Name: Parker Henry
Date: 11/26/24
Course: IT FDN 110: Foundations Of Programming: Python
Github Link: link here

Assignment 07: Implementing Data Classes with Inheritance and Encapsulation

Introduction

In this assignment, I worked on enhancing the course registration program by applying principles of object-oriented programming, specifically focusing on data classes, inheritance, and encapsulation. These concepts helped me to keep my code organized and efficient. I also got hands-on experience with Python's built-in methods like __*init*__ and __*str*__, as well as the *@property* decorator to ensure data validity.

Step 1: Creating the Person Class

The first step was to create a *Person* class that can store basic personal information, such as *first_name* and *last_name*. This class serves as the base information that the next class will “inherit: from. I learned to use the __*init*__ method to initialize elements with a first name and a last name. I also added error handling for \`first_name\` and \`last_name\` to make sure that they contain only letters. This protects the data and ensures it's formatted correctly.

Step 2: Implementing Inheritence with the Student Class

Next, I created a *Student* class that “inherits” from the *Person* class. This step was crucial in understanding inheritance, where the *Student* class automatically gains the attributes and functions inside the *Person* class. I used the *super* function in the *Student* class's __*init*__ function to initialize the inherited attributes. This showed me how inheritance can make my code more concise and maintainable.

Step 3: Adding the Course Name Attribute

In the *Student* class, I introduced the *course_name* attribute that is specific to students. I used the *@property* “decorator” again to create getter and setter functions for *course_name*. The setter method includes error handling to ensure that the course name isn't empty. This practice reinforced the importance of data validation and encapsulation in writing efficient programs.

Step 4: Overriding the __str__ Method

I learned to override the __*str*__ method in the *Student* class to include the course name when printing a student's details. The overridden method combines the output of the *Person* class's __*str*__ method with the *course_name*, providing a comprehensive string with all the information for a given student. This taught me how to make object information more user-friendly and informative.

Step 5: Defining Data Constants and Variables

I defined some constants and variables at the beginning of the code, like the *MENU* for displaying options to the user and *FILE_NAME* for the filename where data is saved. I also introduced *students* as a list to store student data. This step helped me organize my code better and understand the importance of having clear, defined constants and variables.

Step 6: Implementing Error Handling

Throughout the program, I incorporated error handling to manage incorrect inputs. This included checks in property setters to ensure valid data entries. It was a valuable exercise in anticipating potential user errors and ensuring that the program can handle them without crashing.

The rest of the code is reused from previous assignments.

Conclusion

This assignment helped me understand some important programming principles, like inheritance. I learned how to use classes to organize my code and make it more useful. I also got better at making sure my data is correct using the *@property* decorator. The module videos helped me understand these ideas better, and I feel more prepared for future programming projects. I also practiced handling errors and getting input from users, which are important skills in programming. Overall, this assignment helped me improve my coding skills and understand how to design programs better.