Luis Blas
Agust 7, 2024
IT FDN 110 A

Python Adventures: Mastering Separation of Concerns, Functions, and Classes

Introduction

I explored advanced Python concepts for creating well-structured and clean code this week. This week focuses on the Separation of Concerns (SoC) pattern, using functions effectively, and understanding classes. This topic will help in recreating the script of the previous assignment, allowing for a more modular and concise script, which enhances the overall quality of our work.

Separation of Concerns Pattern

This topic is interesting since it is less about developing your technical skills and more about building your programming brain. Your programming brain is your personality as a programmer; there are principles that every programmer shares, but the implementation of these principles is unique. The separation of concerns (SoC) pattern is a fundamental design principle that involves dividing a program into distinct sections, each handling a specific responsibility. Allows us to improve code organization, making it easier to maintain and debug, which reduces the task for updates or modification. My approach to our script is to break down the code into modules that address specific actions, i.e., processing file or input/output, but you are welcome to take a different approach as you build your brain.

Modularization Example

```
# Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        # We reduce the line in the section through the SoC pattern
        student_data = IO.input_student_data()
        if student_data:
            students.append(student_data)
            print(f"You have registered {student_data['FirstName']}
```

```
{student_data['LastName']} for {student_data['CourseName']}.")

    # Present the current data
    elif menu_choice == "2":
        IO.output_student_courses(students)
```

Functions

Functions are nothing new; we saw them in the past but knew them as methods. Methods are a set of instructions we apply to an object, while functions are a set of instructions that perform a task. Function turns into a method when apply to a object. Functions are fundamental building blocks in Python, allowing for reusable code snippets that perform specific tasks by encapsulating logic within them. This aids in breaking down complex problems into manageable parts, making the code easier to test and debug. This means functions are essential to the SoC pattern as they allow us to break down our script, which, in our case, we will create functions/methods to read data, send output, read inputs, and many more.

Function Implementation

```
@staticmethod
    def output_student_courses(student_data: list):
        """ Displays the current student courses

        :param student_data: (list) the student data to be displayed:
        :return: nothing
        """
        print("-" * 50)
        for student in student_data:
            print(f'Student {student["FirstName"]} '
                  f'{student["LastName"]} is enrolled in
{student["CourseName"]}')
        print("-" * 50)
```

Classes

I believe the naming of this topic, “Classes,” is awful, but no matter that, classes are usually used to incorporate Object-Oriented Programming (OOP) in Python. OOP is a set of

principles that could make up your programming personality/brain based on SoC patterns. I won't delve into the details of this as students often get confused, and in our case, we won't implement OOP in our script. Instead, classes allow us to modularize our script by making our function in associate classes such as file processing and input/output, which promotes code reusability.

Class Implementation:

```
class IO:
    """  Performs Input and Output tasks """

    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        """ Displays error messages to the user

        :param message: (string) a custom error message to be displayed:
        :param error: (Exception) optional, an exception object with the
error details:
        :return: nothing
        """
        print(message)
        if error:
            print("-- Technical Error Message -- ")
            print(error.__doc__)
            print(error.__str__())
```

Summary

For Assignment 06, we discovered the Separation of Concerns pattern, which enhances code structure by dividing responsibilities and the start of developing our programming personality/brain. The importance of functions in creating reusable and maintainable code was discussed, and classes that allow for modularization in our Python scripts were delved into. By mastering these concepts, we can build more robust, scalable, and maintainable scripts, laying the groundwork for advanced Python programming.