Alexander Bryant

5/24/2020

Foundations of Programming

Assignment 06

Improving the To-Do List with Functions

Intro

This week’s module focused on defining and using functions and classes within Python. This week’s assignment was to update a code starter using functions to improve the code’s readability and separation of concerns. The core challenge of this assignment is to generalize the code within Assignment 05 to work as functions within an existing script.

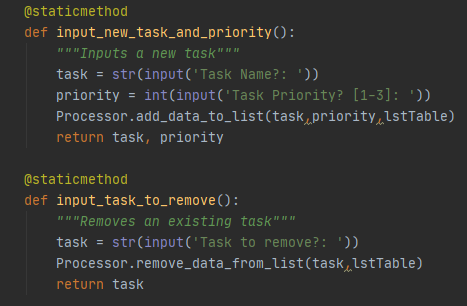
Process

In line with the principle of separation of concerns, the code starter was split into data, processing, presentation, and main body sections. The data section outlines variables that are used throughout the script, the processing section contains the inner mechanics of the script, the presentation section contains functions that serve input and output roles, and the main body of the script is where the functions and processes are executed.

******

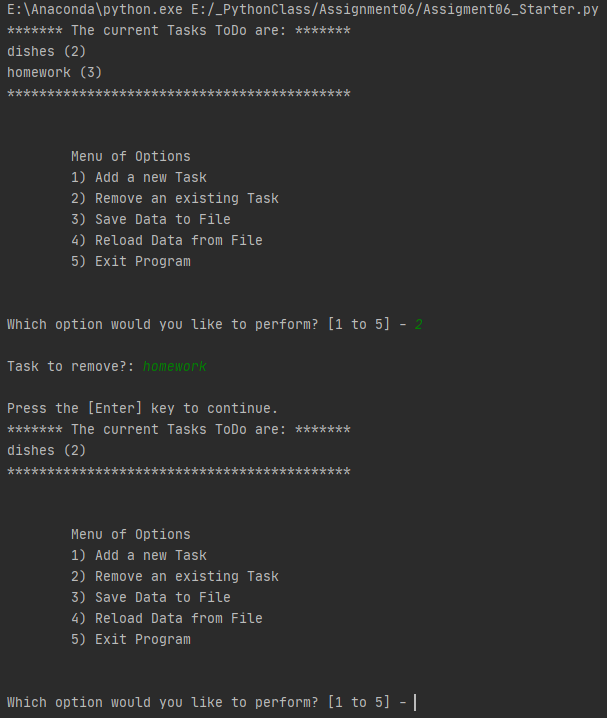
***Figure 1: The processing section contains custom processing functions with docstrings.***

In ***Figure 1*** above, three new functions are created within the Processor class. The static method tag means that the functions can be called class first, as in ‘Processor.add\_data\_to\_list()’ for example. The three functions above in ***Figure 1*** generalize the tasks completed in Assignment 05’s To-Do List program. The parameters within these functions are filled in and populated by other functions within the presentation section. In ***Figure 2*** below, new functions within the IO class request user inputs, which are then fed into the parameters of the functions from the processor class, which then process the data.



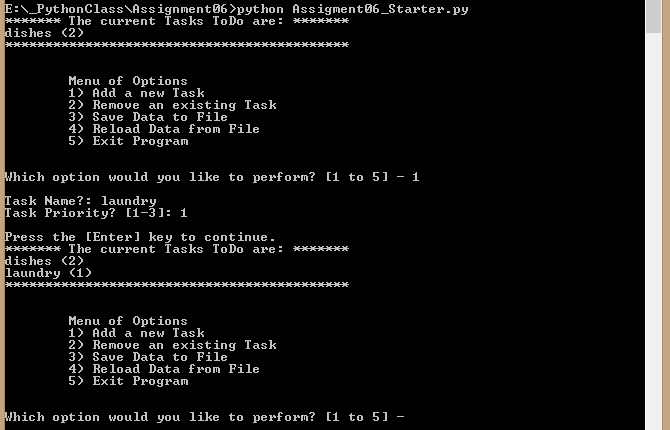
***Figure 2: Functions within the Presentation section request user inputs.***

In effect, the user experience remains mostly the same. In ***Figure 3*** below, a PyCharm user opens the program. The contents of ToDoFile.txt are automatically read and presented to the user, in this case ‘dishes’ and ‘homework’. When the user inputs menu option 2, the program calls up IO.input\_task\_to\_remove(), which then requests a task name. This task name is then fed into Processor.remove\_data\_from\_list() as a parameter, which then goes in and uses a for loop to find and delete the appropriate task. The program requests the user press enter to continue, displays the to do list’s new contents, and then presents the user the menu once again. By splitting the removal process into two functions, it becomes easier to make small updates to the code if necessary. If the user needed to change the lstTable variable to something different, such as lstTable\_2, instead of changing the processing mechanics of the removal function, the updates can be made to the parameters of the functions within the presentation section.



***Figure 3: In PyCharm, the program runs the same as before but now relies on functions.***

Once the user saves the data with menu option 3 and exits the program with menu option 5, the text file is updated using the new function. When the same user realizes that laundry is now the top priority, the process works the same within the command line. In ***Figure 4*** below, the program boots up with only dishes on the to-do list. When the user submits menu option 1, the presentation function IO.input\_new\_task\_and\_priority() collects a task and priority (‘laundry’,1), which are sent to the processing function Processor.add\_data\_to\_list() as the parameters ‘task’ and ‘priority’.



***Figure 4: The user adds a new task from the command line.***

Summary

While the concept of separation of concerns is fairly straightforward, aligning with this best practice can be difficult. While I have previously defined functions that utilize other functions, Assignment06 was the first time I have used this technique specifically to split code into data, processing, and presentation sections. The end result is a very well organized code with well defined sections, and I can imagine that this program will serve as an important reminder of well executed separation in the future.