Jett Magnuson

November 18, 2025

Functions, Classes, and Separation of Concerns

Assignment06

Functions, Classes, and Separation of Concerns

Introduction

This assignment builds on assignment05 by reorganizing the course registration program using functions, classes, and the separation of concerns design pattern. Instead of having all the code in one long script, the program now separates different responsibilities into two classes: FileProcessor for handling file operations and IO for managing user input and output. All functions use the @staticmethod decorator and include descriptive docstrings to explain their purpose.

Understanding Classes and Static Methods

The main change in this assignment was organizing the code into two classes that separate different JSON files, while IO class handles all the user interaction like display menus and collecting student information. Using @staticmethod decorators means these methods don't need to create class instances to work-they're called directly like IO.output_menu or FileProcessor.read_data_from_file(FILE_NAME,students). This approach makes the code more organized because each class has a clear purpose, and you know exactly where to look when you need to modify how files are handled or how user input works.

```
27  class FileProcessor:
28      """A collection of processing layer functions that work with JSON files"""
29
30      @staticmethod
31      def read_data_from_file(file_name: str, student_data: list):
32          """Reads data from a JSON file and loads it into a list of dictionary rows"""
```

Separation of Concerns Pattern

Separation of concerns means dividing a program into distinct sections where each section handles one specific responsibility. In this program, FileProcessor is responsible for all file operations (reading and writing), IO is responsible for all the presentation tasks (displaying information and getting user input), and the main body of the script handles the program flow and menu logic. This pattern makes the code easier to maintain because if you need to change how the data is displayed, you only modify the IO class without touching the file handling code. It also makes debugging simpler since you can quickly identify which class is responsible for a particular feature.

Function Design and Error Handling

Each function in the program has a single, clear purpose and includes a descriptive docstring explaining what it does. The output_error_messages() function particularly is important because it provides a centralized way to display error throughout the program. Instead of repeating the same error-handling code multiple times, every function with exception handling calls IO.output_error_messages() to display both custom messages and technical error details. This makes the code cleaner and ensures consistent error messaging across the entire program.

```
64      @staticmethod
65      def output_error_messages(message: str, error: Exception = None):
66          """Displays a custom error message to the user"""
67          print(message)
68          if error:
69              print("-- Technical Error Message -- ")
70              print(error.__doc__)
71              print(error.__str__())
72
```

Reflection

This assignment taught me how to structure programs using classes and functions to create more maintainable code. The separation of concerns pattern makes it much easier to understand what each part of the program does and to make changes without affecting other

parts. Using static methods and organizing code into FileProccessor and IO classes helped me see how professional programs are structured with clear boundaries between different responsibilities. These organization skills will be valuable for creating larger programs where keeping track of everything on one long script would become unmanageable.

GitHub Repo: https://github.com/slimbiggins007/IntroToProg-Python-Mod06.git