To-Do List Program

# Introduction

In this assignment I will give an overview of how I modified an existing program that allows the user to choose various actions to create a To Do list. The program reads the existing list and can add, delete, and print the tasks and their priorities, or save the information back to the file. When adding tasks, the program asks for the name of a task, and then asks for its priority. Both pieces of data are stored in a dictionary, where each task and priority are a row of data. Each dictionary entry is then added to a list to create a table of data which can then be printed or saved.

# Writing the script

To begin with, we were given a starting template to follow in developing the code as shown in Figure 1. The variables had already been declared and the menu branches laid out. After any branch except choice 5 is completed, the program loops back to the menu.

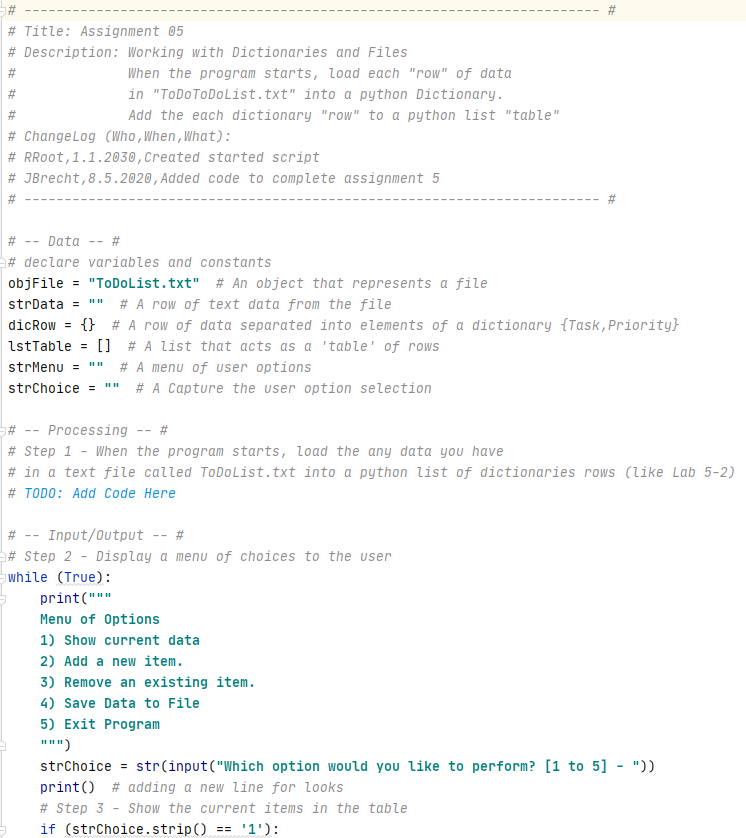


Figure 1: Starting code from assignment

## Step 1: Load the Data into Memory

The first step is to import the data from ToDoList.txt into memory so the program can act on it. As shown in Figure 2, the text file is opened and each row is extracted and split to create a dictionary element. Each dictionary element is then appended to a table, which is then printed.

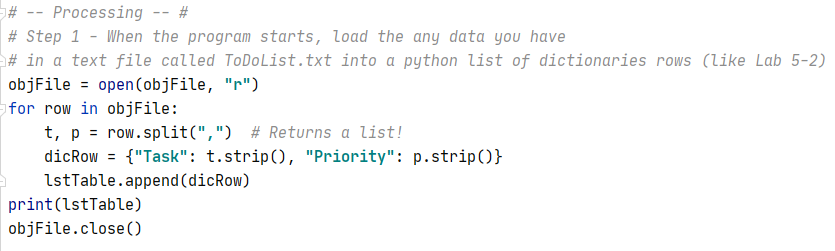


Figure 2: Reading from the file

## Step 2: Display a Menu

The next step was to show the user the possible actions they could choose and to request the user to input 1, 2, 3, 4, or 5 as shown in Figure 3. Since the data would be used later in the program, the input was assigned to a variable. Since most people have more than one task in their To Do list, a while loop was added to allow multiple inputs in a single run and to keep the data in memory until it could be saved.

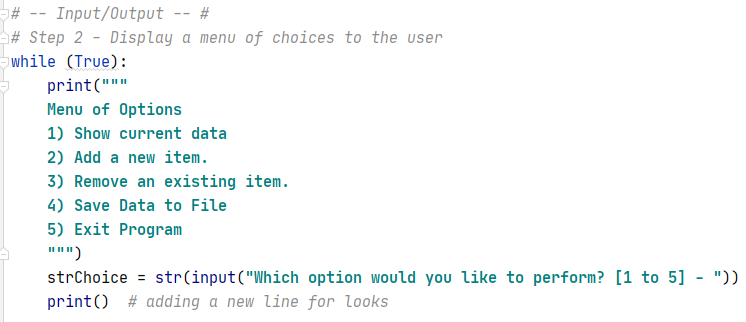


Figure 3: Code to request user input

## Step 3: Print the To Do List

If the user entered “1” then the list stored in memory got printed by row in a tabular format. If there was no data, an empty list was printed. In either case, the program looped back to the menu (see Figure 4).

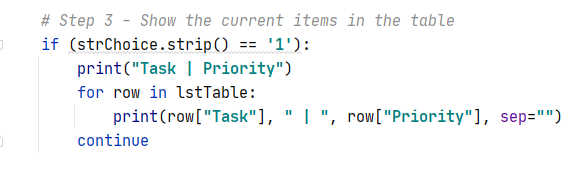


Figure 4: Choice 1 code

## Step 4: Add a New Task

If the user entered “2”, a new key-value pair needed to be added based on the user’s input. The inputs requesting the task name and priority were added as a dictionary row, which was then appended to the existing data table list as shown in Figure 5.

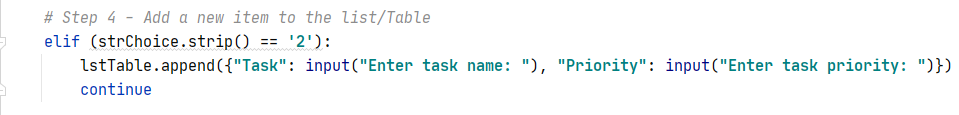


Figure 5: Choice 2 code

## Step 5: Remove a Task

If the user entered “3”, the user was then asked to enter the name of the task to be removed as shown in Figure 6. Each task row was compared to the user’s input. If the task was found, the row was removed, the user notified, and the list printed to show the remaining tasks. If the task was not found, the user was notified and the list printed to show all tasks.

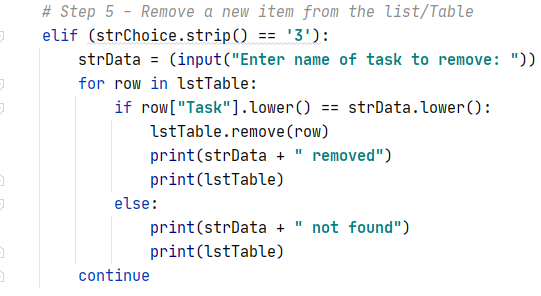


Figure 6: Choice 3 code

## Step 6: Save the Data

If the user entered “4”, the program would save the previously input data to the ToDoList text file with each row on a new line. The file was opened in Write mode rather than append because all of the data had been loaded into memory (see Figure 7).

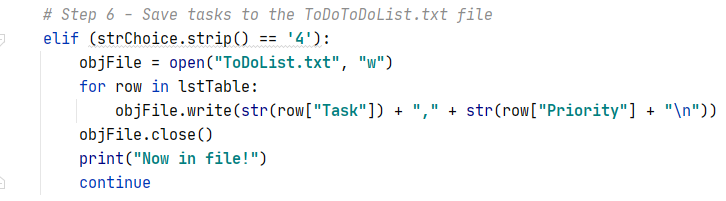


Figure 7: Choice 4 code

## Step 7: Exit the Program

Finally, if the user entered “5”, the program prints “Goodbye” and exits as shown in Figure 8.

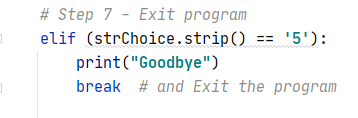


Figure 8: Choice 5 code

# Testing the script

To see whether the code functioned as intended, I first ran it in PyCharm. Since the text file ToDoList.txt did not exist, the program gave the error shown in Figure 9.

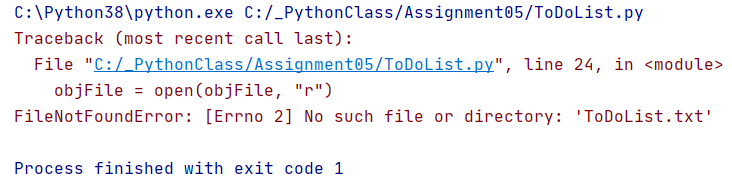


Figure 9: Missing file error

Rather than creating a placeholder text file, I imported os.path and checked to see if the file existed. If it does, the data is read from the file. If not, the menu is displayed directly as shown in Figure 10.

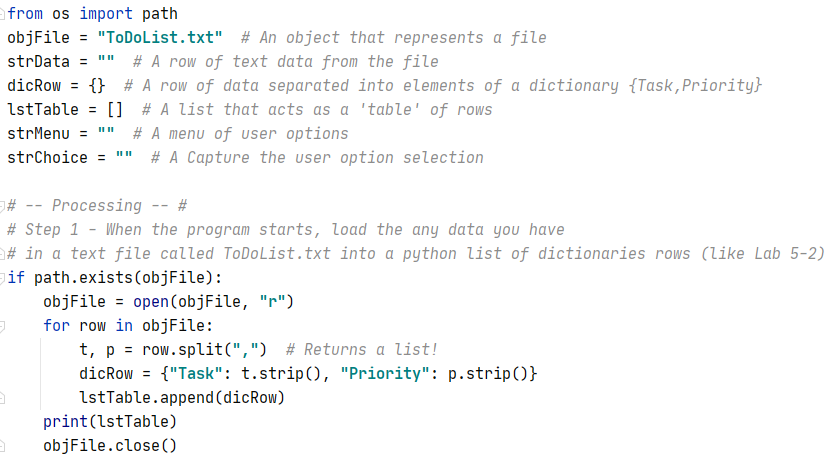


Figure 10: Code to check if file exists

Choosing “1” with no existing text file gave the following result (see Figure 11):

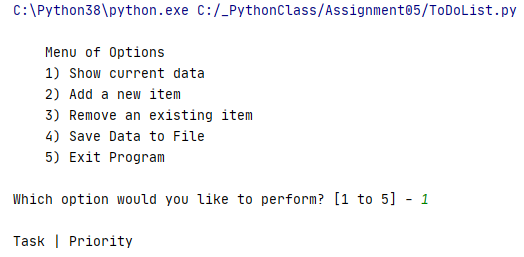


Figure 11: Show current data - PyCharm

When “2” was entered, the program requested a task name and a priority, then returned to the menu. This time, option “1” displayed the data entered by row as key-value pairs as shown in Figure 12.

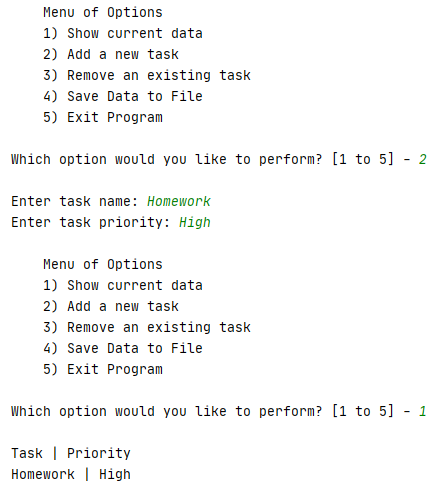


Figure 12: Add a new task - PyCharm

Next, I entered “4” to create the text file and save the data as shown in Figure 13.

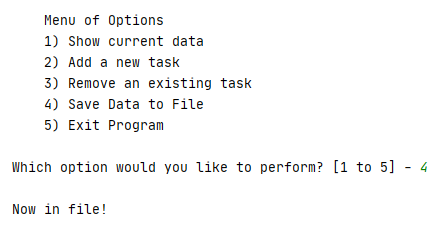


Figure 13: Save Data to File - PyCharm

I used option 5 to exit the program so that I could check that the text file was created (see Figure 14).

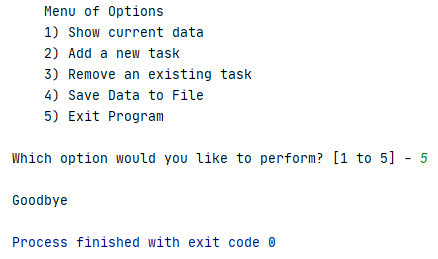


Figure 14: Exit Program - PyCharm

The ToDoList.txt file was created in the working folder as shown in Figure 15.

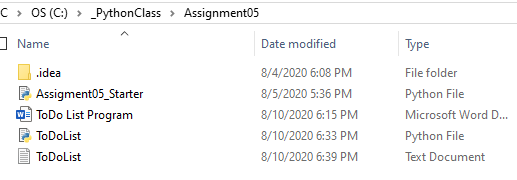


Figure 15: ToDoList.txt file

The data showed in the text file as expected (see Figure 16).

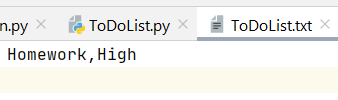


Figure 16: Text file data

Finally, I ran the program from the command window. This time, the text file already existed and the program printed the imported data list. I also used option 3 to remove the Vacuum task as shown in Figure 17.

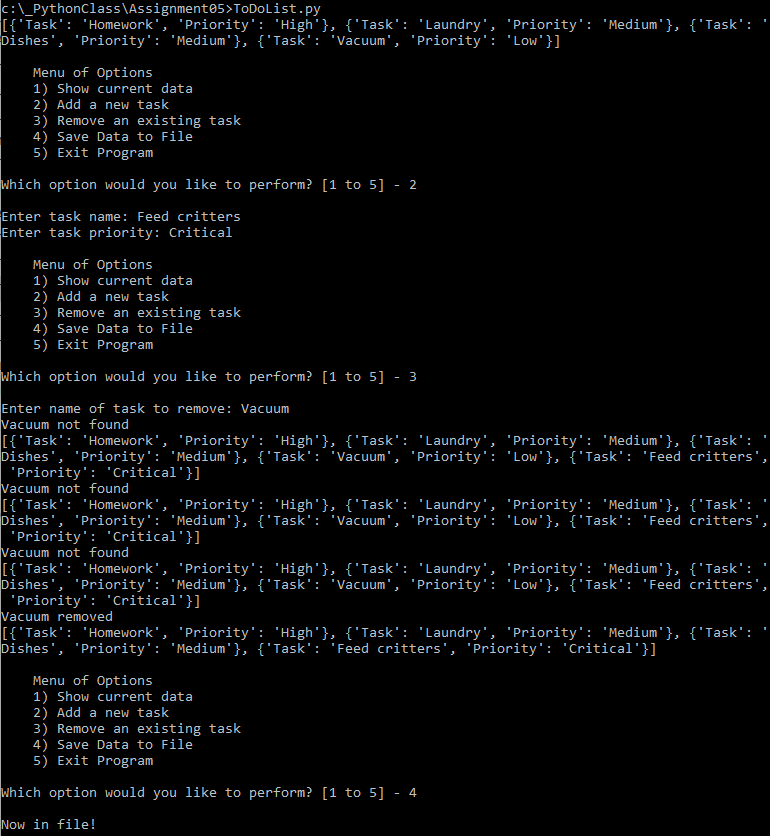


Figure 17: Running the program in the command window

The final results in the text file are shown in Figure 18.

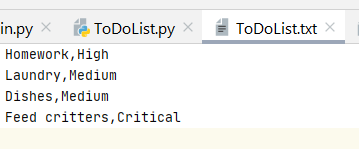


Figure 18: Text file data

The program worked per the requirements of the assignment, but the removal function could be further modified to only print a message for the specific row requested.

# Summary

Using the textbook and the Module 5 documentation, I created the ‘To Do List’ program and successfully ran it in PyCharm and the OS command window.