Nick Hertlein

February 25, 2021

IT FDN 110 B: Foundations Of Programming: Python

Assignment06

GitHub URL: https://github.com/nhertlein/IntroToProg-Python-Mod06

Functions & Classes

# Introduction

This document will illustrate the knowledge gained from the sixth lecture and the media portion of the third assignment. In addition, I will cover the modification of an existing Python script that manages a “ToDo list” based on user selection. The program will give a number of different options to allow the user to read from a current saved to do list, see what is currently in the to do list, add a task to the list, remove a task from the list, and write the list to a text file. One difference with this assignment is that we had a “starter” file with some comments to guide us, but also ensure we followed a specific structure. My assignment is late because I had a brutal week at work, please have mercy ☺

# Writing the To Do List Script

For this week’s assignment I made a project in PyCharm in the Assignment06folder of the C:\\_PythonClass directory. As we were starting with a “starter” file for this assignment the first step included updating the header change log.

The first section of the script was labeled “Data” and there was a comment to use that space to declare variables and constants. There were some variables pre-established for us to use, so I kept those for consistency. See Figure 1 for variable and constant layout.

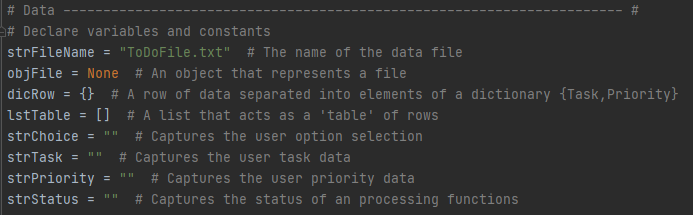


Figure 1. Declaration of variables

As this module was intended to teach us about functions and classes the next section of the script was a class defined for processing tasks to be used through the program. The first method for the class was read\_data\_from\_file and it was already populated for us in the starter file. The next method was add\_data\_to\_list which needed to be filled out. I started by creating the header for the function to document what the parameter inputs and functions outputs would be for reference. For the code section I reused some code from the for loop used for reading in data to keep the style similar (Figure 2).

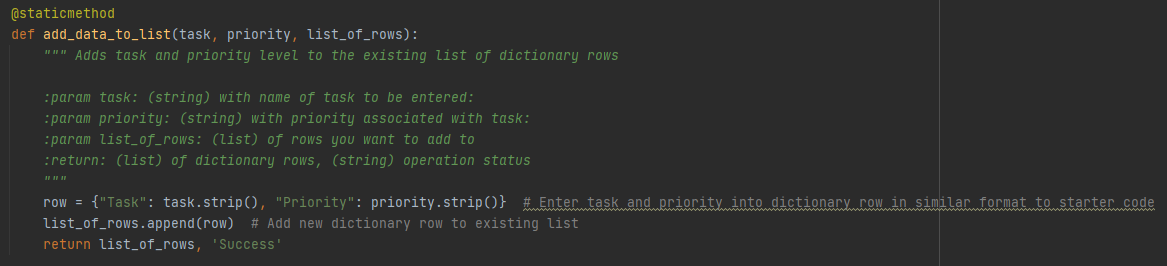


Figure 2. add\_data\_to\_list method

The third method in the processing class was remove\_data\_from\_list which also needed to be populated. I created the header for the function with the relevant information related to inputs and outputs for future reference. The block of code structure was copied from my Assignment05 as we had a similar task and I modified it to use the preferred variable names for the function (Figure 3).

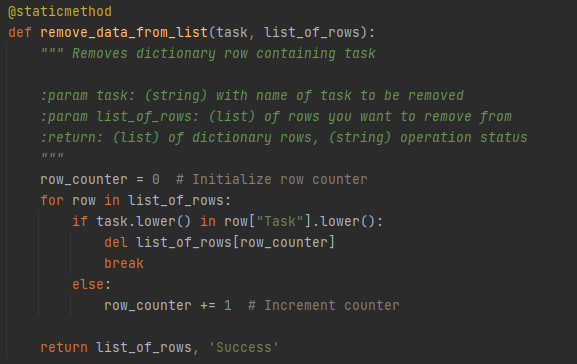


Figure 3. remove\_data\_from\_list method

The fourth method in the processing class was write\_data\_to\_file which needed to be populated. I created the header for the function similar to the previous methods. The block of code for this method was also copied from Assignment05 and modified to use the preferred variable names for the function (Figure 4).

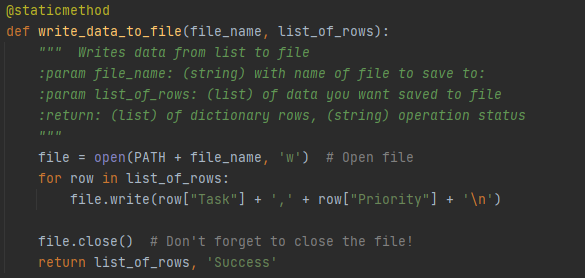


Figure 4. write\_data\_to\_file method

The next section of the script was focused on input/output and included a class to combine all the IO functions into one group. Many of these methods to the IO class were already populated so I will not cover those here. The first method we had to populate was input\_new\_task\_and\_priority. I first filled out the header section as was done on previous functions. For the code section I made an input for the task and one for priority. Both inputs are converted to string which is unnecessary as the default output from the input function is a string, but I did this for clarity. The task and priority are output as a tuple in the correct order to align with the list they may be added to (Figure 5).

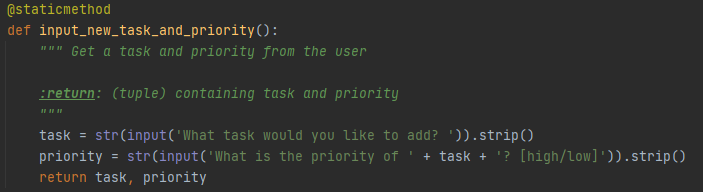


Figure 5. input\_new\_task\_and\_priority method

The last method in the IO class is input\_task\_to\_remove which needed to be populated. I first filled out the header section as was done on previous functions. For the code section I just made a simple input question asking which task needed to be removed from the list (Figure 6). I did not run any checks to confirm the task was included in the list as this function just passes the task to be removed to the remove\_data\_from\_list method of the Processing class where the user will be notified if the task cannot be found in the list.

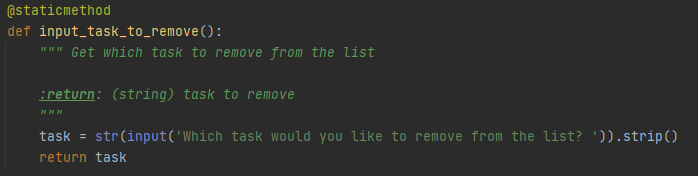


Figure 6. input\_task\_to\_remove method

The last section of the script was the main body of the script which was used to call the other functions at the appropriate time to help keep the code clean and easier to read. The main feature of the section was a while loop that repeats to allow the user to make selections until they chose to exit. Another feature of the loop is it displays the current list of tasks and priority with every iteration so the user can see the current list and make a selection appropriately.

The first menu option allowed the user to make a new entry to the list. When this option is selected the user is prompted to enter a task and priority by calling the IO method input\_new\_task\_and\_priority. The IO.input\_new\_task\_and\_priority method returns a tuple of the user entered task and priority. The tuple is unpacked into variables specified in the starter file at the output of the function so the task and priority can be referenced independently. The outputs from the IO function are then input into a Processor method add\_data\_to\_list which then takes the inputs and adds them to the list. The Processor.add\_data\_to\_list function outputs the appended list and the status of the operation. The status is then fed to the IO function input\_press\_to\_continue so the user knows the operation was successful and waits for user input to continue (Figure 7).

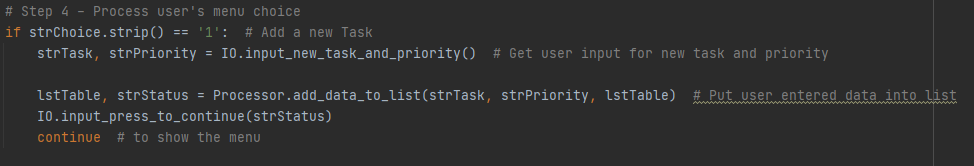


Figure 7. Add new task menu option

The second menu option allowed the user to remove a task from the list. When this option is selected the user is prompted to enter a task to remove by calling the IO method input\_task\_to\_remove. The IO. input\_task\_to\_remove method returns a string of the user entered task to remove. The output from the IO function are then input into the Processor method remove\_data\_from\_list which removes the task from the list if it can be found. The Processor.remove\_data\_from\_list function outputs the updated list and the status of the operation (Figure 8). Ideally if the task could not be founf in the list I would have updated the status of the operation to be “failed” with an error message, but I was already behind so I didn’t add this feature.

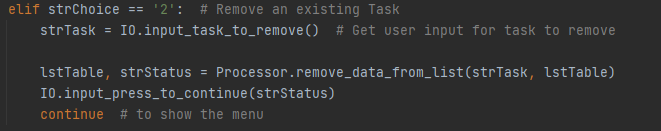


Figure 8. Remove existing task menu option

The third menu option allowed the user to save the list to a file. When this option is selected the user is asked to enter a response to the yes or no question of whether they want to proceed with saving the list to a file through the IO method input\_yes\_no\_choice. The IO. input\_yes\_no\_choice method displays the question passed to the function as an input parameter so the same function can be used for multiple different questions and it returns a string of the character the user entered. If the user entered anything other than “y” the user is notified the save is cancelled and is returned to the menu of options the user can select. If the user enters “y” the Processor method write\_data\_to\_file is called which writes the data to the specified file name. The Processor.write\_data\_to\_file function outputs the list saved to the file and the status of the operation (Figure 9).

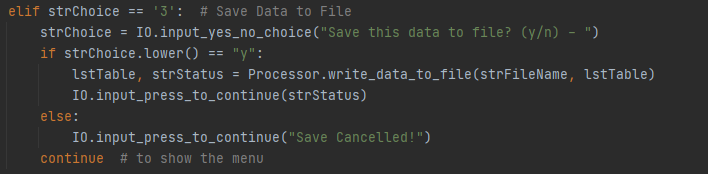


Figure 9. Save data to file menu option

The fourth menu option allowed the user to reload a list from the file. When this option is selected the user is given a warning letting them know that reloading the list will erase any changes currently made. After this the user is asked to enter a response to the yes or no question of whether they want to proceed with reloading the original list through the IO method input\_yes\_no\_choice. The IO. input\_yes\_no\_choice method displays the question passed to the function as an input parameter so the same function can be used for multiple different questions and it returns a string of the character the user entered. If the user entered anything other than “y” the user is notified the reload is cancelled and is returned to the menu of options the user can select. If the user enters “y” the Processor method read\_data\_from\_file is called which reads the list from the specified file. The Processor.read\_data\_from\_file function outputs the list read from the file and the status of the operation (Figure 10).

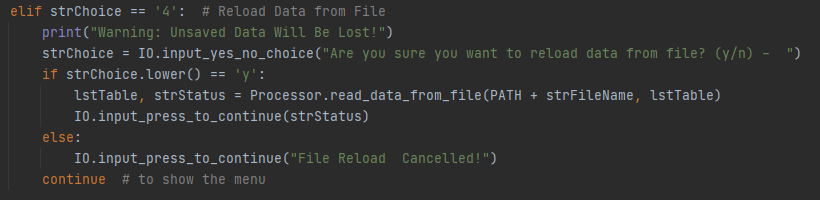


Figure 10. Reload data from file menu option

The final user option is an option to exit the program. This option features notifying the user the program will be exited and breaking out of the while loop.

I ran the program initially in the PyCharm environment to make sure the program would work how I intended to add entries to the list (Figure 11), remove entries from the list (Figure 12), save data to the file (Figure 13), and exit program (Figure 14). The script was a success and my entries were saved to the ToDoList.txt file in the Assignment06 folder (Figure 15).

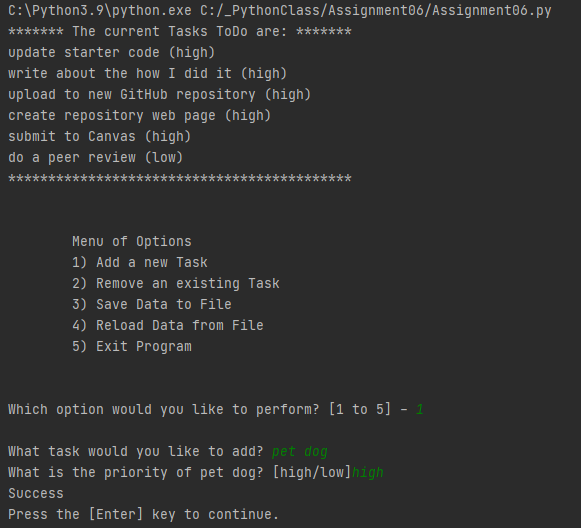


Figure 11. Adding entries to the list in PyCharm

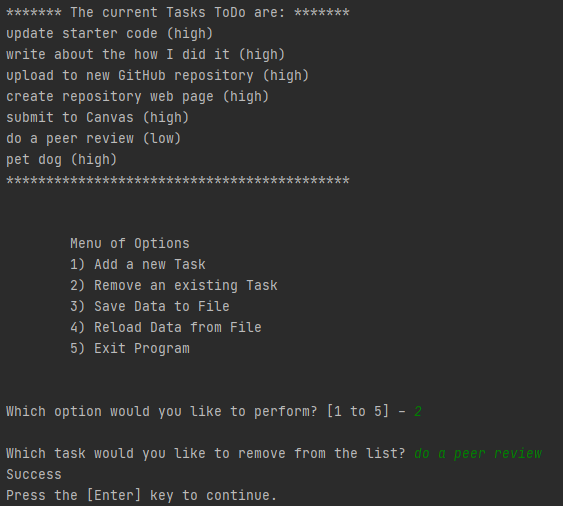


Figure 12. Removing entries from the list in Pycharm

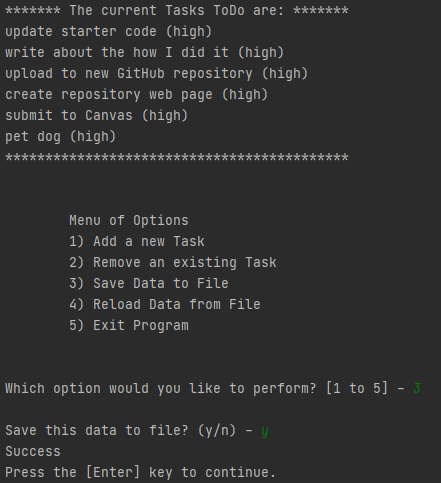


Figure 13. Saving list data to file in Pycharm

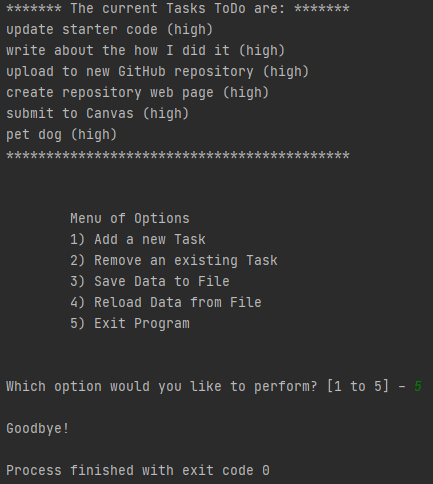


Figure 14. Exit program in PyCharm

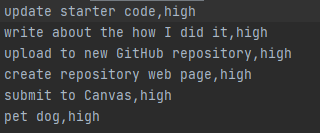
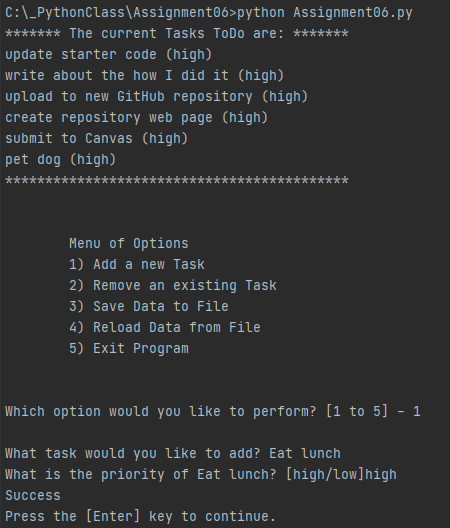


Figure 15. ToDoList output after saving in PyCharm

After running the program successfully in PyCharm I wanted to make sure it would execute correctly from the system terminal. I accessed the terminal through PyCharm and made another inventory list (Figure 16).



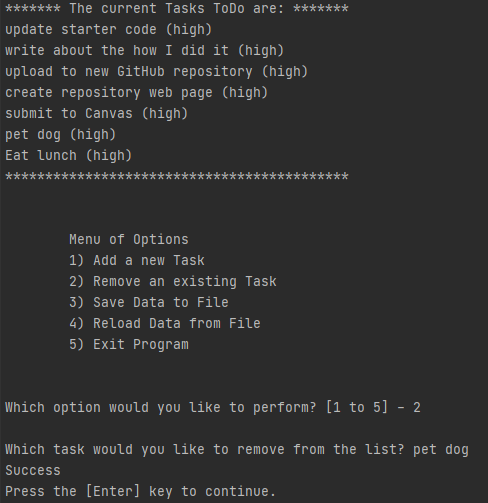


Figure 16. Running the program in the Terminal

# Summary

This module was intended to be an introduction to functions and classes. This was an extension of the last assignment where we broke the existing code into classes of functions in order to keep the code split into areas of concern. This module was helpful as it showed the advantages of making classes and functions to make the code more readable and reusable for more than one task. I had never used classes before, so this was a good experience to learn how to use them!