Spencer Neiffer

ITFDN110A

Assignment 06

https://github.com/SBNEIFFER/Python110_2025.git

Functions

Introduction

This document indicates the steps I took in separating my script into functions and classes. Prior to this module, my script was fully functional but only consisted of the logic with no organization. To solve this, I used the logic for each portion of the script to define individual functions, then sorted them between two different classes. This gave me the ability to call the function whenever needed, instead of using the actual logic. This abstraction is known as Separation of Concerns.

Classes

The first step I took in implementing Separation of Concerns within my script was creating classes to store my functions in. Since my script consists of either reading and writing to a JSON file, or capturing and displaying user input, I created two classes. The first class I called ***FileProcessor***, in which I stored my functions that directly interact with the JSON file. The second class I called IO, and here is where I stored my functions that capture input and output regarding the user.

Functions

The first function I defined reads data from the JSON file, and I called it ***read_data_from_file***. I gave it global variables of ***filename***, ***students***, and ***file***, then put the existing logic directly beneath this code. After placing it withing the ***IO*** class, I called the function at the bottom of the script and added a ***while True*** loop to begin the main body of the code. I followed this same method for the remaining existing code and used the following function definition names and classes:

write_data_to_file - FIleProcessor

input_menu_choice - IO

output_menu - IO

input_student_data - IO

output_student_data - IO

output_error_message – IO

At this point, the main body of my script contains the function calls in order that they occur in the program.

Parameters and Arguments

The next step I took in implementing SoC within my script was replacing my functions' global variables with parameters in the function definition portion. I put the following parameters in the parenthesis for the functions below:

write_data_to_file – file_name, student_table

read_data_from_file – student_table, file_name

input_student_data – student_table

output_student_courses – student_table

input_menu_choice – none

output_menu – menu

output_error_messages – message, exception

After this, I added arguments to each called function in the main body of my script. The values passed to each function are as follows:

file_name = FILE_NAME

student_table = students

menu = MENU

Finally, I added the class location to the beginning of each function call, so the program knows where to find the functions.

Error Handling

After adding parameters and arguments, I updated the error handing within each function definition, so the ***output_error_messages*** function is used to display custom error messages to the user. Finally, I added docstrings to each function, describing what they do, who created them, and when they were created.

Summary

In conclusion, much of the code for this script already existed and I used it to define functions in the header of the script. I then gave parameters to those functions and called them in the main body of the script using arguments. I also sorted my functions into classes using an abstraction method called Separation of Concerns and added the class to the called function, so the program knows where they're located. Finally, I added docstrings to each function indicating its purpose, who created them, and when they were created.