Kristin Landgren Martinez
December 1, 2021
Foundations of Programming: Python
Assignment 06
GitHub Page
GitHub Repo

Functions

Introduction

This assignment introduces functions and classes. Though the objective of the Assignment06 script is identical to Assignment05, the code is rearranged into two classes of functions and organized into separate sections for data, processing, and presentation. For outward presentation, this module includes creating a new GitHub repo and web page.

Using Functions

A function is a group of one or more statements that are defined by a name. These statements are executed when the function is called later in the script. If the function takes in values for processing, these values are called parameters or arguments. The values that a function generates as a result of its code are called return values. There can be one to many, and they can be captured in variables for use elsewhere in the script.

Variables used inside a function are only visible in that space, not in the rest of the script: these are local variables. Global variables, in contrast, are defined in the main body of the script (ideally in the Data section), and can be used anywhere in the script. If used inside a function, global variables need to use the keyword “global” to avoid shadowing the global variable.

Using functions helps to organize code in a number of ways. Used in a script that follows the “Separation of Concerns” structure, they allow you to separate out processing code from presentation code. For example, in this assignment, code sections that had originally contained both a presentation component (ask the user which ToDo task to remove) and a processing component (remove the corresponding row) were broken into two functions. Down the line, these modular pieces could be useful in other programs and could easily be reused. Using functions and a defined structure also makes it easier for others to follow the code, particularly with the aid of built-in docstrings to describe each function. Classes, which group functions together, further aid in organizing a script and making it logical for others.

As scripts become more complex, debugging tools in PyCharm can be useful in understanding errors in the code, the contents and data types of variables, and more. The tools can be accessed alongside the Run option in the menu. The debugging tools can “step into” code so you can review it line by line—a very helpful feature. You can also set a breakpoint where you want the debugger to stop so you can check the code.

This document and the files for this assignment are posted on a new GitHub repo. This module introduced GitHub Pages for presenting content on a website: the site and the repo are both linked to in the header. The GitHub page makes it easy to present repo contents (or any other text and information) in a slick, public-facing web page.

To-Do List Script 2.0

The ToDo list script for Assignment06 is based on a starter file. The purpose of the program is identical to Assignment05, but the goal in this case is to organize and add to the code so that it uses functions and follows the Separation of Concerns structure.

Functions

It took me some time to read through the stater code and understand how the various functions were going to be called in the main body of the code, and which variables were being used. I then started bringing in portions of code from my Assignment05 script. For each user selection, I copied the processing portion to the corresponding function in the Processor class, and the presentation section to the IO class, and updated the variable names as needed. I also added docstrings for each function, based on those provided in the starter file.

Main Body

The main body of the code uses a while loop and if/elif statements for each user selection, just like Assignment05. However, it calls the Processor and IO functions defined earlier. I worked through this one selection at a time, after bringing in the functions for that selection. For the "add an item" option, I saved the return values from the IO function as variables and used those variables as arguments in the Processing function, as shown below:

```
if choice_str.strip() == '1':  # Add a new Task

    task_add, priority_add = IO.input_new_task_and_priority()
    # Prompt for new task & priority, upnack tuple

    Processor.add_data_to_list(task_add, priority_add, table_lst)
    # Use input values as arguments

    continue  # to show the menu
```

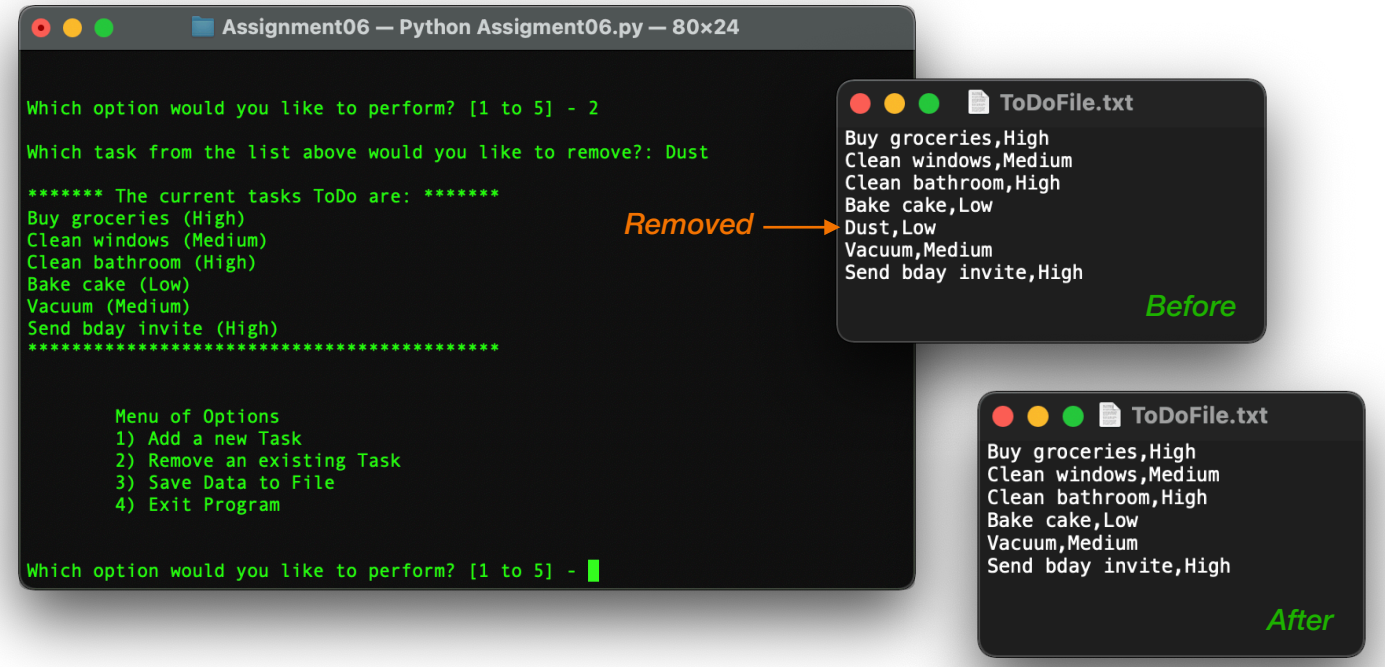
Debugging & Running

For each user selection pair that I brought in, I tried running the script with that user selection. A few worked off the bat, but several required some debugging. An error in the function remove_data_from_list was due to not capitalizing the dictionary key, and in the main script I initially neglected to replace list_of_rows with table_lst. Because the code itself had worked in the previous script, the troubleshooting focused more on how to arrange the code in functions and call the functions properly in the main script.

Once each user selection option worked as expected, the script ran smoothly in both PyCharm and in Terminal.

Figure 1: Script running on PyCharm, with txt file before and after

Figure 2: Script running in Terminal, with txt file before and after



Conclusion

Organizing scripts using functions, classes, and the Separation of Concerns structure allows them to be far more re-usable, scalable, and standardized. In Assignment06, I learned how to re-organize code into this structure, and in the process thought more about the purpose of individual parts of the code. This module also introduced GitHub Pages to share and present files, in addition to the assignment repo.