Erin Fox
11/26/2024
IT FDN 110
Assignment07
https://github.com/emfox55/IntroToProg-Python-Mod07

Classes and Objects

Introduction

In this assignment we were asked to expand on our prior work on functions and separation of concerns by layering in classes, objects and constructors. In this document I will detail the steps I took to complete this task. All assignment instructions and informational materials that informed my approach can be found in the UW IT FDN 110 “Module07 - Classes and Objects” course materials.

Notes on Previous Documentation

The main functionality of this code was established in Assignments 06, so I focus primarily on the new methods here. Please refer to the previous assignment documents for more detail. All development and testing was completed using the PyCharm IDE.

Objects, Properties & Constructors

In this assignment we created new data classes called Person and Student. We then store the student data in an object instance, Student, of the Student class. This allows us to preserve all the data validation and error handling in the class constructor properties rather than individual IO/processing class methods as in Assignment06. This promotes isolation, reusability and abstraction in our code (***Mod07 Notes - Objects vs. Classes, Constructors, Properties***).

Figure 1: Properties.

```
class Person:  1 usage
    """
    A class representing person data.

    Properties:
        first_name (str): The student's first name.
        last_name (str): The student's last name.

    ChangeLog:
        EFox,11/26/2024,Created the class.
    """

    def __init__(self, first_name: str = '', last_name: str = ''):
        self.first_name = first_name
        self.last_name = last_name

    @property  8 usages (6 dynamic)
    def first_name(self):
        return self.__first_name.title()  # formatting code
```

Another handy aspect of using the new Person and Students classes is that Students can inherit the properties and constructors already defined in the Person class(***Mod07 Notes - Inherited Code***). This gives us the advantage of yet more reusability and more concise code.

Figure 2: Inherited properties using super().__init__

```
class Student(Person):
    """
    A class representing student data.

    Properties:
        first_name (str): The student's first name.
        last_name (str): The student's last name.
        course_name (str): The registered course name of the student.

    ChangeLog: (Who, When, What)
    EFox,11/26/2024,Created Class
    """

    def __init__(self, first_name: str = '', last_name: str = '', course_name: str = ''):
        super().__init__(first_name=first_name, last_name=last_name)
        self.course_name = course_name
```

One “downside” of leveraging objects is that it complicates the code a bit when we read from and write to the json file, since we are working with a list of objects now, instead of a list of dictionary rows as in the json.

Figure 3: Converting a list of dictionary rows to a list of objects.

```
try:
    file = open(file_name, "r")

    list_of_dictionary_data = json.load(file)  # the load function returns a list of
    for student in list_of_dictionary_data:  # Convert the list of dictionary rows i
        student_object: Student = Student(first_name=student["FirstName"],
                                          last_name= student["LastName"],
                                          course_name=student["CourseName"])
        student_data.append(student_object)

    file.close()
```

Testing

The program ran fine initially in PyCharm, but did throw a non-fatal error when attempting in Terminal:

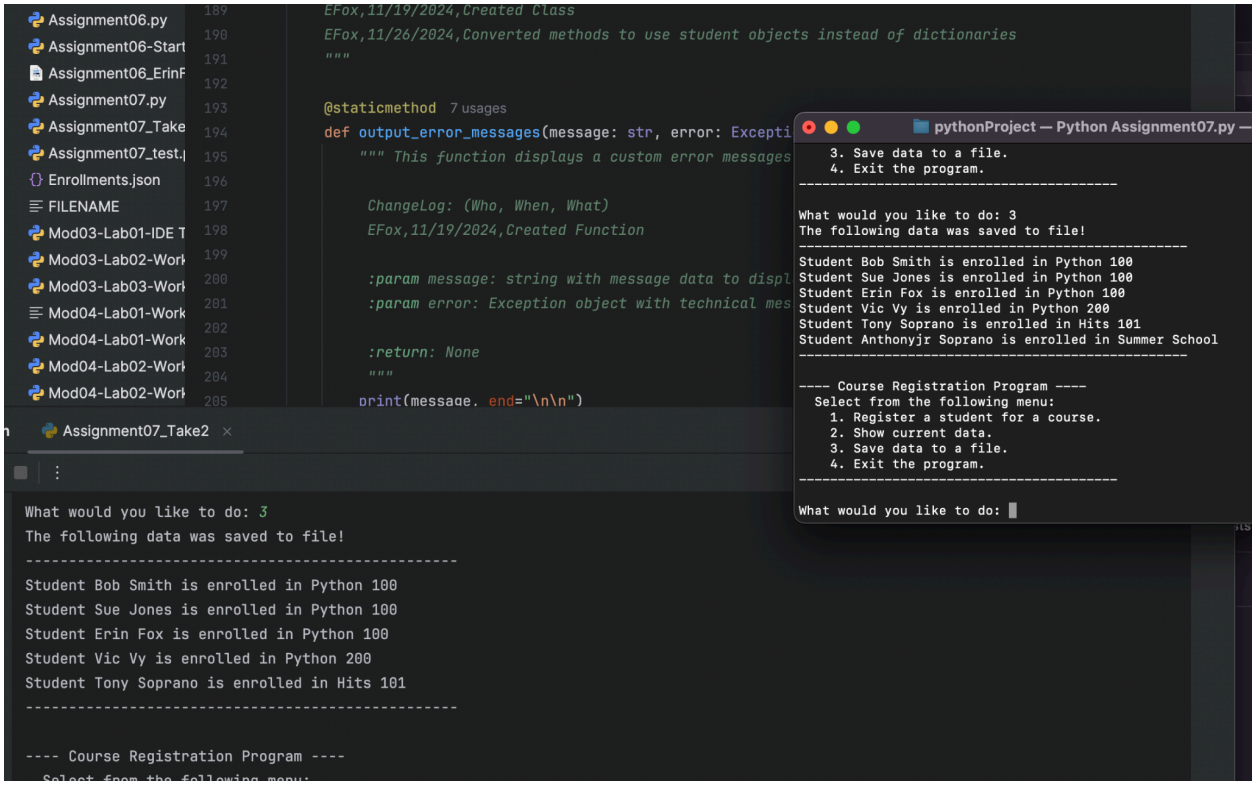
Figure 4: Terminal run-time error.

```
[erinfox@Brians-MacBook-Pro pythonProject % python3 Assignment07.py
Error: There was a problem with reading the file.

-- Technical Error Message --
'str' object has no attribute 'has_numeric'
Attribute not found.
<class 'AttributeError'>


---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------


Enter your menu choice number:
```

Originally I included the has_numeric method I created in Assignment06, but I struggled to get it working properly within the scope of the new classes and properties. It was created in the IO task, but is first referenced in the student_first_name and student_last_name properties in the Person class, which precede it in the code. This resulted in attribute errors. Given time constraints I opted to use the built-in isalpha method since it is pre-defined and therefore does not run into scope issues with the properties.

I was able to successfully run this code in both PyCharm and Terminal

Figure 5: Successfully testing code



Summary

Using the documents and videos provided in the Module07 course materials, I was able to understand and display how to use functions and classes to achieve modularity, reusability and separation of concerns. I was able to validate the accuracy of my script by running it in both IDLE and Mac Terminal.