David Grunloh

May 14, 2021

IT FDN 110 A

Assignment 06

<https://github.com/dgrunloh-uw/IntroToProg-Python-Mod06>

To Do List With Functions

# Introduction

In Assignment 06, I altered a Python application utilize functions, which was originally created by RRoot as one long script. This application is used to gather tasks and their priority from a user and store that data in a list. This assignment built on the previous assignment 05. The functionality of the application is mostly the same, but with the addition of the capability to reload the data in the list to a file. This code or very different as it is broken into functions. This can make the code easier to manage and reuse in the future.

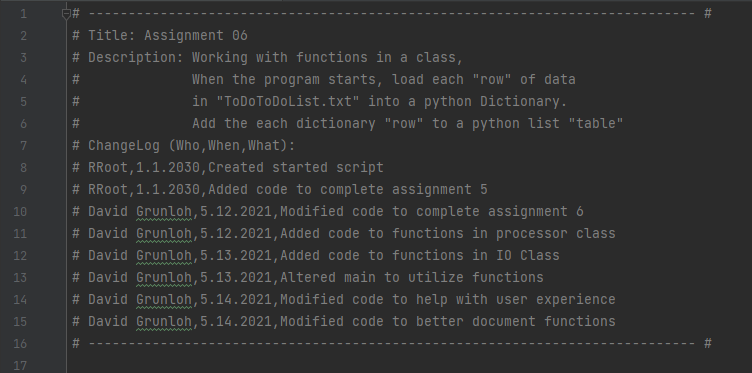
# **Writing the script**

In the script for this assignment there are 6 sections of code, header, define variables, processor class, IO class, and main. These will be described in more detail below as they are used within the application.

## Header

This portion of the script is used to provide information about the purpose of this script to anyone who needs to review or edit the script in the future. The main components that should be included are title, description, author, and change log.

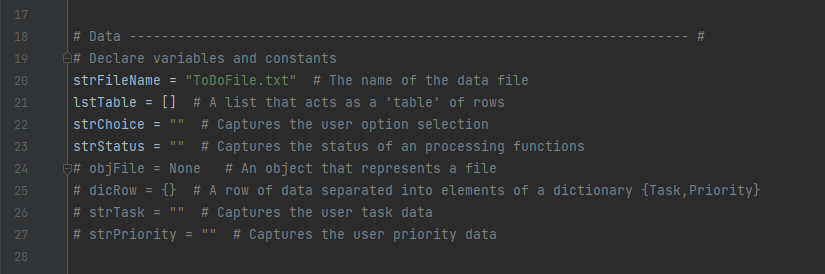
As you can see in Figure 1 below, the application was created by RRoot, but I made several updates and changes to this application. Each of those adjustments were documented in the change log portion of this header.



***Figure 1***

## Define Variables

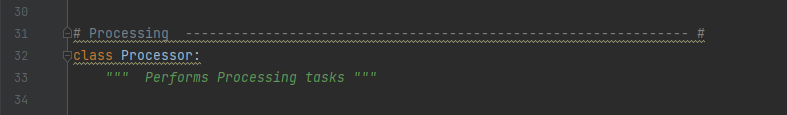
In this section of code, I define some variables that will be used throughout the application. Notice the commented-out variables below in figure 2 are no longer used due to the use of functions.



***Figure 2***

## Class Processor

This class, which you can see defined in figure 3 below includes the functions that do all the processing for the application. Within this class there are several different functions which complete task including read data from files, add data to list, remove data from list, and write data to file. These functions will be further explained in the description of the main class below as they are called.



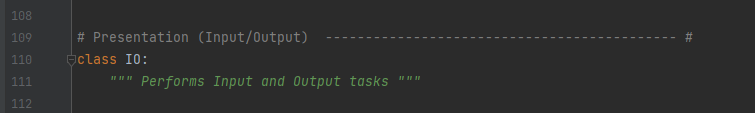
***Figure 3***

## Class IO

The IO Class, which you can see defined in figure 4 below includes the functions that do all the input output operations that interact with the user. Within this class there are several different functions which complete task including:

* Printing the main menu
* Getting the menu choice from the user
* Printing the current task list
* Getting a yes or no choice from the user
* Pausing the application until the user hits enter
* Getting a new task and priority from the user
* Getting a task to remove from the user

These functions will be further explained in the description of the main class below as they are called.



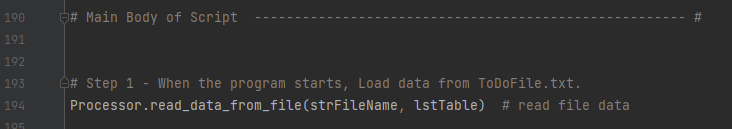
***Figure 4***

## Main

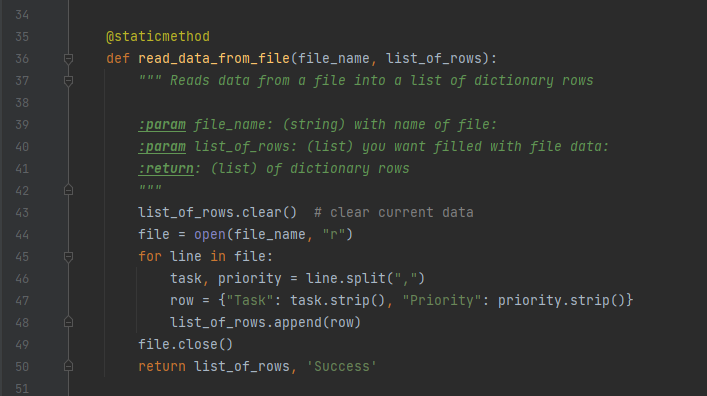
In the main body, you can see the code in assignment 06 is significantly different than Assignment 05. In assignment 06, the main body of code is an initial call to a function to load the list and then you enter a while loop, that calls two IO class functions to get a choice from the user, and then enters into the appropriate if condition based on the input from the suer during the initial IO functions. I will walk through each of the if conditions below.

### Load Data from file

When the application starts, data is loaded from the ToDoList.txt file. You can see the function all in figure 5 below.

  
***Figure 5***

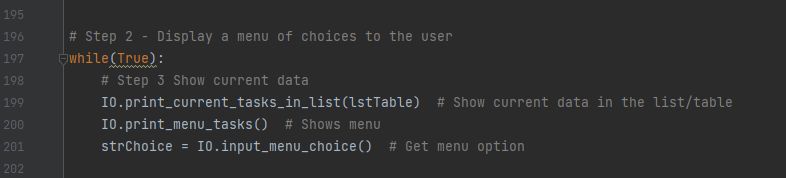
In the code below, you can see the function pulls the data into memory to allow the user to interact and manage that data available to the application. You can see when pulling the data via the function (figure 6), the data is pulled one row at a time into a dictionary object to format the data as its being pulled into a list for use throughout the application.



***Figure 6***

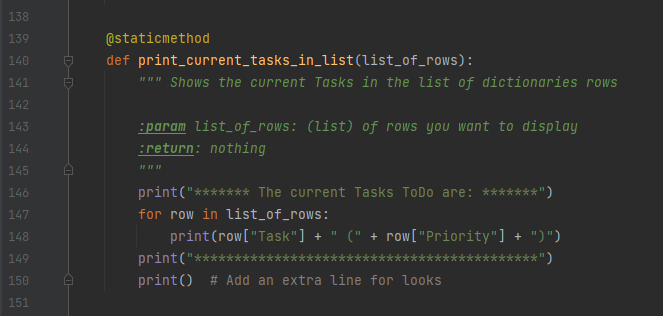
### Loop and Menu Options

In the below section of code shown in Figure 7, you can see the while loop functions calls to print the current task, print menu options and obtain the menu choice from the user. RRoot set up the while loop that ensures the user will be able to loop through the options as many times as they wish. The user will only be able to break out of this condition once they have chosen the correct option, 5, which allows the user to exit the application. As users navigate through the different options, they will always be routed back to this section where the 3 functions explained above allowing the user to use the input function that allows them to select what activity they would like to complete.



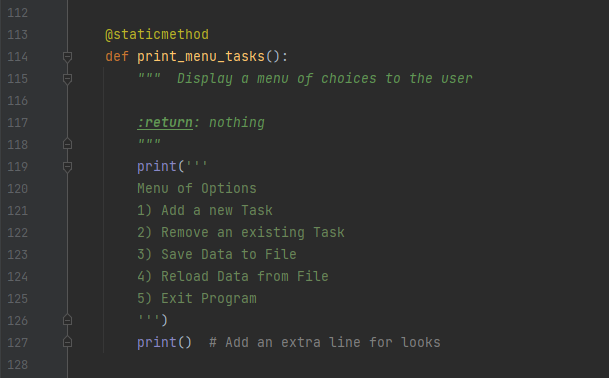
***Figure 8***

Below is figure 9 that shows the function from the IO class that is used to print the current list. This function passes in the current list parameter, list\_of\_rows.



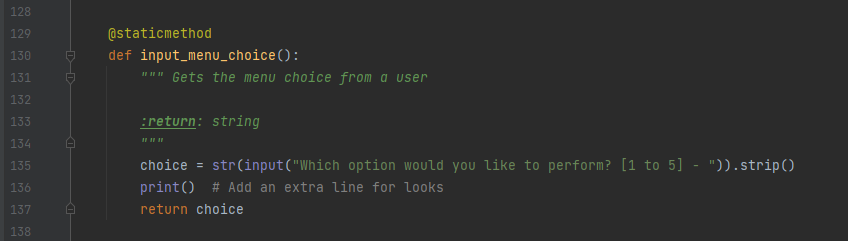
***Figure 9***

Below is figure 10 that shows the function from the IO class that is used to print the menu options. This function is only used to display the list of menu options to the user and does not pass in any parameters nor returns anything to be used later.



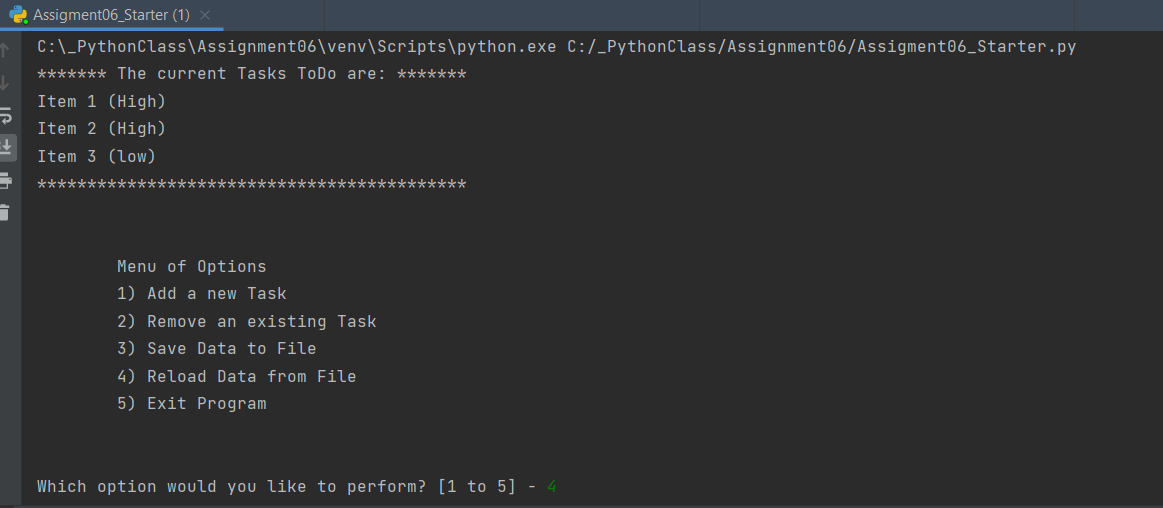
***Figure 10***

Below in figure 11, you can see the input\_menu\_choice function. This function is used to allow the user to determine which section of the application they would like to execute. You can see this function returns the choice to be used in each option if, elif statement.



***Figure 11***

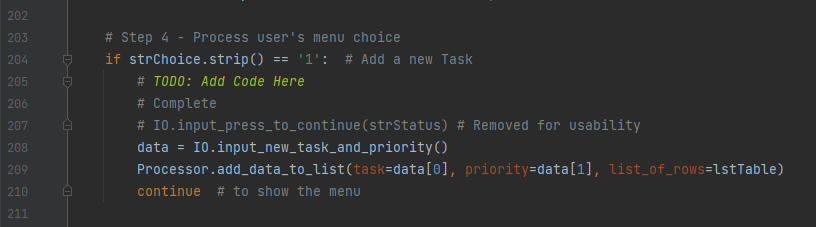
Below in figure 12, you can see this section of code being executed in PyCharm.



***Figure 12***

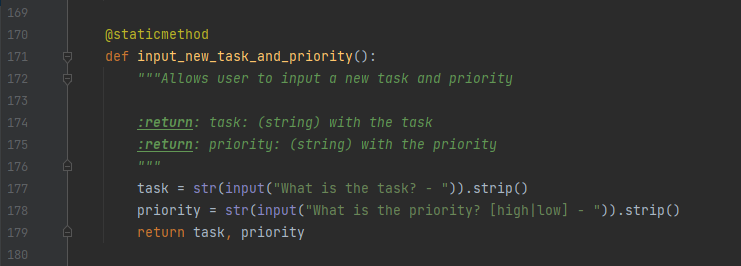
### Option 1 – Gather Input and Store in List

In the below section of code shown in figure 13, which is initiated when the user enters option 1, 2 functions, IO.input\_new\_task\_and\_priority and Processor.add\_data\_to\_list are executed. These functions include all the code to gather a task and priority from a user and then add that to the current list in memory.

******

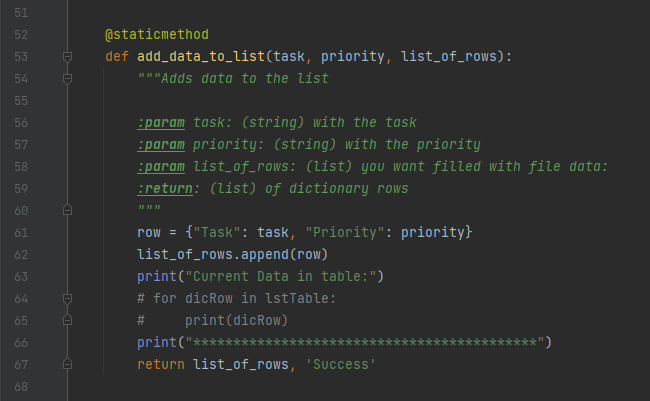
***Figure 13***

You can see below in figure 14 the function input\_new\_task\_andpriority uses variables and the input() function to gather data from the user. This data is then passed out of this function as return values to be processed in the next function.



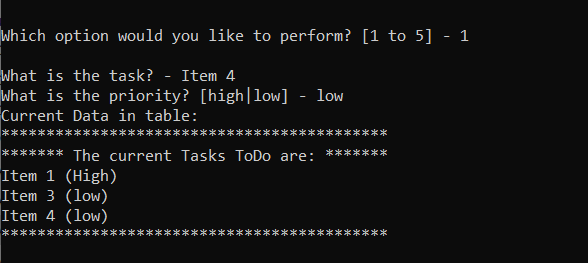
***Figure 14***

In figure 15 below, you can see the code for the function add\_data\_to\_list, which uses the return values from the previous function to add the data to the list stored in memory. In addition to the parameters that are passed in, it also returns an updated list of rows to be used throughout the application.



***Figure 15***

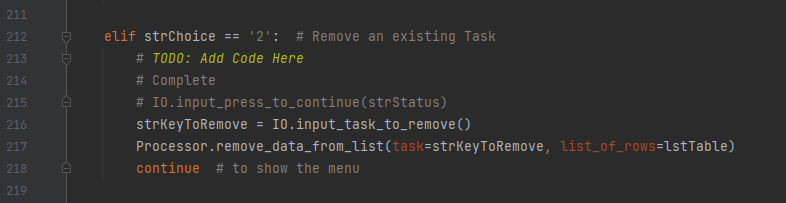
Below in figure 16, you can see this section of code being executed in a command prompt.



***Figure 16***

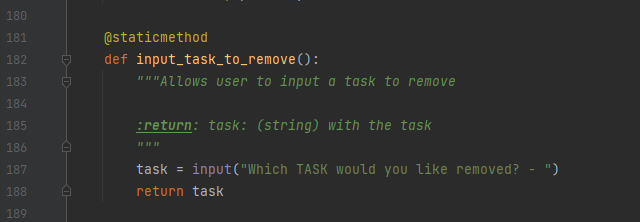
### Option 2 – Remove an existing item

In this section shown in figure 17, the user has chosen to remove an existing item. Within this section, I allow the user to choose which item they would like to remove by entering the Task name. As long as this task name matches a dictionary row Task Key. If there is a match, the code removes that line and returns the user to the main menu. This is completed through the use of the 2 functions below IO.input\_task\_to\_remove and Processsor.remove\_data\_from\_list.

******

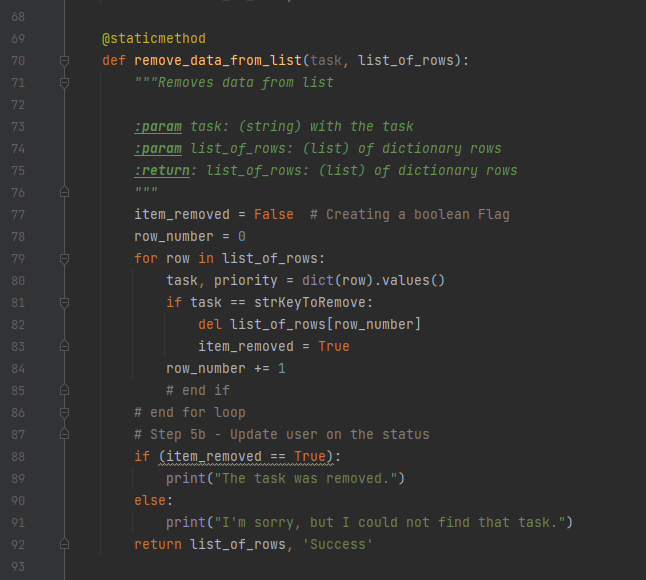
***Figure 17***

Below in figure 18, you can see the function input\_task\_to\_remove. This code asks the user which task they would like to remove and returns that value to be used in the next function.



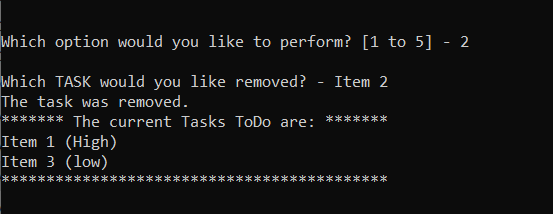
***Figure 18***

Below in figure 19, you can see the function that processes the removal of the data from the list. This uses the parameter from the input\_task\_to\_remove and the current list\_of\_rows to remove the specific task the user specified.



***Figure 19***

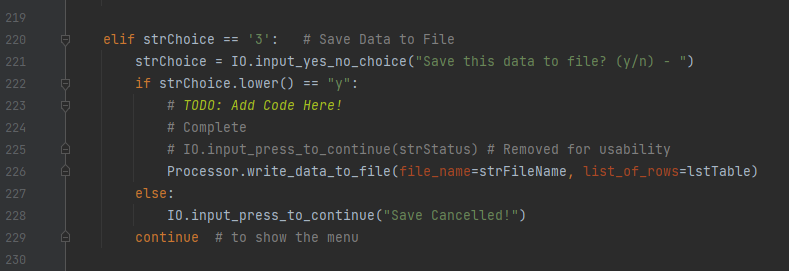
Below in figure 20, you can see this section of code being executed in a command prompt.

******

***Figure 20***

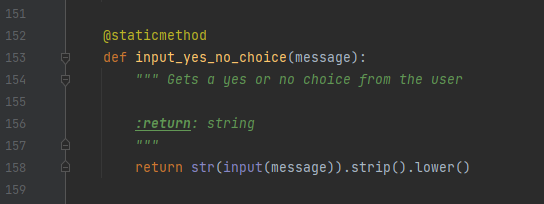
### Option 3 – Save Data to File

In this section shown in figure 21, the user has chosen to save the data to the file. Within this section, the application will write all lines from the list that is stored in memory to the text file that is specified in the variables section. This code will overwrite the text file each time it is saved. You can see that this section uses 3 different functions, IO.input\_yes\_no\_choice, Processor.write\_data\_to\_file, and IO.input\_press\_to\_continue.

******

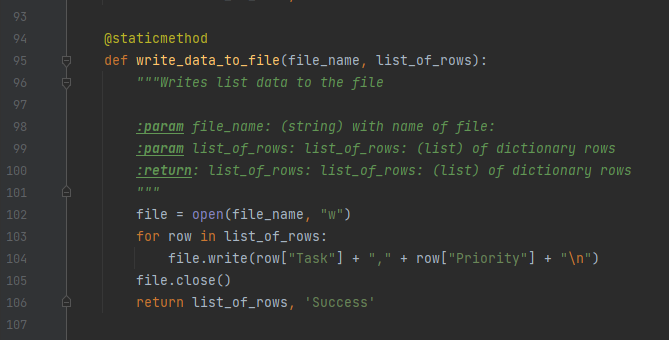
***Figure 21***

The below function, shown in figure 22, the input\_yes\_no\_choice is used to allow the user to confirm they actually want to save the data to the file.



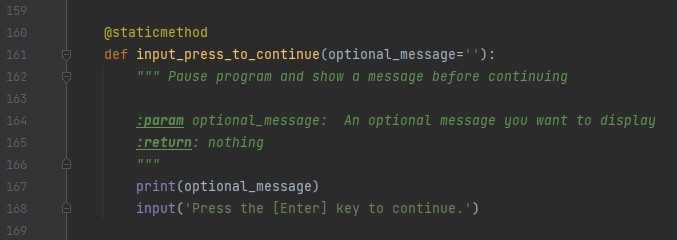
***Figure 22***

If the user chooses y, the user is routed to the Processor.write\_data\_to\_file function shown in figure 23.



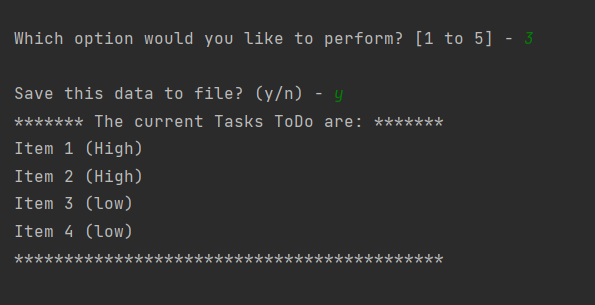
***Figure 23***

Below in figure 24, you can see the IO.input\_press\_to\_continue function that displays the message “Save Cancelled” to the user.



***Figure 24***

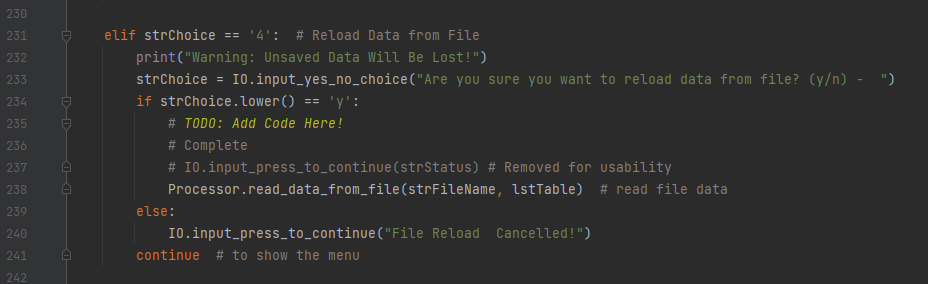
Below in figure 25, you can see this section of code being executed in PyCharm.

******

***Figure 25***

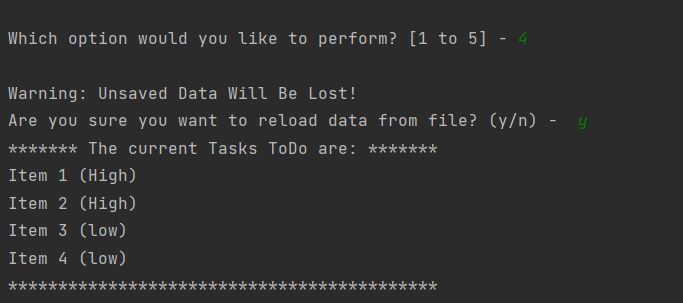
### Option 4 – Reload Data from File

In this section of code shown in Figure 26, the user has chosen to reload the data from the file. This calls the input\_yes\_no\_choice function that is used in option 3 as well as the same function that is used at the very start of the application, Processor.read\_data\_from\_file or the IO.input\_press\_to\_continue function that displays a message “File Reload Cancelled”.

******

***Figure 26***

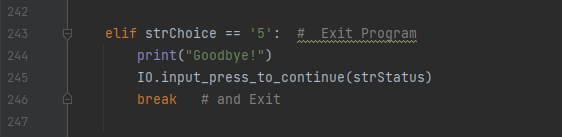
Below in figure 27, you can see this section of code being executed in PyCharm.

******

***Figure 27***

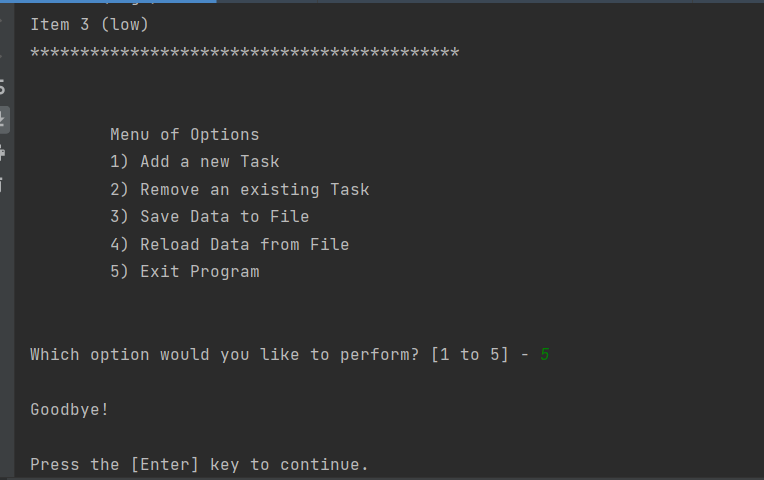
### Option 5 – Exit the program

In this section of code shown in Figure 28, the user receives a prompt asking them to hit enter to exit the program. This is completed via the IO.input\_press\_to\_continue function. Upon hitting enter, the user will exit the program due to the break statement.



***Figure 28***

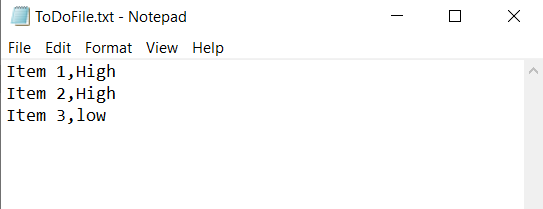
You can see option 5 in action below in Figure 29. This shows the user selecting option 5 and exiting the program using PyCharm.

******

***Figure 29***

## Output Results

Below in figure 30, you can see the output that was written to our data file, ToDoList.txt.



***Figure 30***

# Summary

In this assignment, I utilized python functions in addition to variables, list, and dictionaries to gather input from a user and process it into a list. Users have the ability to loop through options multiple times to add (option 1) and remove (option 2) items to the list. Then option 3, allows the user to save the data to a text file, while option 4 allows the user to reload the data from the text file. Finally, option 5 allows the user to exit the program. This code was also formatted and commented to ensure that it could be reused and is editable in the future.