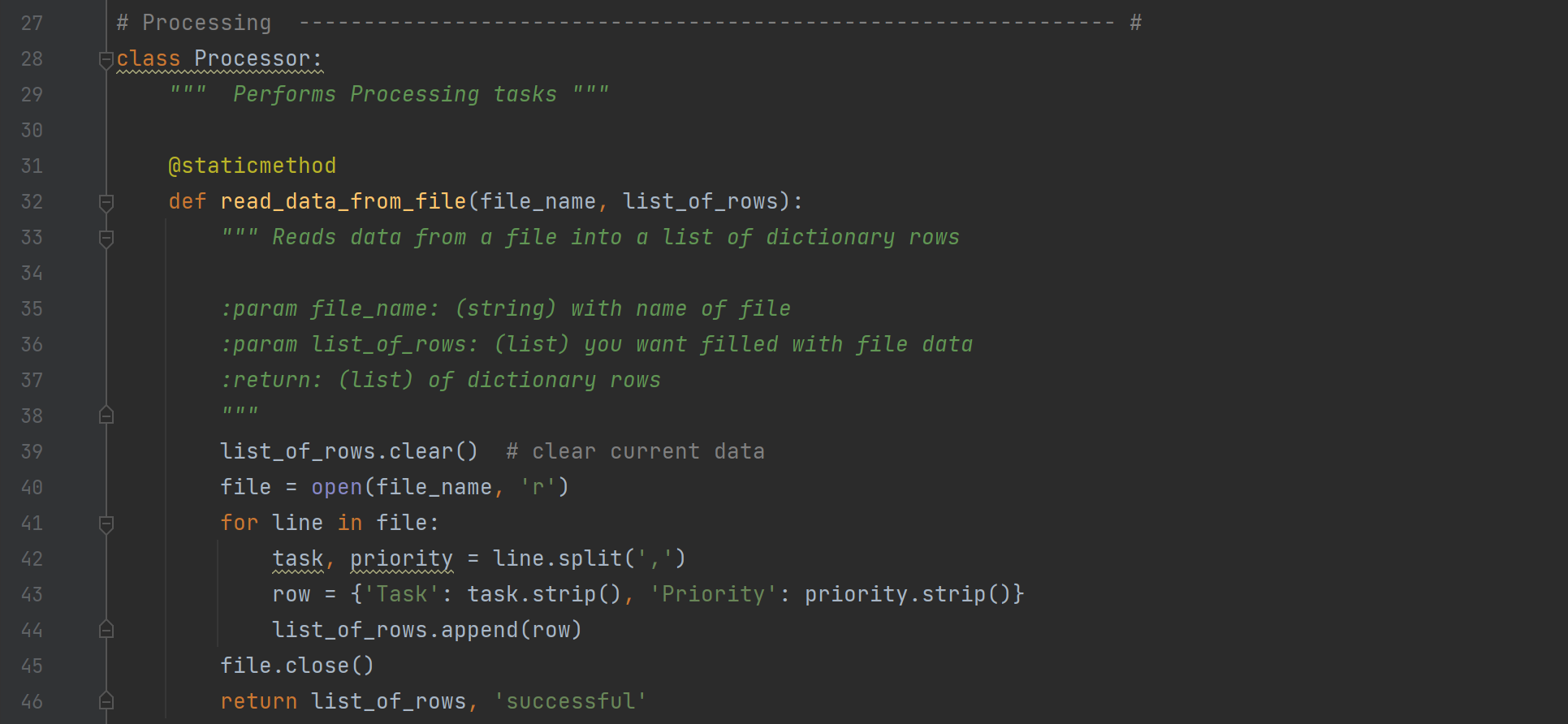


Figure Loading Data from File to Memory (a List Table)

1. Data Processing – the *add\_data\_to\_list()* function: As shown in Figure 4, I added an if statement to control the entered task against the existing task in the list table. If the entered task exists, then the code asks if the user wants to replace the existing one or cancel the action. This prevents the existing task from being accidentally overwritten. At the end, the updated list table and action status are returned.

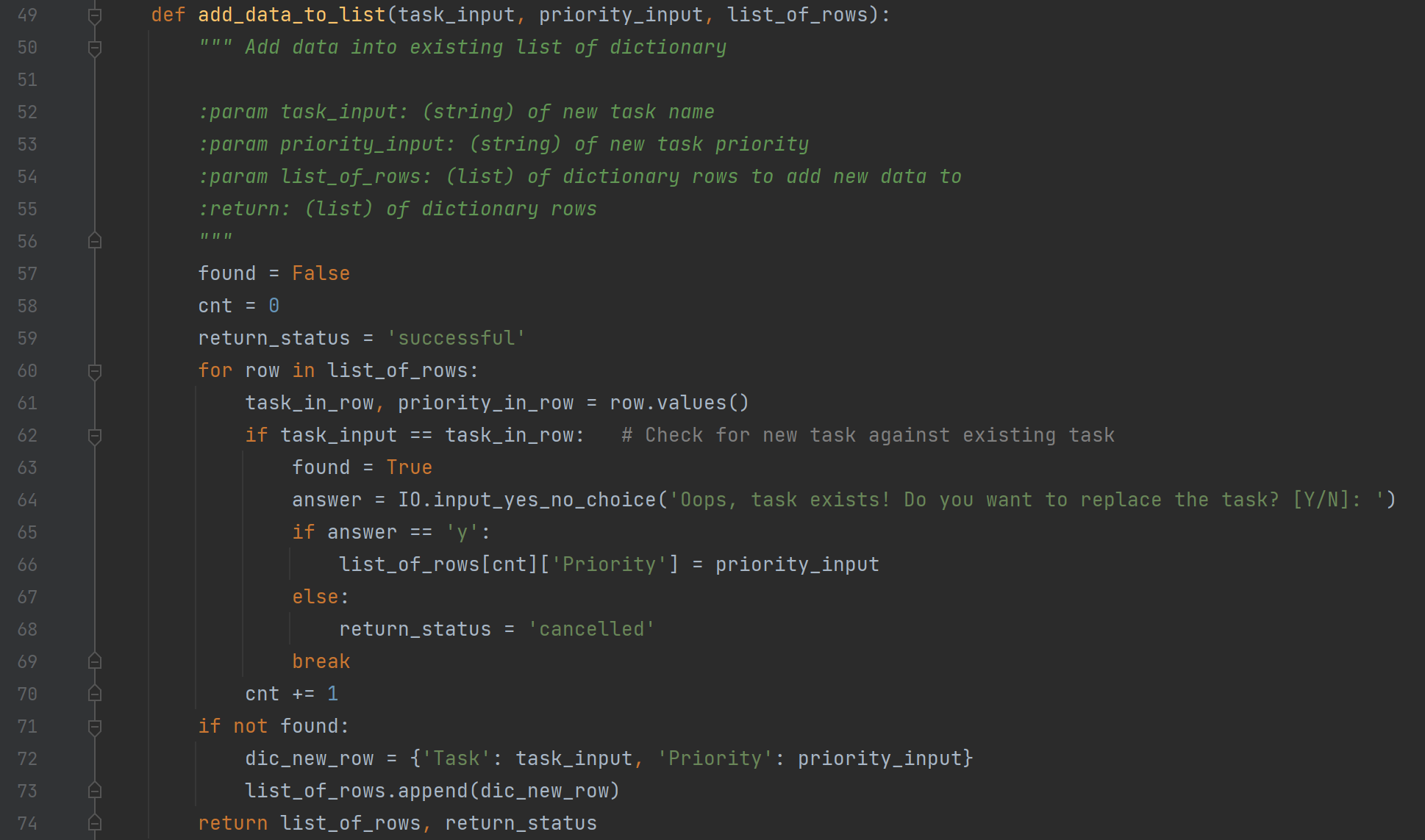


Figure Displaying the Menu of Options and Asking for User Selection

1. Data Processing – The *remove\_data\_from\_list()* Function: In this function, I added an *if* statement to check the user input against the exiting tasks in the list table. If the task does not exist, the function warns the user and asks for another task until the user choses to cancel. The *IO.input\_yes\_no\_choice()* is called to perform the task input if the user chooses to re-enter. At the end of the function, the updated list table and *return\_status* variable (string) are returned based on the removing being successful or cancelled. See Figure 5 for details.

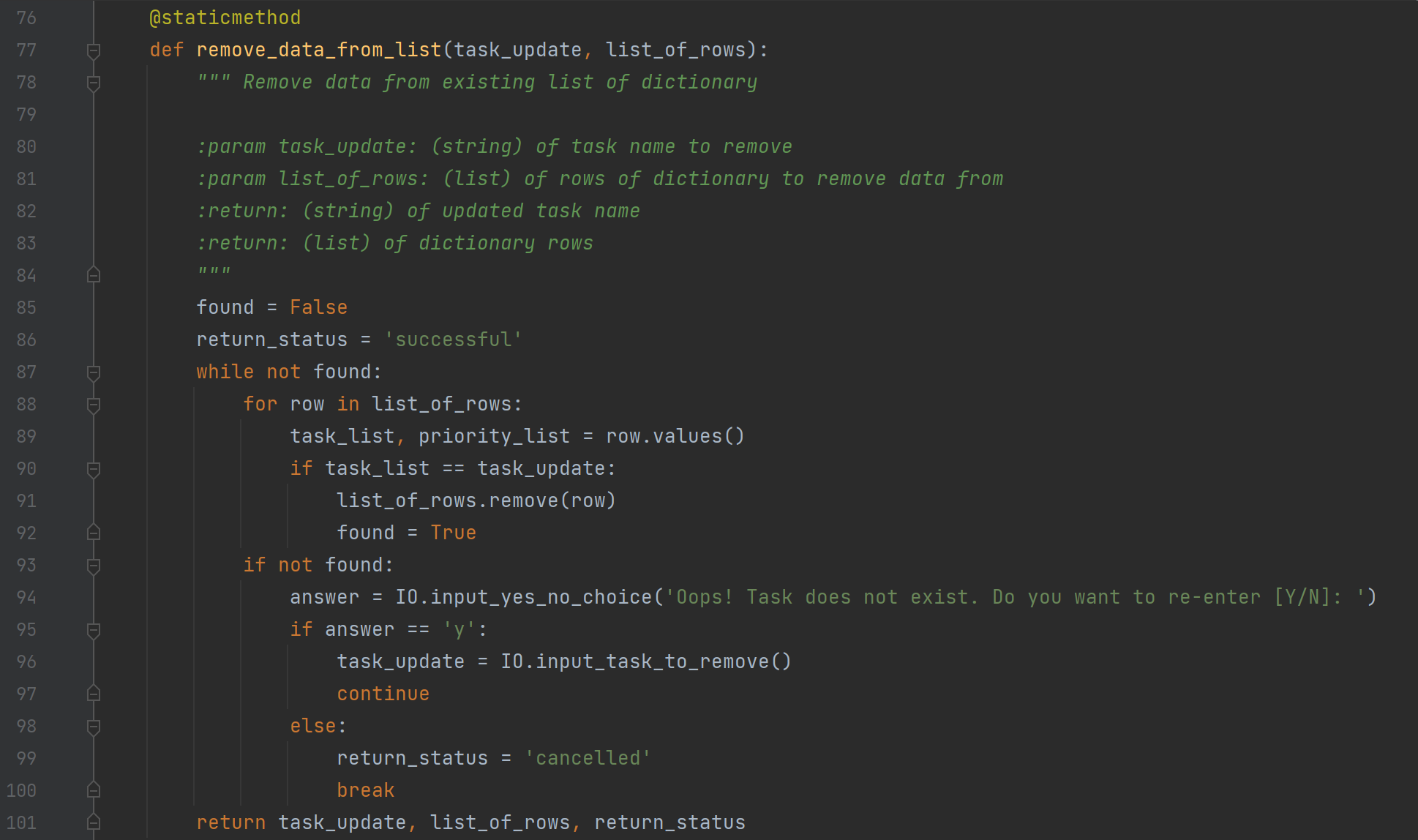


Figure The remove\_data\_from\_list() Function

1. Data Processing – The *write\_data\_to\_file()* function: This function writes current data in the list table into “ToDoList.txt”, then it returns the list table and the status of this action which is set as “successful”. The code is shown in Figure 6.

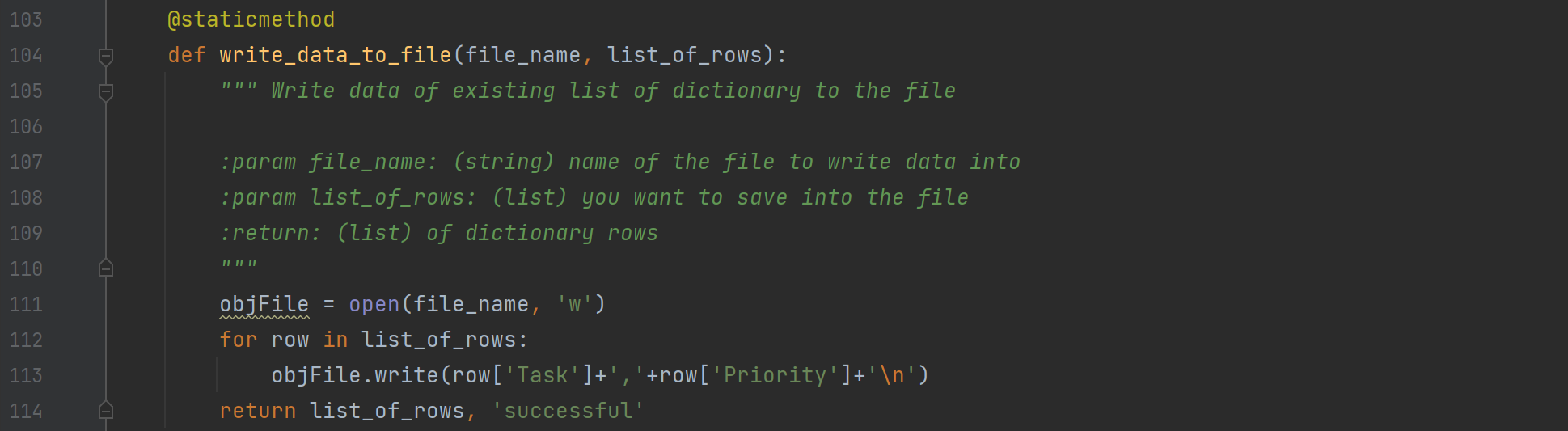


Figure The write\_data\_to\_file() Function

1. Presentation – In the presentation section, an IO class is composed of the following functions.

* *print\_menu\_Tasks()*
* *input\_menu\_choice()*
* *print\_current\_Tasks\_in\_list()*
* *input\_yes\_no\_choice()*
* *input\_press\_to\_continue()*
* *input\_new\_task\_and\_priority()*
* *input\_task\_to\_remove()*

Because these functions are provided and no change was done, no detail is included in this report.

1. Main program – In the main program section, I fulfilled the code to process user menu choice options 1-4, which are related to adding a new task, removing an existing task, saving data to file, and reloading data from file. Figure 7 shows the implementation of the script for options 1-4. Option 5 was provided and not modified, so it is not included in this Figure.

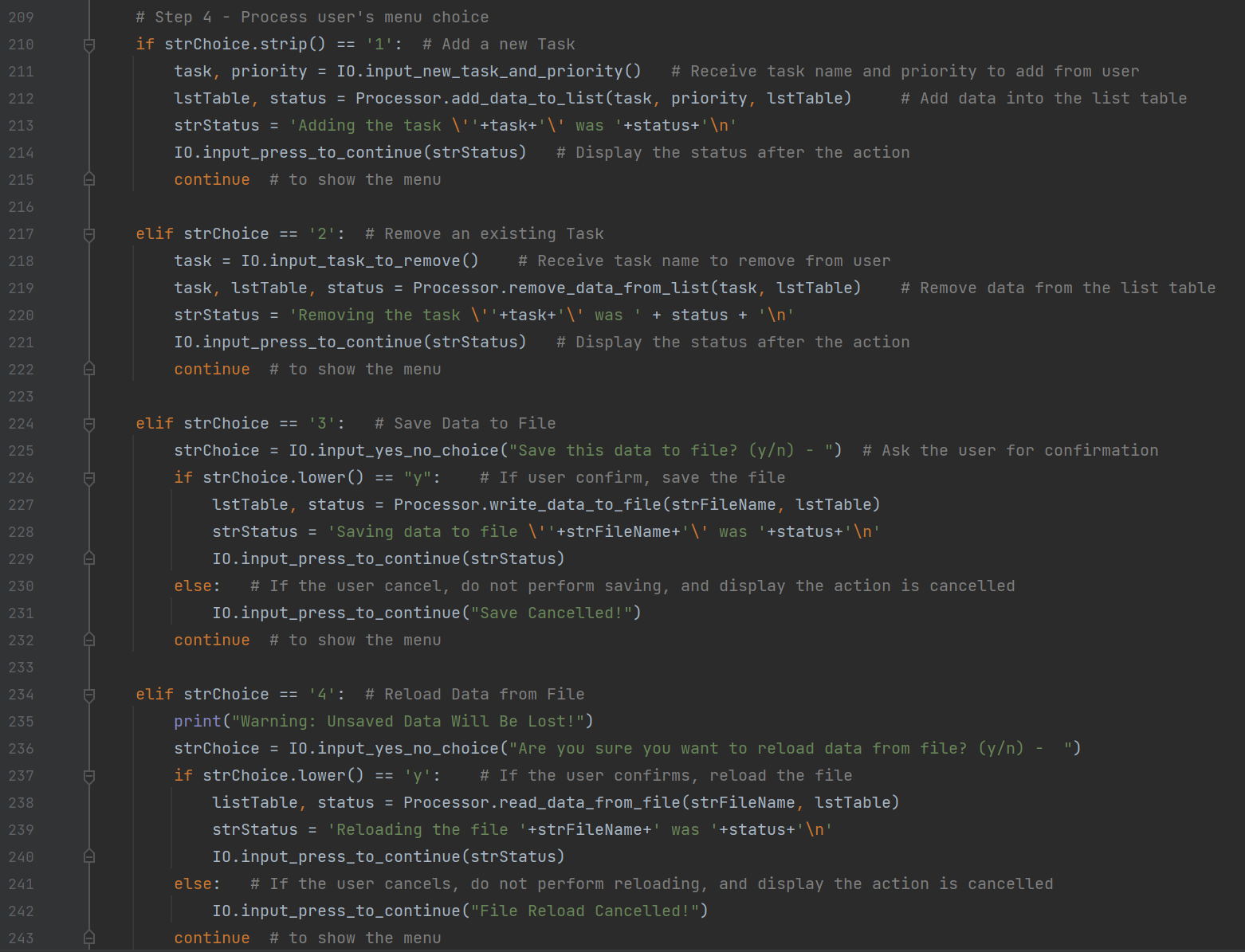


Figure Main Program that Processes User Choice Options

## Executing the Script in PyCharm

In this section, I am demonstrating the script in its performing actions per the user’s choice among the four options shown in Figure 7. Proper functions in the IO and Processor classes were called to perform the requested action by the user. All the following runs were sequentially conducted in one single execution of the script. They are broken down to Run 1 through Run 4 to facilitate the presentation of the process.

### Run 1 – Adding a New Task

In this run, I tested the script against its adding of data. I started with an initial data in “ToDoList.txt” as shown in Figure 8.

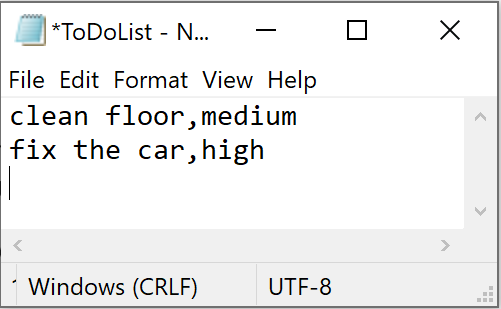


Figure Initial items in ToDoList.txt

Then I added one item of task {‘task’: ’buy grocery’, ‘priority’: ‘high’}. In the “adding” action, I intentionally entered a task that already existed in the list to test if the script captured this and alerted the user. Figure shows the screenshot of this action in PyCharm.

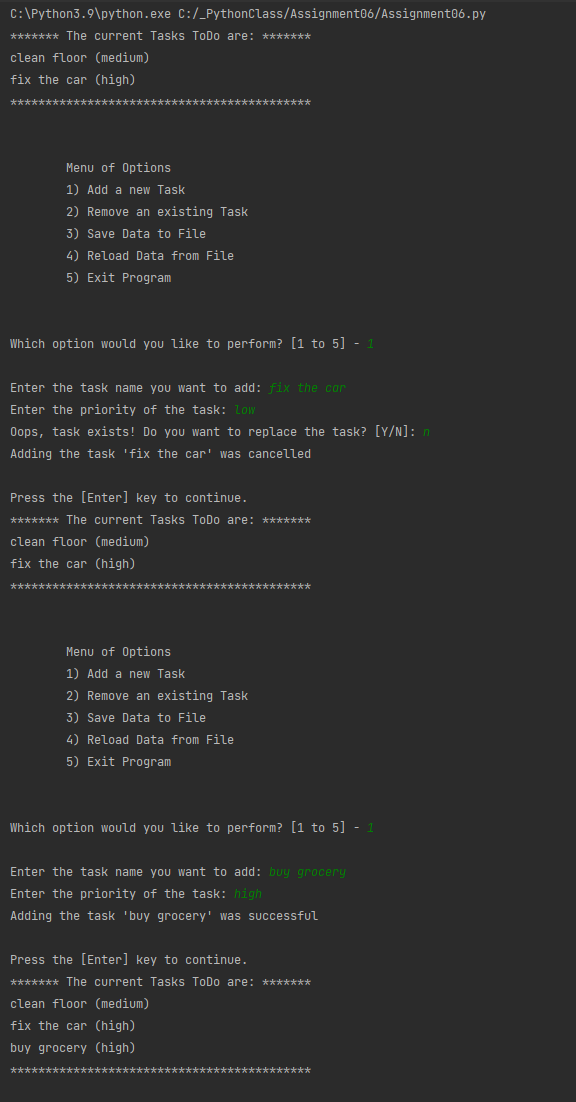


Figure Test Running Option 1: "Adding a New Task"

### Run 2 – Removing an Existing Task

In testing the “removing” action, I removed {‘task’: ’fix the car’, ‘priority’: ‘high’}. I intentionally entered “fix” that did not match the existing tasks in the list, in order to test if the script captured this and alerted the user. Figure 10 shows the screenshot of the results in PyCharm.

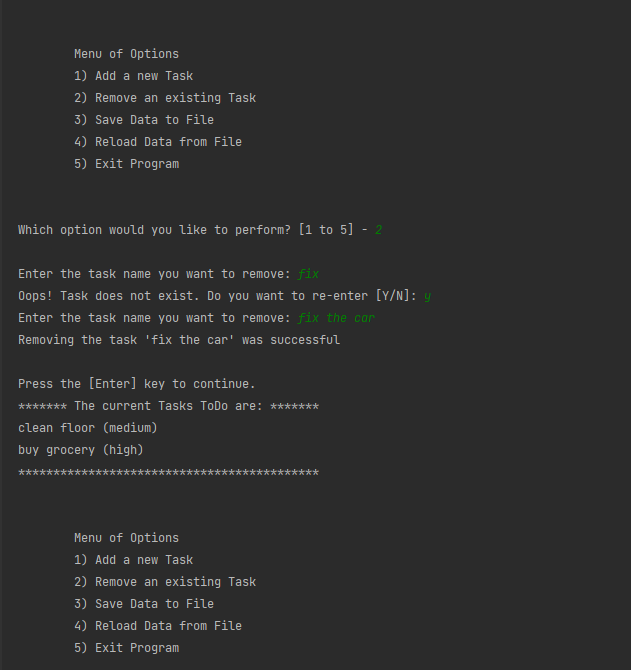


Figure Test Running Option 2: "Removing an Existing Task"

### Run 3 – Saving Data to File

In this run, I saved the data back to the file. I also tested the answer “n” to check if the action was cancelled. Figure 11 is a screenshot of the results in PyCharm.

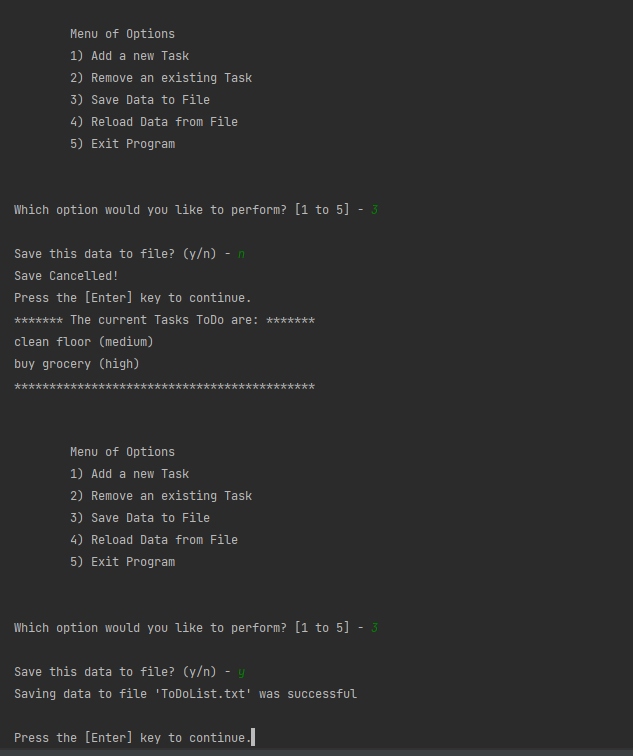


Figure Test Running Option 3: "Saving Data to File"

At this point, I opened “ToDoList.txt” to verify the data were saved as intended. As shown in Figure 12, the data in the file were updated and were the same as Figure 11.

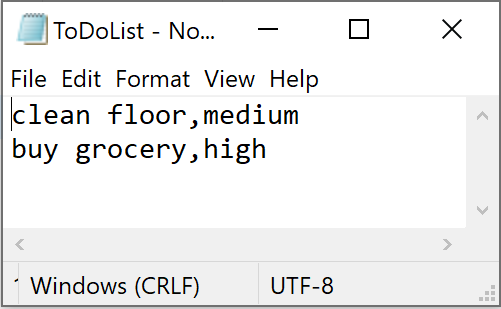


Figure Data in "ToDoList.txt" After the Adding and Removing Data Actions

### Run 4 – Reloading Data from File

In this run, I tested the script against its reloading data from the file “ToDoList.txt”. I first added one new dictionary row {‘task’: ’repair the door’, ‘priority’: ‘low’}, then I reloaded the file to verify the file remained the same without this additional dictionary row. As shown in Figure 13, although the data in the list table showed that the new task “repair the door” was included, the “ToDoList.txt” was not affected when I reloaded it into the memory. The file would not be updated unless option 3 was performed.

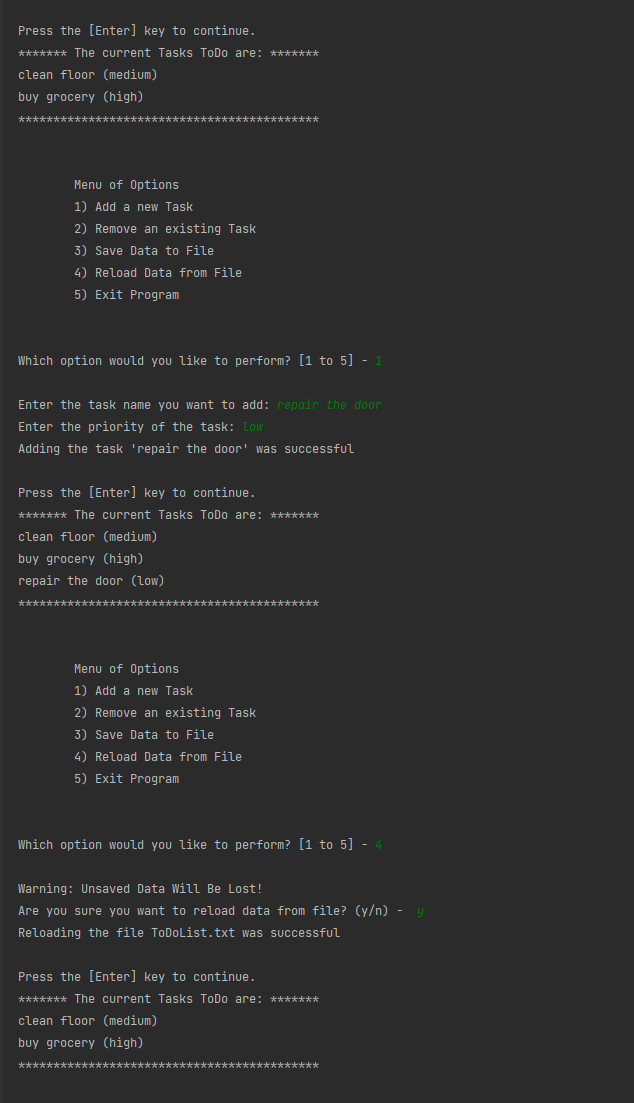


Figure Test Running Option 4: "Reloading Data from File"

## Executing the Script in Windows Command Prompt

### Running Procedure

To continue in Windows command prompt, the initial content of the “ToDoList.txt” was set the same as in Figure 12. Same steps as the PyCharm run were followed, except that I added one item {‘task’: ‘cook dinner’, ‘priority’: ‘high’} and removed {‘task’: ‘clean floor’, ‘priority’: ‘medium’}. I went to “C:\\_PythonClass\Assignment06\” and run Assignment06.py, which is shown in the screenshot in Figure 14. The final data in “ToDoList.txt” were expected to contain the data of the list table: [{‘task’: ‘buy grocery’, ‘priority’: ‘high’}, {‘task’: ‘cook dinner’, ‘priority’: ‘high’}]

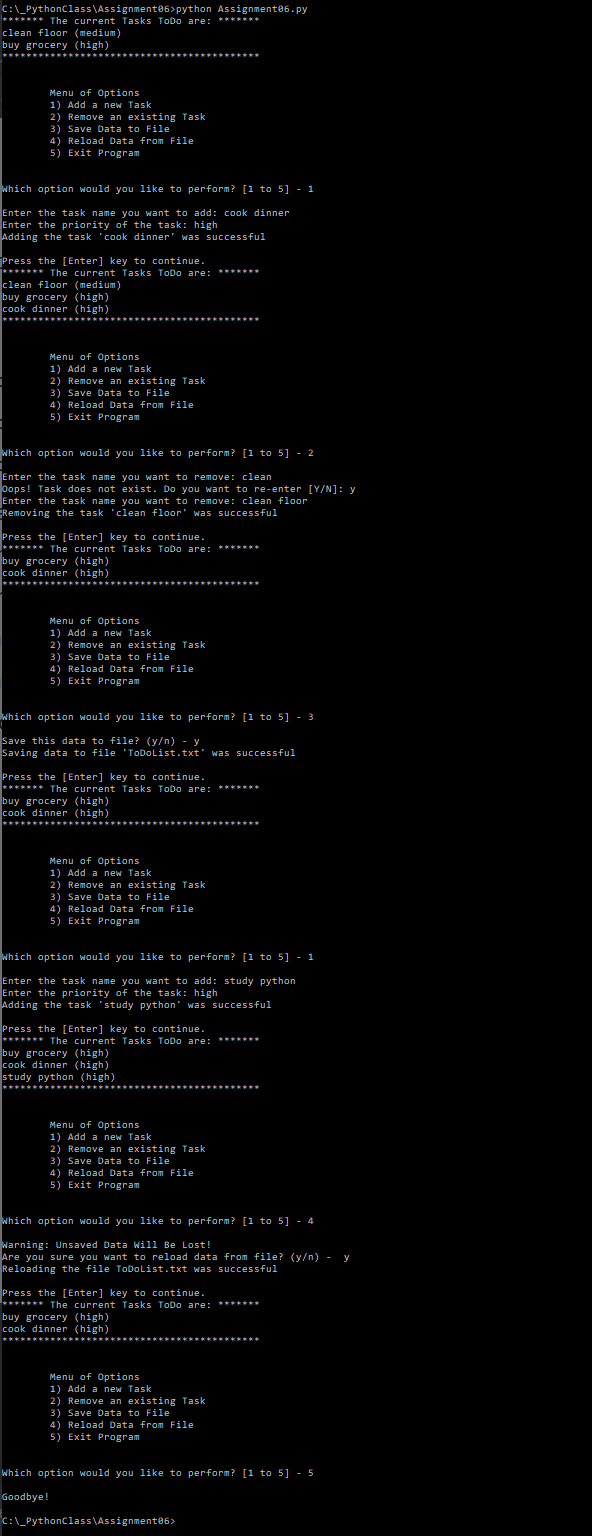


Figure Run Results in Windows Command Prompt

### Checking the Output File

After the Windows command run was completed, I checked the data in “ToDoList.txt” in the working folder. Now the new data in the file are verified and shown in Figure 15. The script is thus completely validated in both PyCharm and Windows command.

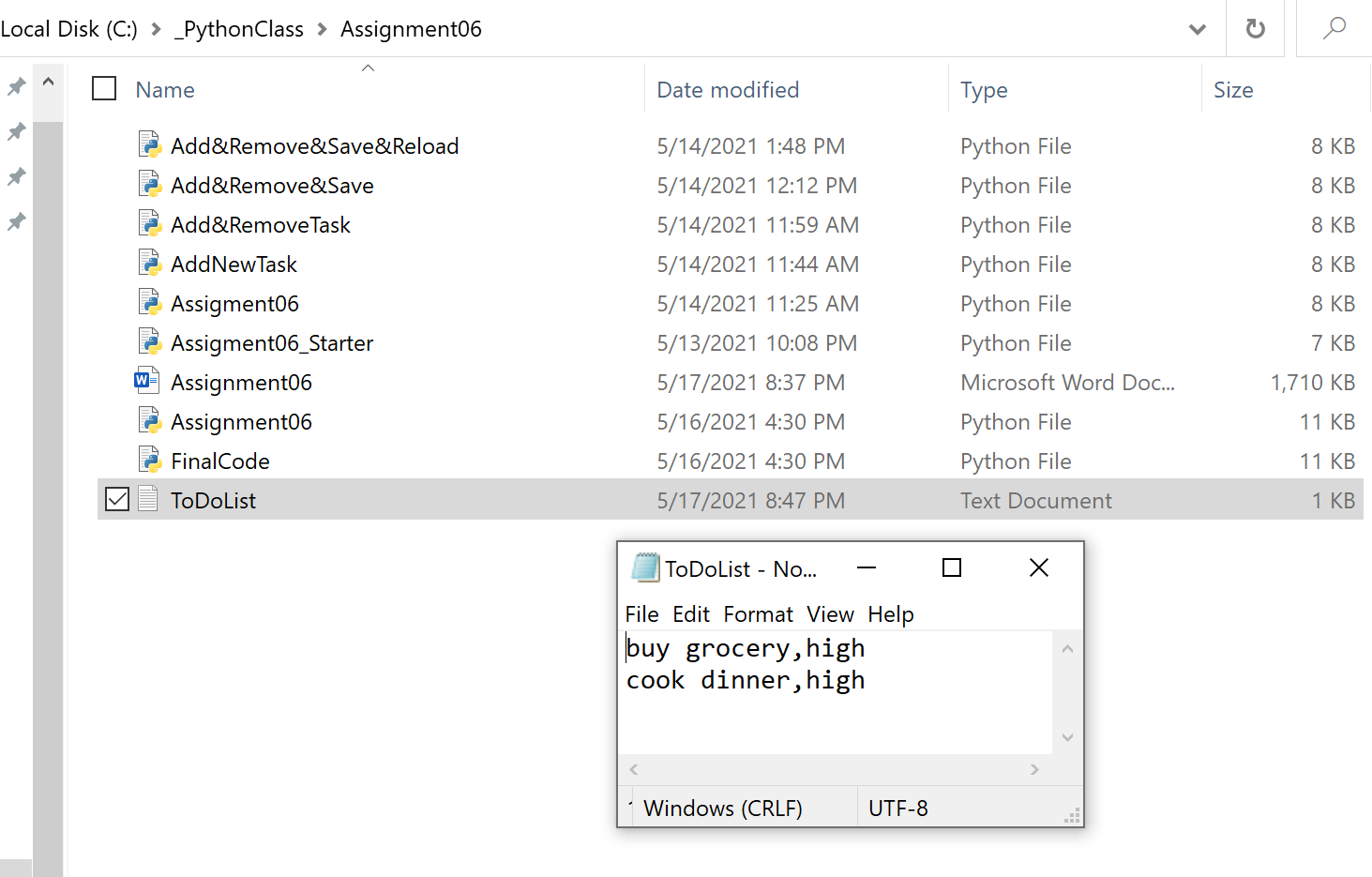


Figure Data in ToDoList.txt After the Run in Windows Command

# GitHub Webpage

I followed the instructions to set up a webpage associated with the Assignment06 repository. The repository name is “IntroToProg-Python.Mod06”. I added simple test in *index.md* to let the webpage show two links (Figure 16).

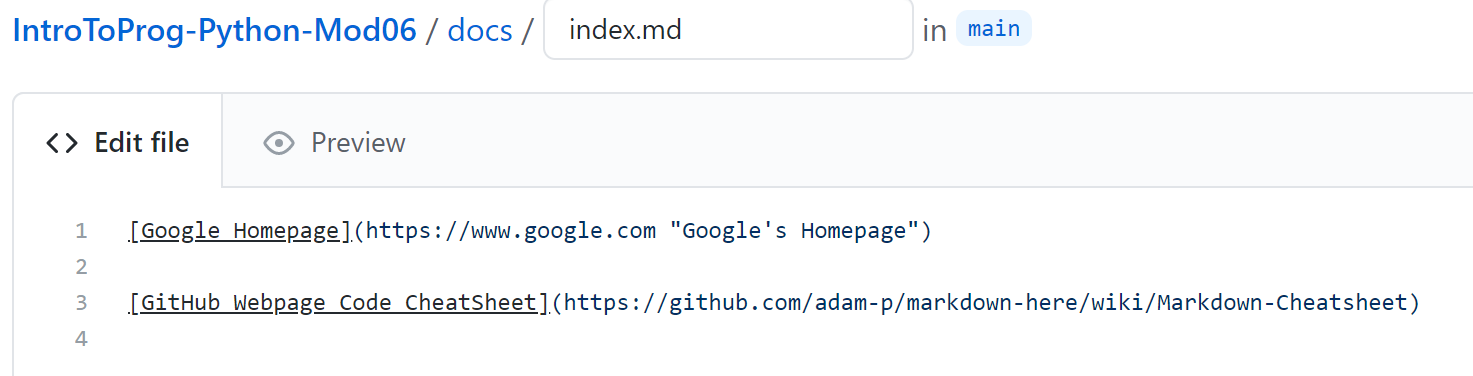


Figure Simple texts in index.md in the "docs" folder

The GitHub Page setting was completed as shown in Figure 17 .

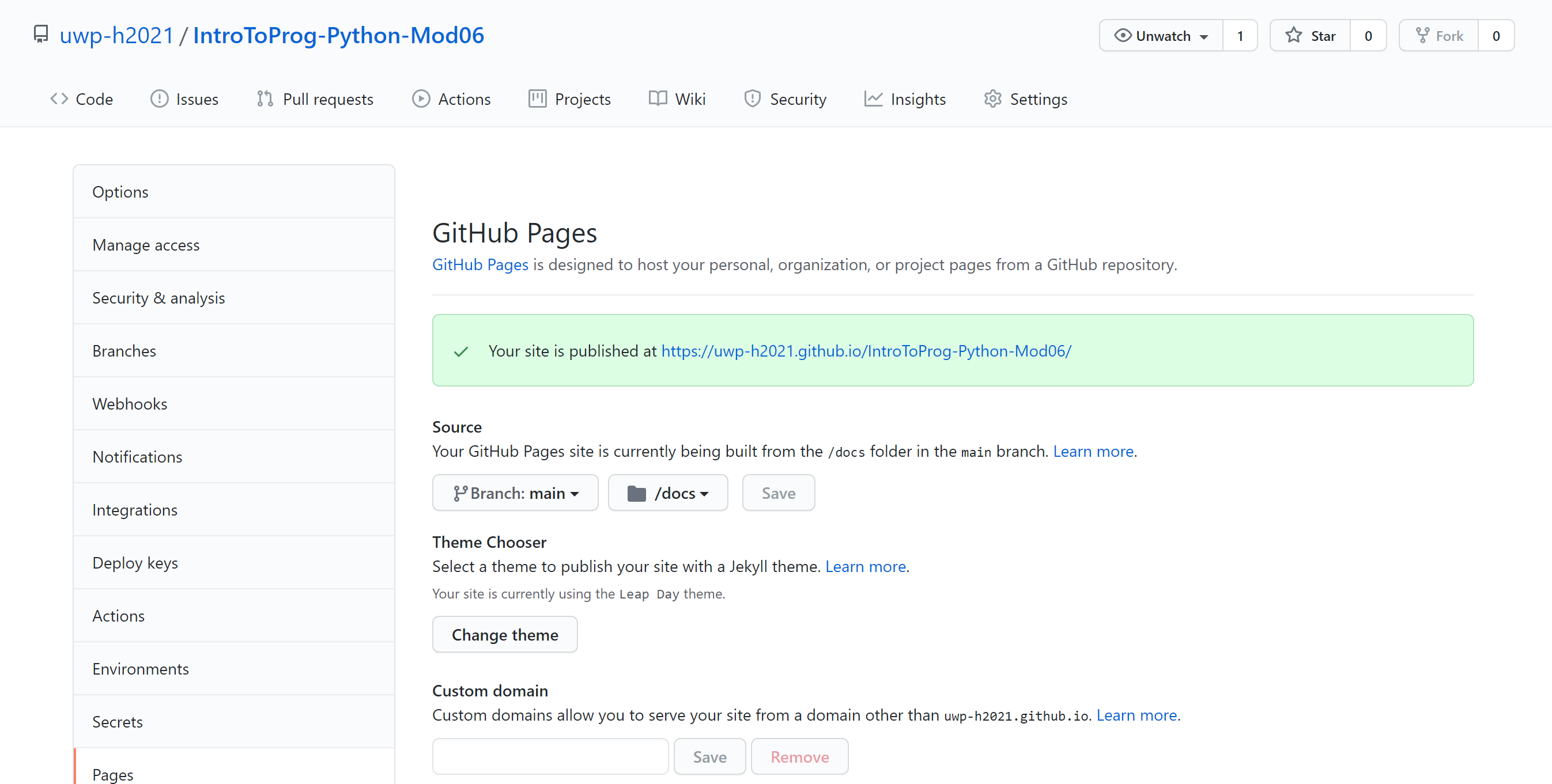


Figure GitHub Page Setting

Current GitHub page is shown in Figure 18, which looks simple but is a good start for my next assignment.



Figure GitHub Webpage for IntroToProg-Python-Mod06

# Summary

In this assignment, I have documented my key learning in Module 06. In the ToDoList script, I learned how to create functions, call functions and assign arguments, execute a function and return variables. I have learned how functions and classes are used to organize an otherwise complex program. Docstrings allow the functions to be understandable for developers. The control+q key allows the quick access to the function information. This assignment has helped us to progress to an intermediate level of programming proficiency. The GitHub webpage setup was simple, and I expect more learning of it in future assignments.