Written Assignment 06

Alex Harteloo

Feb. 11, 2024

IT FDN 110A

Assignment 05

https://github.com/harteal/IntroToProg-Python-Mod06

Introduction to the Standard Organization of Code

Introduction

In this week's module, we explored functions, classes, parameters, and basic code organization strategies. Functions allow for code reuse and enhance the modularity of a program. A parameter is information fed into a function, enabling the use of local variables and encapsulating data. Classes bundle similar functions together, improving program modularity and maintainability. The Separation of Concerns (SoC) is a design principle that divides code into distinct, self-contained components, enhancing code manageability.

Our assignment required us to create a program displaying a student's registration message, similar to last week's task. This time, we had to incorporate custom functions and classes, and apply SoC principles. The program also needed to include previous concepts like reading and writing to JSON files, error handling, and custom exception messaging.

Topic 1 - Python Functions

A function is a defined segment of code that declares its purpose and is later referred to by a chosen name. Functions enhance modularity by breaking down large programs into manageable portions. Once defined, a function can be reused throughout the program, increasing development efficiency and reducing redundancy.

In this week's program, we defined seven functions, listed in Fig. 6.1. The functions read_data_to_file() and write_data_to_file() are under the FileProcessor

class, while the rest are under the IO class. Each function is defined after declaring the program's variables and constants.

Initially, I had an issue with the function read_data_from_file(). It was not extracting information from Enrollments.json to the students variable. After debugging, I realized I had forgotten to include a return statement, which captures a value in a variable and returns it to the program. This makes the function act like an expression. Following the error messaging, I added a return statement for student_data.

Class	Function Name	Parameter(s)	Description
FileProcessor	read_data_to_file()	file_name: str, student_data: list	Takes student data and places into a file in JSON format.
FileProcessor	write_data_from_file()	file_name: str, student_data: list	Reads a JSON file and extracts its data as 'student_data.'
IO	output_menu()	menu: str	Presents menu to user.
IO	input_menu_choice()		Records the user menu choice.
IO	output_error_message()	message: str = ... , error: Exception = None	Presents user with error message.
IO	input_student_data()	student_data: list	Records user inputted student information.
IO	output_student_data()	student_data: list	Presents student information to user when presented a list.

Fig. 6.1 - Functions found in Assignment06.py.

Regarding parameter usage in this assignment, parameters serve as local variables, used only within the function and while its active. This approach enhances code readability and minimizes the risk of passing incorrect arguments to a function. A notable parameter usage is providing default values to a function when an argument isn't provided. You can see this in the output_error_message() function. This function displays a custom error message if an error occurs.

However, if there's no error, no message is shown. This feature is enabled by setting the error parameter to None, hence establishing a "default state" of no error.

```
@staticmethod
def output_error_message(message: str = "There's a non-specif
ic error!", error: Exception = None):
    # Presents error message to user and technical error mess
age.
    print(message, end="\n\n")
    if error is not None:
        print("--Technical Error Message--")
        print(error, error.__doc__, error.__str__(), sep= " |
")
```

Fig. 6.2 - Code for the function output_error_message().

Topic 2 - Classes

A class is a method of grouping similar functions together. It's a highly effective way to organize code as it pairs functions with the data they manipulate, which in turn improves the structure and readability of larger projects.

In my assignment, I utilized two classes: FileProcessor and Input/Output (IO). Details of each can be found in Figure 6.3. With these classes, creating the main body of the program was significantly simplified. Once each function was defined and working correctly, completing the main body was fairly straightforward. This experience has deepened my appreciation for modularity, particularly in the context of large-scale applications we use daily.

Class	Description
FileProcessor	*A collection of functions that process files in JSON format.*
IO	*A collection of presentation layer functions that manage user input and output.*

Fig. 6.3 - Classes and their description in Assignment06.py

Topic 3 - Separation of Concerns

The final topic of this week's module introduced the principle of Separation of Concerns (SoC). The primary idea of SoC is to divide code into separate, self-contained components, each responsible for a specific area of a program's functionality. The functionality can be categorized into three domains: 1) presentation concerns, 2) logic concerns, and 3) data storage concerns. Presentation involves elements that the user can see or interact with, logic manages the main functionality of the application, and data storage handles the management of data, from databases to system files.

In my program, functionality is divided into these three layers of concern. The FileProcessor class, which resides in the data access layer, primarily reads and writes files to "Enrollments.json," aligning with the data storage domain. The logic concerns, or processing layer, is where the main body of the program exists. In my program, this is where the application layer's while loop runs. Finally, the presentation layer houses the IO class and its functions, capturing user input, such as student information. This information is then passed to the processing layer for cleaning and housing before returning to the presentation layer for user display.

Summary

The assignment covers the organization of code in Python, focusing on functions, classes, parameters, and the principle of Separation of Concerns (SoC). Functions improve code modularity and reusability, while parameters allow for local variables and data encapsulation. Classes bundle similar functions together for better code management. SoC breaks code into self-contained components for easier management. The assignment involved creating a program with custom functions, classes, and applying SoC principles, demonstrating these concepts in practice.