Jett Magnuson

November 26, 2025

Data Classes and Object-Oriented Programming

Assignment07

Data Classes and Object-Oriented Programming

Introduction

This assignment expands on assignment06 by introducing data classes and object-oriented programming concepts to the course registration program. Instead of storing student information as dictionaries, the program will now use custom student objects that inherit from a person class. This assignment demonstrates how to create classes with properties, validation, and inheritance to organize data in a more structured way and reusable way.

Creating the Person and Student Classes

The main change in this assignment was creating two classes to represent data. A person class with the first and last name properties, and a student class with that inherits from person and adds a course name property. The person class serves as a base class containing common attributes that any person would have, while the student class extends it with student-specific information. Using inheritance means the student class automatically gets all the functionality from a person without having to rewrite the code. Each class includes an __init__() constructor method to initialize the object and a __str__() method that returns the data in a comma separated format.

```
30      def __init__(self, first_name: str = '' , last_name: str = ''):
31          self.first_name = first_name
32          self.last_name = last_name
```

Properties with Getters and Setters

Instead of using regular variables, this assignment required implementing properties with getter and setter methods using the @properties decorator. Properties allow you to control how data is

accessed and modified by adding validation logic in the person class. The first_name and last_name properties use setters to validate that names only contain letters using the isalpha() method, raising a ValueError if numbers are detected. The getters automatically format names using title() method so they're always capitalized properly. This approach provides data protection because users can't directly modify the private variables like __first_name without going through the validation in the setter methods.

Converting Between Objects and Dictionaries

A key challenge in this assignment was converting between student objects and dictionary data for JSON file storage. When reading from the file, the program loads JSON data as a list of dictionaries, then loops through each dictionary to create student objects using the dictionary values. When writing to the file, the program does the opposite, it loops through the list of Student objects and converts each one into a dictionary with "FirstName", and "CourseName" keys before using json.dump() to save the data. This conversion process is necessary because the JSON files can't directly store custom python objects, only basic data types like strings, numbers, lists, and dictionaries.

```
105     def write_data_to_file(file_name:str,  student_data: list):
106         """Writes data to json file with data from a list of student objects"""
107         file = None
108         try:
109             list_of_dictionary_data = []
110             for student in student_data:
111                 student_json = {"FirstName": student.first_name,
112                                 "LastName": student.last_name,
113                                 "CourseName": student.course_name}
114                 list_of_dictionary_data.append(student_json)
```

Reflection

This assignment taught me how to use object-oriented programming to create more organized and maintainable code. Using classes with properties and validation makes the program more robust by catching invalid data before it causes problems. The inheritance relationship between person and student shows how to create reusable code that can be extended for different

purposes without duplicating functionally. Understanding how to convert between objects and dictionaries is an important skill for working with file storage that requires specific data formats. These object oriented programming concepts will be valuable for building more complex applications that need to represent real-world entities with specific behaviors and attributes.

Github Repository