

Michael Okimura

05/26/2024

Foundations of Programming - Python

Assignment 07

<https://github.com/MOkimura/IntroToProg-Python-Mod07>

Getting and Setting for Class Inheritance

Introduction

This was another interesting learning curve assignment as I had some difficulty understanding the getter, setting, and “magic methods.” I spent extra time researching additional resources online to learn the reasons they are utilized. Going through another assignment utilizing classes has helped me understand that basic principles of them much better though. I can see classes are a very important part to writing programs. For this write-up, I will focus primarily on the new code we learned versus my previous write-ups where I essentially repeat the assignment requirements and that I simply followed them.

Creating the Program

Program Objective

Building on our last assignment, we’re adding class inheritance, setter/getter decorators, and an additional “magic method.” The results we’re looking for are still the same as previous assignments to accept user inputs for student and course name, multiple entries, displaying data, and reading/writing to a json file.

Coding the Student Class

The key parts of the assignment are consolidated into a new class we had to code which was the one for inheritance. The Student class inherited the Person class, passing through the input data from the Person class. Using the getter decorator in the screenshot of my code below allows us to define “course_name” as a method but access it like an attribute. Then we use the getter decorator because with the getter in place we will not be able to set “course_name” as the attribute.

```

65 # Student class inheriting Person class ----- #
66 class Student(Person):
67
68     # Call to Person constructor to pass first_name and last_name data
69     # Adding assignment to course_name property
70     def __init__(self, first_name: str, last_name: str, course_name: str):
71         super().__init__(first_name=first_name, last_name=last_name, course_name=course_name)
72         self.course_name = course_name
73
74     # Creating setter for course_name
75     @property
76     def course_name(self):
77         return self.__course_name
78
79     # Creating getter for course_name. Used replace method to remove "space" from triggering error in "isalnum" method
80     @course_name.setter
81     def course_name(self, value: str):
82         if value.replace(" ", "").isalnum() or value == "":
83             self.__course_name = value
84         else:
85             raise ValueError("The course name should not contain special characters.")
86
87     # Override __str__() method to return Student data
88     def __str__(self):
89         return f"{self.first_name},{self.last_name},{self.course_name}"

```

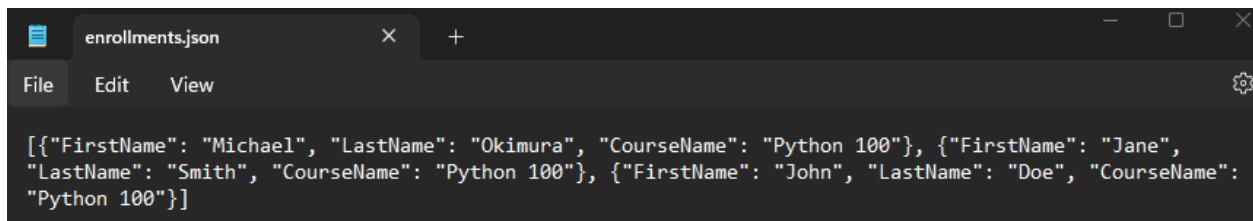
Example 1: Student class containing the key new coding items we learned for this assignment.

Processing and Testing

Outside of writing the code for the classes, the remainder of the assignment for error handling, menu display, functionality of input registration, and reading/writing the json file was similar to last week's assignment. Running the code, it follows the testing and error handling requirements. To confirm the accuracy of the code, I ran it on PowerShell and the VSC terminal getting the same results on the file.

The screenshot displays two side-by-side terminal windows. The left window is a Windows PowerShell terminal, and the right window is a Visual Studio Code (VSC) terminal. Both windows show the output of a Python program titled 'Course Registration Program'. The program prompts the user to 'Enter your menu choice number:'. In the PowerShell terminal, the user enters '2', and the program displays the current data: 'Student Michael Okimura is enrolled in Python 100', 'Student Jane Smith is enrolled in Python 100', and 'Student John Doe is enrolled in Python 100'. The user then enters '3', and the program displays the same data. In the VSC terminal, the user enters '2', and the program displays the same data. The user then enters '3', and the program displays the same data. The program's menu options are: 1. Register a Student for a Course, 2. Show current data, 3. Save data to a file, and 4. Exit the program.

Example 2: PowerShell and VSC terminals execution of menu options 2 and 3.



```
[{"FirstName": "Michael", "LastName": "Okimura", "CourseName": "Python 100"}, {"FirstName": "Jane", "LastName": "Smith", "CourseName": "Python 100"}, {"FirstName": "John", "LastName": "Doe", "CourseName": "Python 100"}]
```

Example 3: Json file output results displayed in Notepad from example 2's entry above.

Summary

Classes remain a challenge for me as I tried recreating examples on my own from scratch but still need to refer to my notes to complete the code correctly. Practice makes perfect as they say. Finding real-world examples to follow and attempt coding will be more ideal to for learning. As I mentioned above in the introduction, I can see how important classes are to writing code and imagine it's used widely and frequently so taking the time to really understand it's uses and practicing how to write it will be vital in becoming a successful programmer.