Crisandra Wilkie

3/12/25

Foundations of Programming: Python

Assignment 7

GitHub: https://github.com/cris-wilkie/IntroToProg-Python-Mod07.git

Introduction

Module 7 focused on creating a course registration program utilizing object-oriented programming (OOP) principles with data classes and structured error handling. This program allows a user to register a student for a course, view existing registrations, save data to a JSON file, and exit the program.

Key Components

The program uses a JSON module and is the first thing imported following the script header. Then the constants and variables are created, they include:

- MENU: A formatted menu string for user interaction.
- FILE_NAME: The JSON file used for storing enrollment data.
- students: A list to store student objects.

Classes

This program includes 4 classes that utilize encapsulation and inheritance. The first is the Person Class. This class contains the first_name and last_name attributes. The properties of first_name and last_name are added to a constructor. This class also uses the getter and setter methods to validate input, ensuring names contain only letters. This class overrides __str__() to return a formatted string.

```python
# TODO Add first_name and last_name properties to the constructor
def __init__(self, first_name: str = '', last_name: str = ''):
    self.first_name = first_name
    self.last_name = last_name

# TODO Create a getter and setter for the first_name property (Done)
@property  # (Use this decorator for the getter or accessor)  4 usages (2 dynamic)
def first_name(self):
    return self.__first_name.title()  # formatting code

@first_name.setter  3 usages (2 dynamic)
def first_name(self, value: str):
    if value.isalpha() or value == "":  # is character or empty string
        self.__first_name = value
    else:
        raise ValueError("The first name should not contain numbers.")
```

Figure 1. Person Class

The second class is the Student Class, this class is a subclass of the Person Class, indicated by referencing "Person" in parentheses following the class name, as shown below. This class inherits the first and last name attributes from the Person class, it also adds course_name, which represents the course the student is being registered for. This class overrides __str__() to include course_name.

```python
# TODO Create a Student class the inherits from the Person class (Done)
class Student(Person):  3 usages
    """
```

Figure 2. Student Class

The third class is FileProcessor. This class handles file operations using JSON and includes reading data from the Enrollments.json file and writing to this same file. These methods are called, read_data_from_file() and write_data_to_file. This class also implements exception handling for file errors.

The fourth class handles user input and output and is called IO. This class contains methods that allow for the menu to be displayed to the user (output_menu()), validates user menu selection (input_menu_choice()), displays all student registrations (output_student_courses()), takes the user input to register a student (input_student_data()), and handles and displays errors (output_error_messages()).

Main Loop

The program execution starts by opening and loading the student enrollment data from the JSON file. A menu is displayed, and the user is prompted to select 1 of 4 options, entering student registration information, viewing enrollment information, saving to a file, and/or exiting the program. The users' choices are handled using structured exception handling. All of this continues to loop until the user chooses to exit the program.

Conclusion

As in previous weeks, this program focuses on managing student course registrations in a simple and user-friendly manner. This script effectively demonstrates OOP principles, file processing, and user interaction while ensuring data integrity through error handling and validation.