Daylen Haines

November 5, 2019

Foundations of Programming: Python

Assignment 05

**Python Household Inventory, Interaction, and Storage**

**Introduction**

In this document, I will describe how to write and display a Python script obtaining user inputs for a To Do List, where we provide the user input options to access a preexisting dataset, add, remove, display, save, and exit the program. I will also provide an overview of the difference between a list and a dictionary, the difference between an index and a key, reading data from a file, and explain the “Separations of Concerns.” This document provides the steps I took and can serve as a guidance for beginning programmers writing an interactive script capturing, storing, and displaying user input.

**List vs Dictionary and Index and a Key**

Both lists and dictionaries are simple ways to hold collections of objects that can be utilized in many different ways through Python. The major differences between a list and dictionary is the operators utilized ([] for lists and {} for dictionary). To depict the sequence, lists utilize an index (numeric subscript) while dictionaries utilize key (character) subscripts. Dictionary keys are very similar to columns in a spreadsheet. Its helpful to think of them as a row of data. These rows can be added to a list to form a collection of rows which creates a table for the data. Dictionaries have built-in methods that can make working with the data easier. The main methods or functions of dictionaries are; items, values, and keys. (Root R., Mod5PythonProgrammingNotes. UW Foundations of Python Programming, 2019).

**Reading Data From a File**

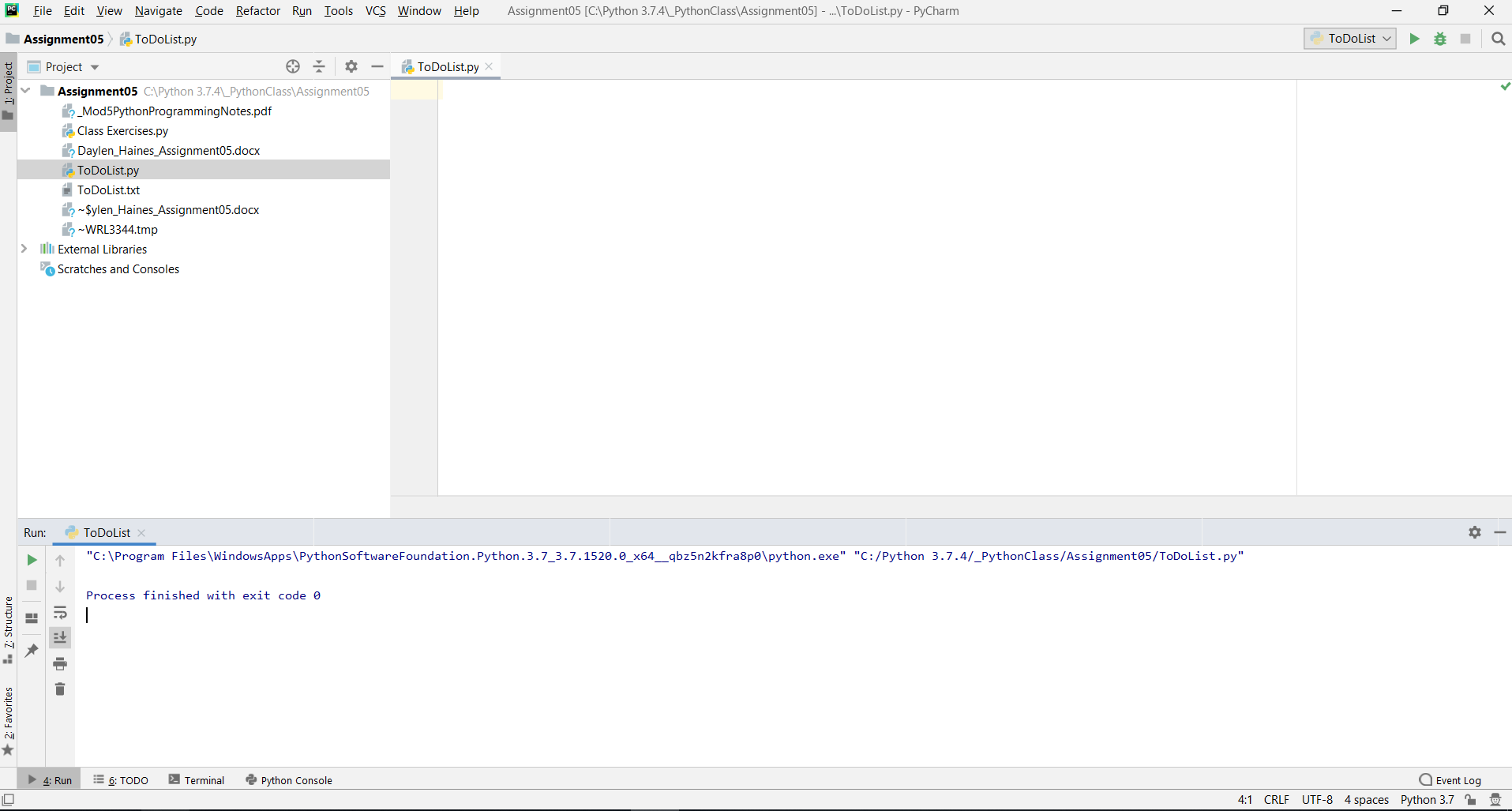
In programming, data is usually held in the computer’s memory and when an application or program is closed, all the data is lost. To ensure data is available no matter if the program is running or not, it is highly recommended to store your data in a file (Root R., Mod3PythonProgrammingNotes. UW Foundations of Python Programming, 2019). As a beginning programmer, storing the data in a simple text document (txt) is a good idea. Through Python, you can reference a given file in your script to read, write, or append, a given data file you would like to utilize in your coding. This data can be added into both a list and dictionary file.

**Separations of Concerns**

As we begin to develop more complex Phyton scripts/ programs, keeping our data clean and concise gets more and more important. Not only will this benefit the programmer, it will also benefit anyone trying to read your code. Separations of Concerns deals with separating your program into predefined area that address a different concern (Root R., Mod3PythonProgrammingNotes. UW Foundations of Python Programming, 2019). The main areas are Data (ex. Declaring variable and constants), Processing (Tasks you are going to perform on data within your code), and Presentation (input/ output, obtaining user input). By structuring your script/ program to separate these three areas will allow for much cleaner and easier programming. The use of functions can provide greater ability to fit your code in the predefined areas.

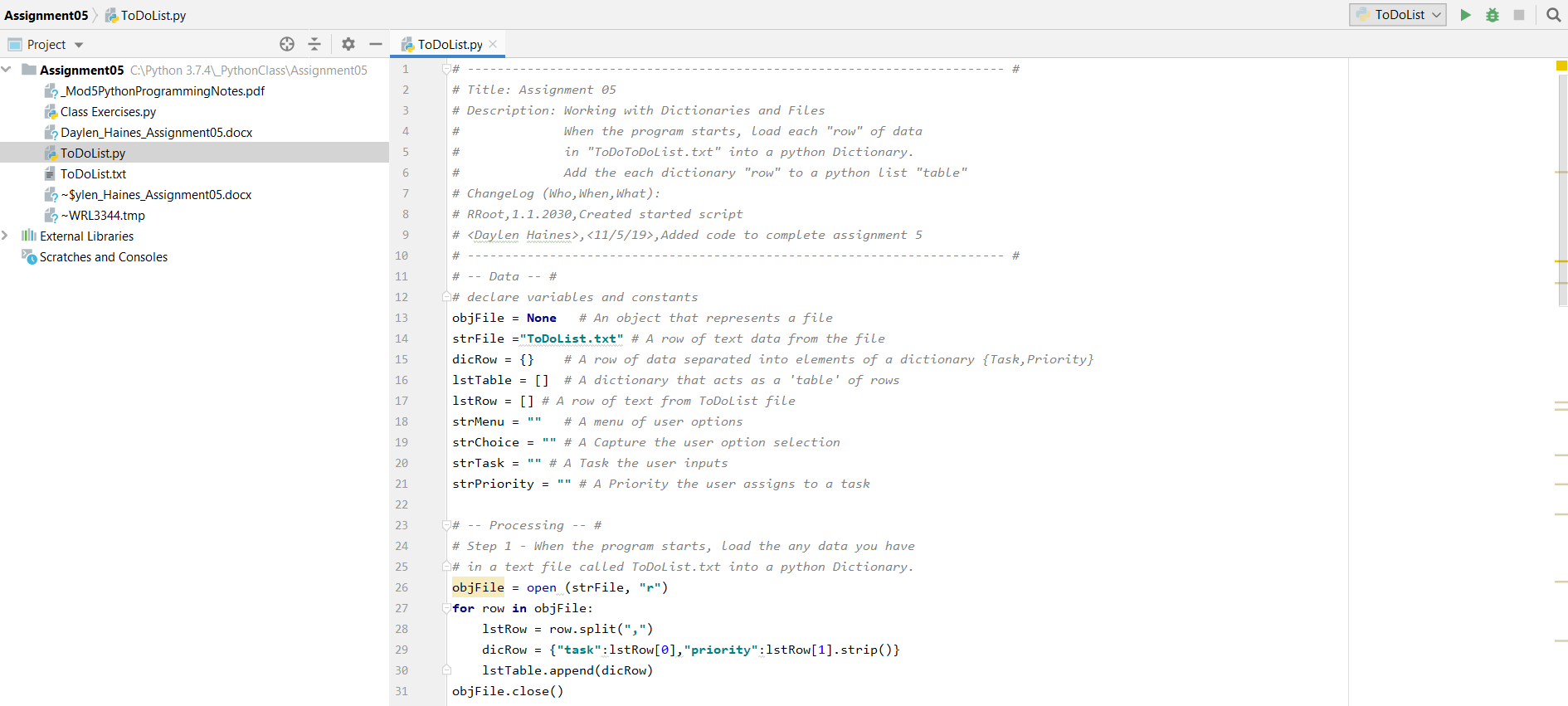
**Writing the Requested Python Program**

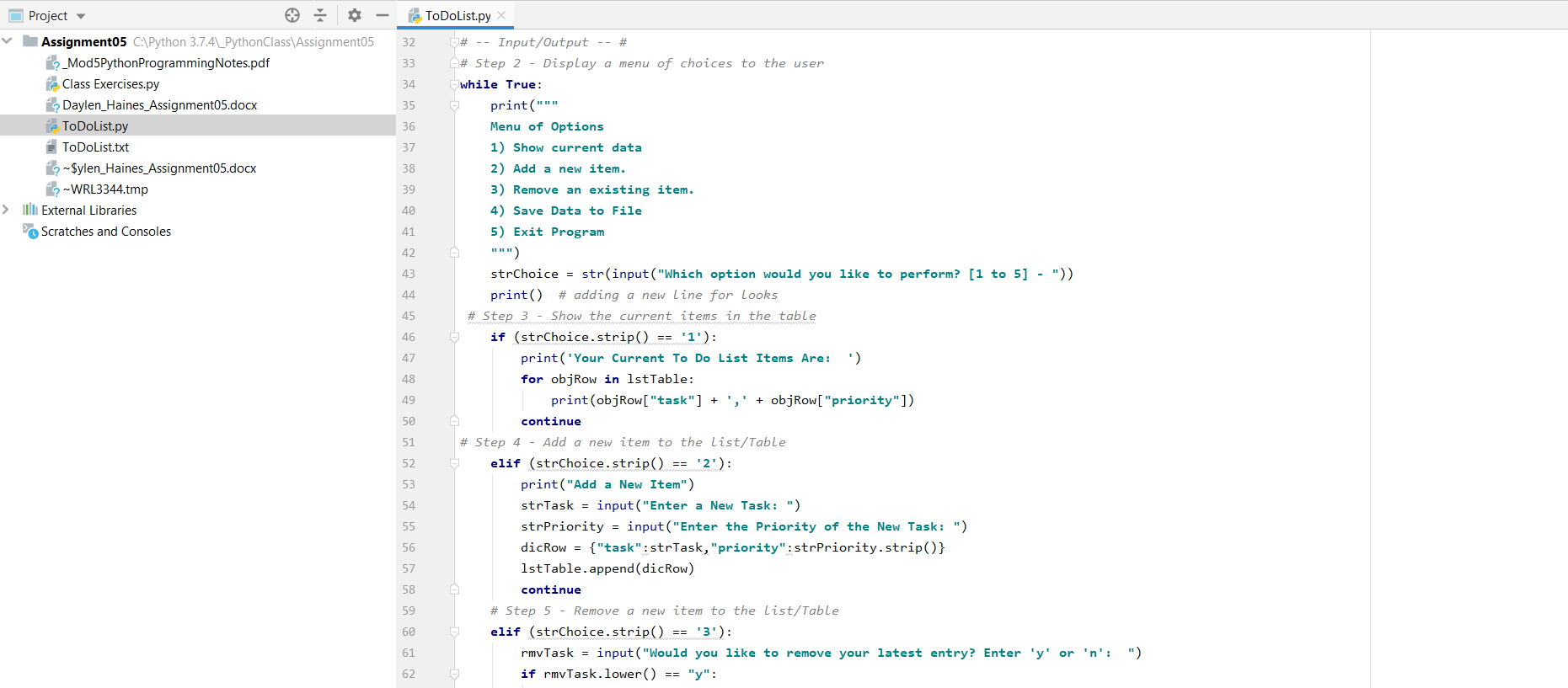
I first created a new sub folder for Assignment05 and opened up the PyCharm program. Once the program was opened, I followed the steps that professor Root provided in the “Intro to Python Mod03” (<https://www.youtube.com/watch?v=EoCXrbkdI0c&feature=youtu.be>) (External Site) video, but made some minor modifications to create ToDoList.py file in my Assignment05 folder. Once I had PyCharm up and ready to go, I utilized the “Intro to Python Mod05” (<https://www.youtube.com/watch?v=8pv_USzS_Ko&feature=youtu.be>) (External Site) video Professor Root provided and Python Programming for the Absolute Beginner book (Dawson M., Python Programming For The Absolute Beginner. Course Technology, 2010) to complete the requested task of modifying a new script to manage a To Do List that contains tasks, priorities, and the following interface steps. Creating a script that loaded the data I had in a text file into a Python dictionary, displayed a menu of choices for the user, showed the current items in the table, added a new item to the list/ table, removed a new item from the list/ table, saved tasks to the To Do List text file, and exited the program.

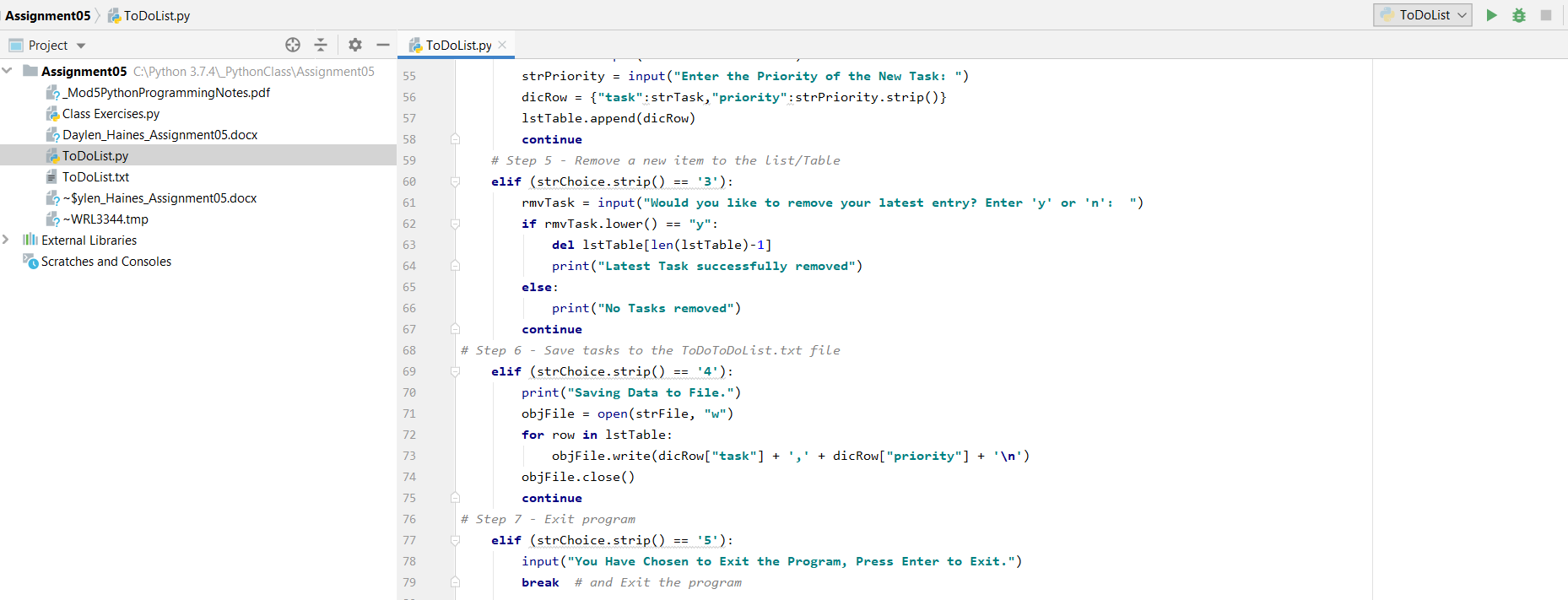


**Figure 2.0 – PyCharm Upon Launching**

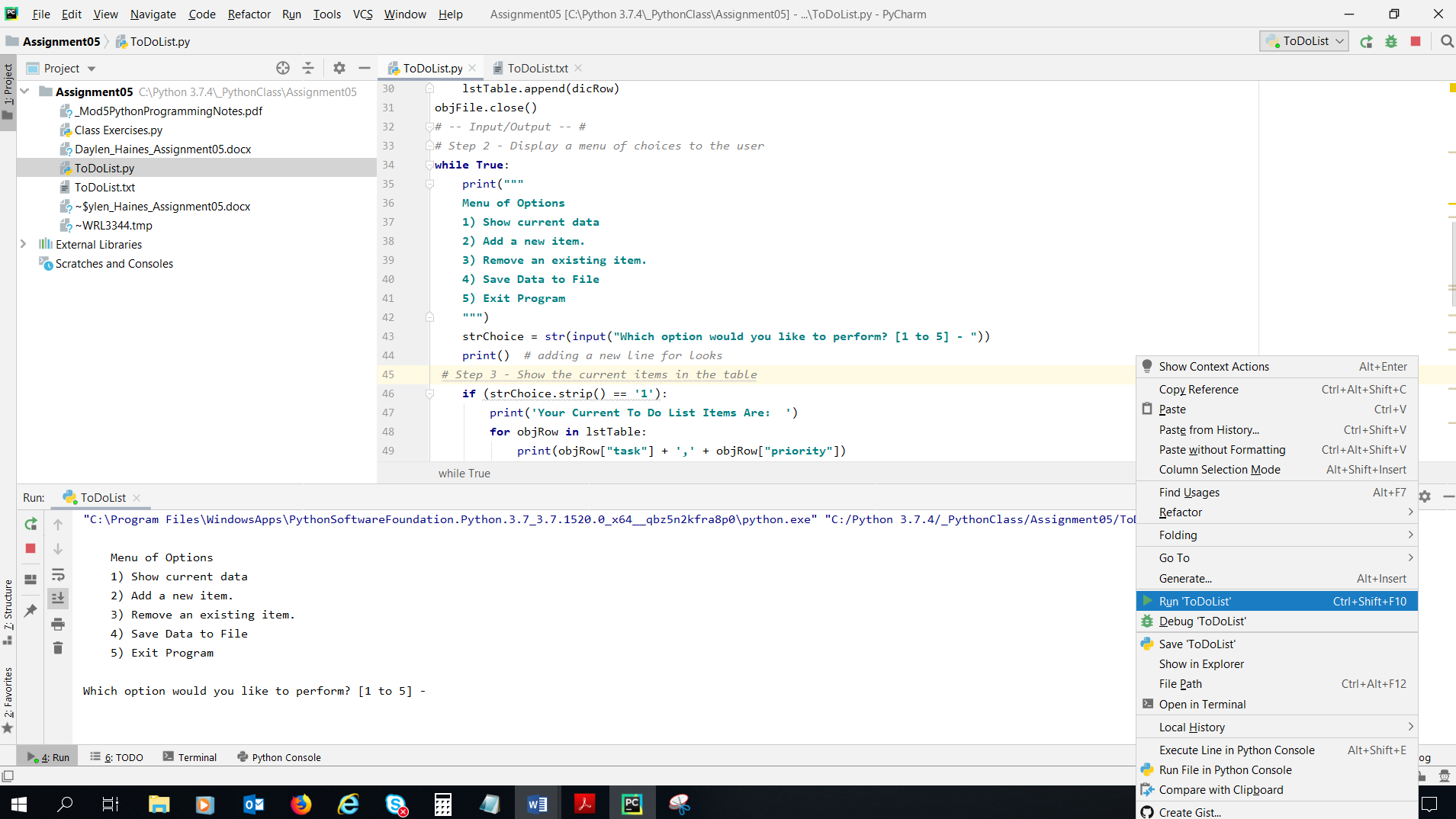
Figures 3.0, 3.1, 3.2 depict the program script I modified for the To Do List activity and its output.



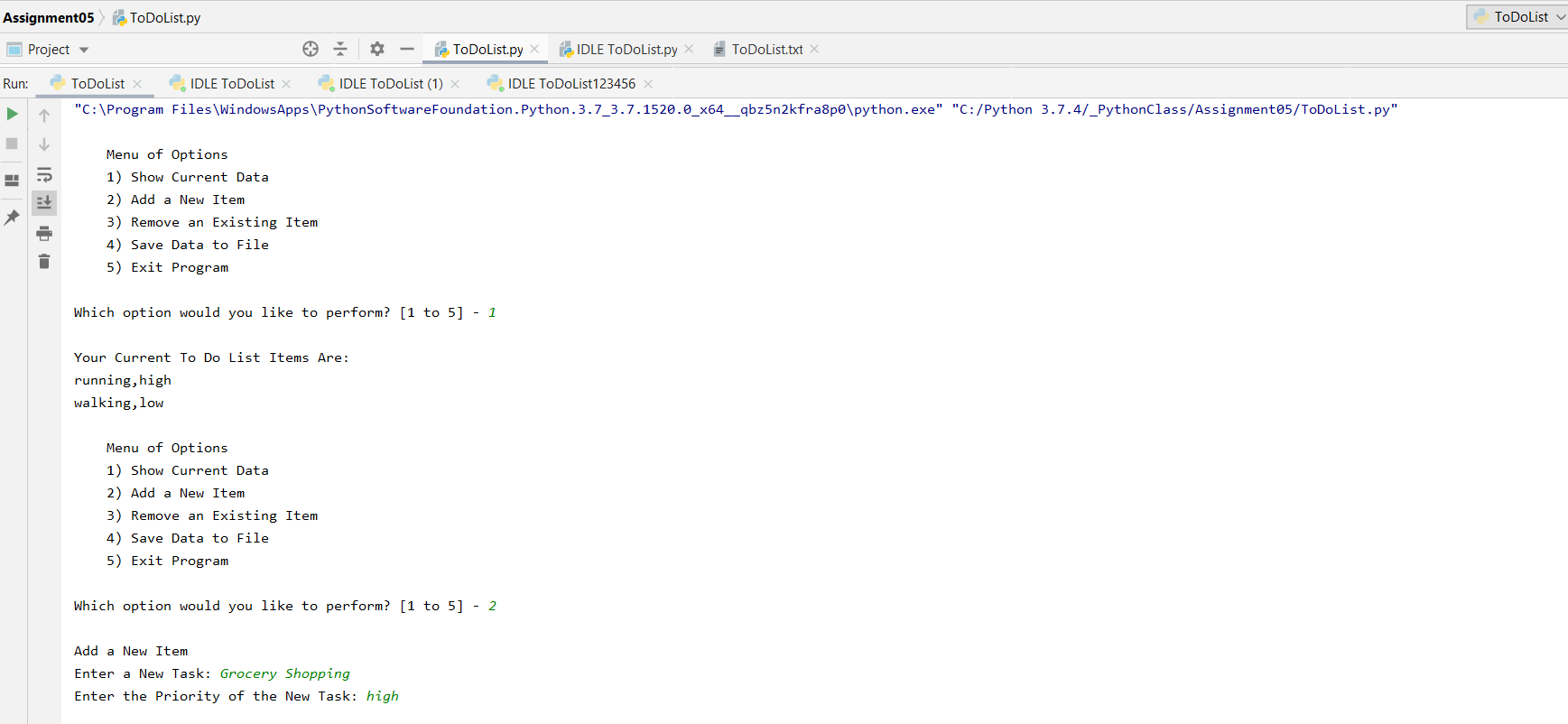




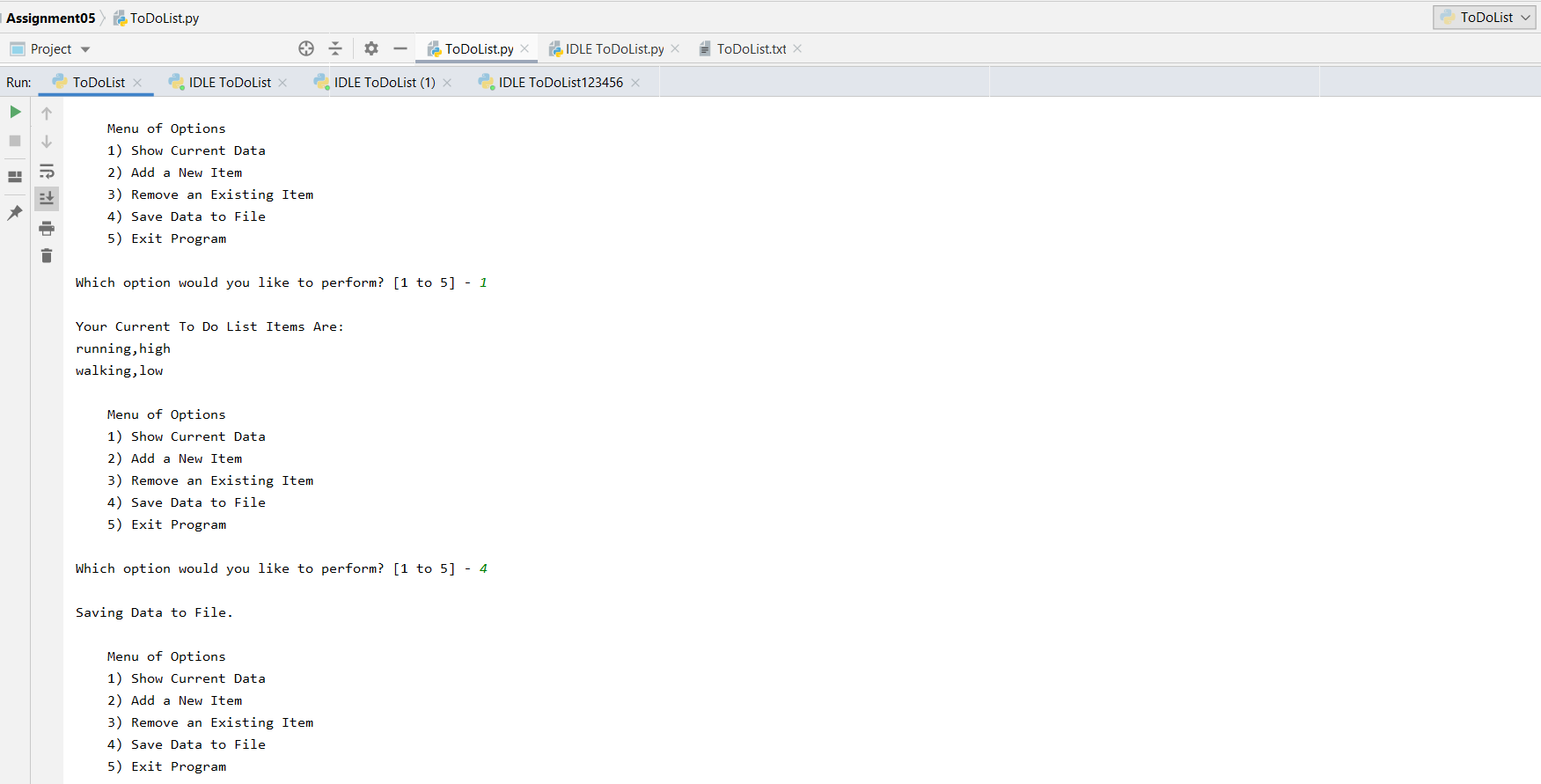
**Figures 3.0 – Showing the Test Python File I Modified in PyCharm for the To Do List Script**

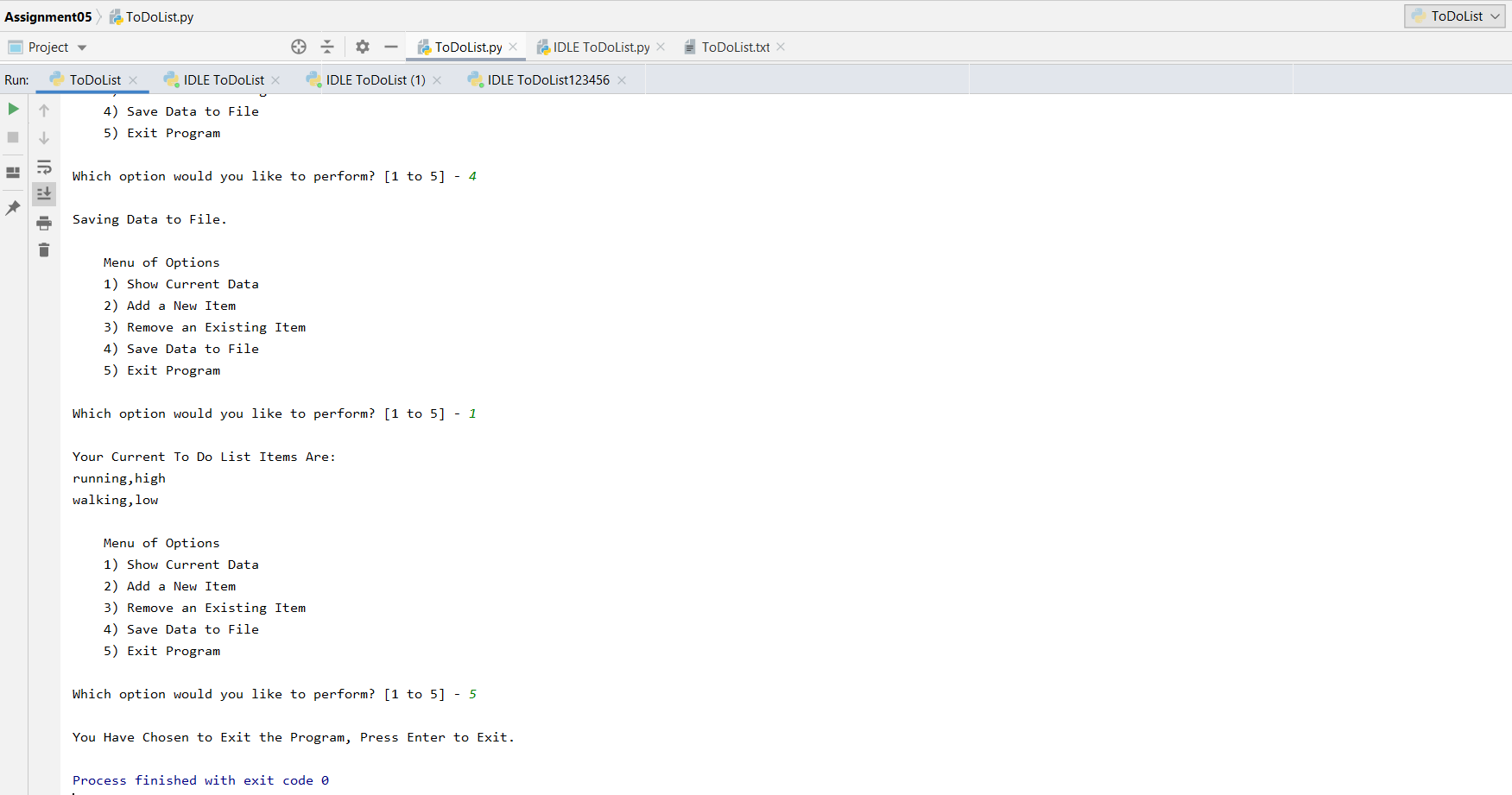
****

**Figure 3.1 – Showing the Test, Run, and Interactive PyCharm Functions for the To Do List Script**

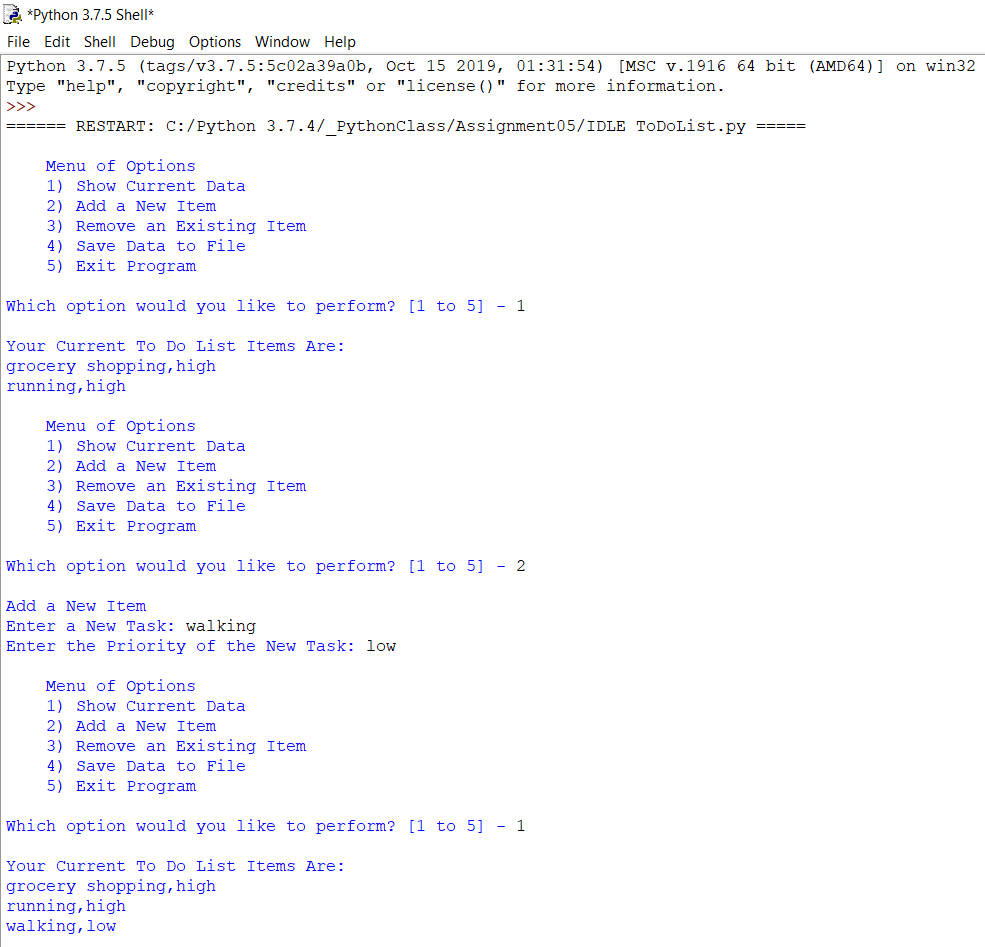


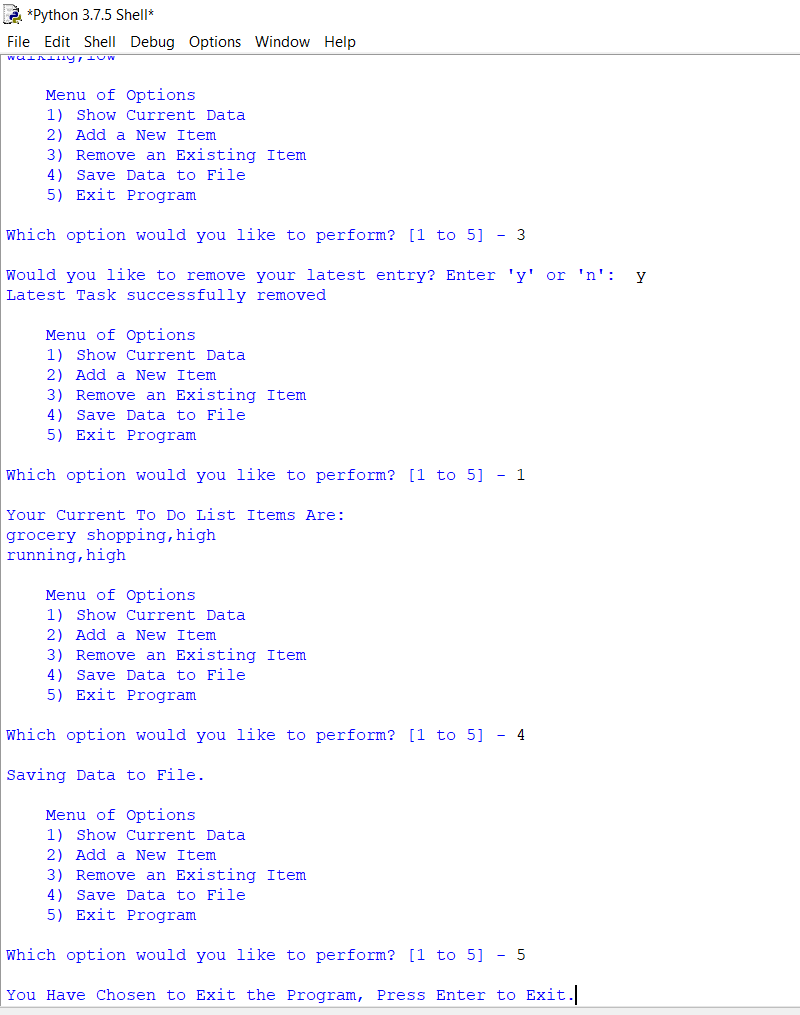




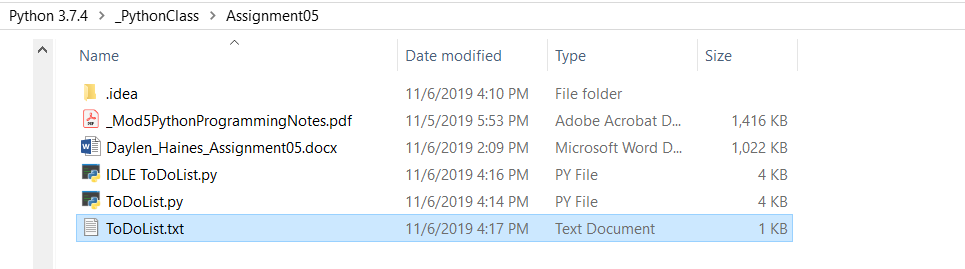


**Figure 3.2 – Showing the PyCharm Interactive Version of the To Do List Script**

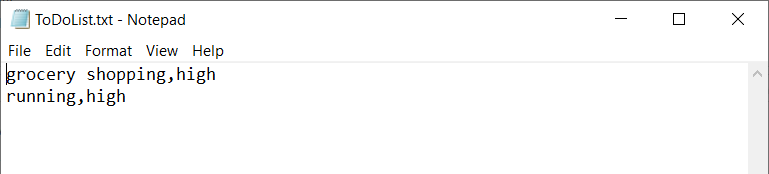




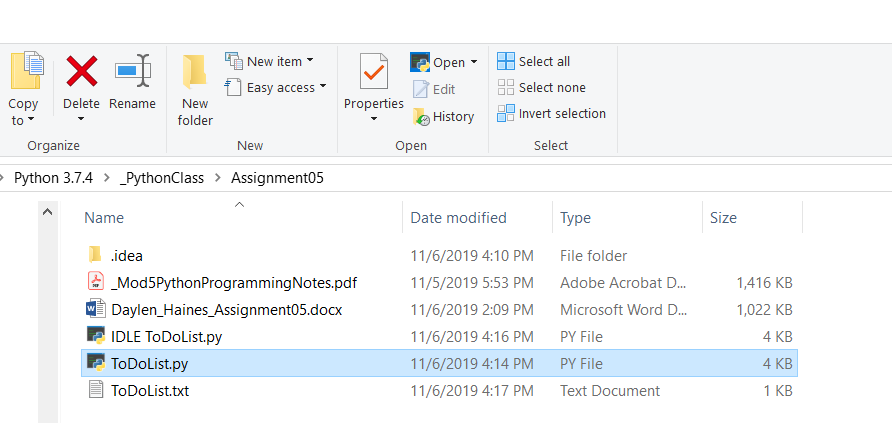
**Figure 3.3 – Showing the IDLE Command Shell To List Script**



**Figure 3.4 – Showing the Assignment05 Folder**



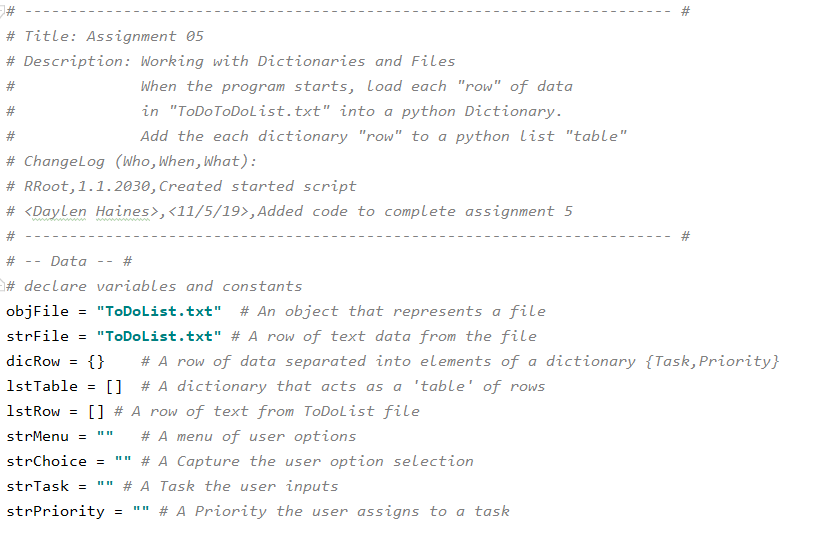
**Figure 3.5 – Showing the Text Input from the IDLE To Do List Script**



**Figure 3.6 – Showing Accessing Command Shell for To Do List Script – Didn’t Run for Some Reason**

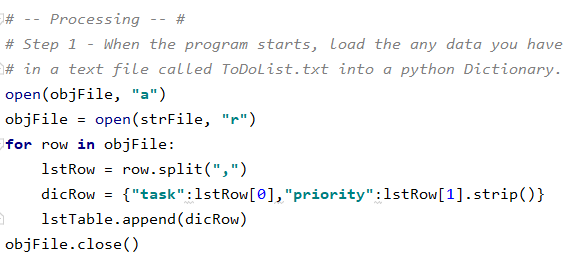
**Script Details**

I first created a new project in PyCharm located in my PythonClass Assignment05 folder. Next, I created a new Python file named ToDoList.py in PyCharm. Now that I was up, running, and saved, I began writing my script. I began by copying and pasting the Assignment05\_Stater.py script that professor Root provided into my ToDoList PyCharm project (<https://canvas.uw.edu/courses/1342958/files/59419542?module_item_id=9942791>) (External Site). Once the starter text was pasted into PyCharm, I began updating the key information, pseudo comments, and declaring the variables and constants that will be utilized throughout the rest of the script (Figure 4.0).



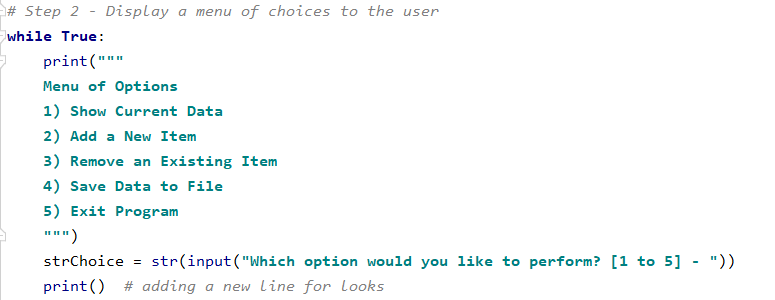
**Figure 4.0 – ChangLog updates and Declaring Variables**

Once all of the variables and constants were declared for the ToDoList script, I was ready to start on Step1 (Figure 4.1). In order to load data from a ToDoList into a Python Dictionary, I first created the ToDoList.txt file in my Assignment05 folder. Once the text file was created, I was ready to reference the file, split the variables by a comma to define the “task” and “priority” for the dictionary. After I defined the dictionary row (dicRow), I then appended the dictionary table on the lstTable. Lastly I closed the objFile to enable the script to move on to the next step.



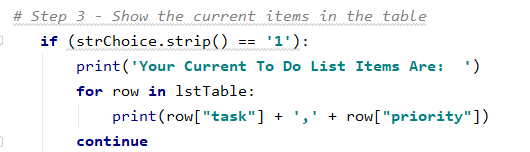
**Figure 4.1 – Load data from ToDoList.txt into Python Dictionary**

Now that Step 1 was complete, I was ready to move onto step 2 which is to Display an interactive menu of choices to the user. For this, I used the while True: command to continuously run the script until the user exits the program. In order to display the menu, I simply utilized the print function and then defined the Menu. Lastly I created an input string to allow the user to select which option they would like to choose from the Menu of Options (Figure 4.2).



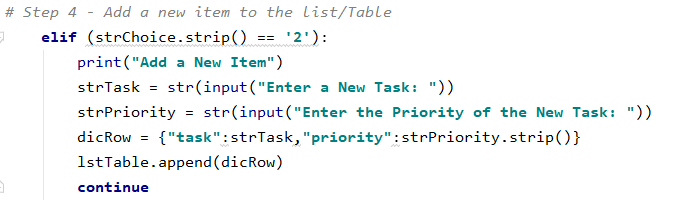
**Figure 4.1 – Interactive Menu for User Interface**

Once the user was able to interact with the diplay menu, I needed to perform step 3 which was to display the current items in the table. To do this, I inserted a print function which tells the user that their To Do List items will be displayed below. This will provide feedback to the user weather the list is empty or populated. Next I referenced the lstTable to print the “task” and “priorities” for all items in the To Do List. This displayed everything that has been entered to the To Do List script (Figure 4.2), completing step 3 and moving on to step 4.



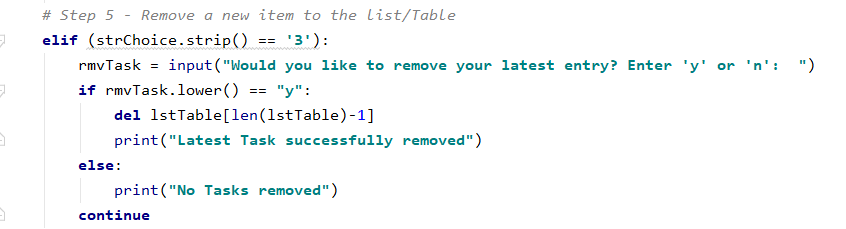
**Figure 4.2 – Displaying Current Items in Table**

Now that I had enabled the user to display their current To Do List items, I was ready to move onto Step 4 which was to Add a New Item to the List/ Table. To add items to the To Do List, I defined two input functions to capture the users input regarding the “task” and “priority” they would like to add to the list. Once their inputs were captured, I added them to the dicRow and appended them to the lstTable (Figure 4.3). This allowed the user to input their desired inputs and adds them to the script dictionary.



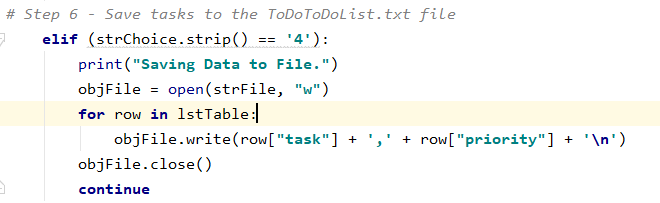
**Figure 4.3 – Add a New Item to the List/ Table**

After completing Step 4, I was ready to move onto Step 5. Step 5 entails removing a new item that was added to the To Do List / table. In order to remove a new item, I needed to gather the users input to ensure they would like to remove their latest entry. I utilized an input function and asked the user to select yes (y) or no (n) if the user selected yes, the last entry in the lstTable would be deleted and communicated to the user. If the user selected no, then the file would tell them that no tasks had been removed (Figure 4.4)



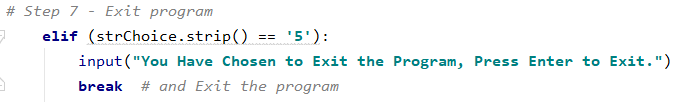
**Figure 4.4 – Remove a New Item to the List/ Table**

Step 6 asked to Save the tasks to the ToDoList.txt file. If the user selected the 4th item on the Menu, I told them that the data is being saved to the file via a print function. In the background, the script opened the ToDoList.txt file and chose to write (depicted by W) to the file. Utilizing a for loop to open the lstTable and write the “task” and “priority,” followed by a return (“\n”) to go to the next row in the ToDoList.txt file. This continues to write the “tasks” and “priorities” input through menu option 2, until all the items entered to this point have been saved or removed (step 5) from the ToDoList.txt file. Once the file is updated, the txt file closes (Figure 4.5).



**Figure 4.5 – Save Tasks to the ToDoList.txt File**

For the last and final step in our To Do List script, step 7 asked to exit the program. For exiting the program, I wanted to make sure the user truly wants to exit the script. To do this, I utilized an input function and request they hit Enter to Exit the script. Once they hit Enter (or any key with the input function I wrote), the script breaks and closes (Figure 4.6).



**Figure 4.6 – Exit the To Do List Program**

Once I had all the Steps complete and the script established, I was ready to run the To Do List script in PyCharm. I right clicked and went to Run ‘ToDoList.py’ which put it in an interactive window within PyCharm (Figure 3.1). Here I could try out my script to make sure it functioned as intended (Figure 3.2). Once I was happy with my script and my ToDoList.txt file was being written to as I desired (Figure 3.5), I ensured my file was saved so I could run it in Command Shell. To run my script in Command Shell, I accessed the folder location on my computer, highlight the ToDoList.py (Figure 3.6) file selected the Python Open drop down, selected Python 3.7 and this launched my script in the Command Shell. For some reason, my script would not run in Command Shell. It would run perfectly fine in PyCharm without errors but would automatically close in Command Shell without prompting any inputs. To see if there was something wrong with my code, I ran the file in IDLE python Shell (Figure 3.3) where again the script function as intended. After hours of reviewing class notes, searching the web, and trying to rewrite the file, I was unsuccessful with getting the script to run in the Command Shell. However, I was able to ensure the user inputs were captured as desired through IDLE Shell and PyCharm, by opening the ToDoList.txt file and making sure the inputs were captured correctly (Figure 3.5).

**Summary**

Python has proven to be powerful, and user friendly programming language for creating interactive programs where you can access txt files, capture, display, save, and write user inputs. Utilizing what we learned in Modules01, Modules02, Modules03, Modules04, and expanding our knowledge base to include lists, and dictionaries, made Python efficient to use for this activity. That being said, there were some programming difficulties I had with Step 1 which is why I believe my script is not properly executing in Command Shell. Going forward with the guidance of Professor Root and Author Dawson, I am confident that I will be able to continue to expand my Python programming capabilities and successfully write more complex scripts and programs.