Name: Ronnie Sokha

Date: 12/1/2023

Course: Foundations of Python Programming

GitHub Link: https://github.com/RSokha/PythonMod7

Assignment 07: Classes and Objects

Introduction

This assignment does not stray much from the previous. In module 7, we learn the notions behind Object-Oriented Programming (OOP) and Classes. Python is an OOP; that is, we can represent a collection of data in Python as an object. Classes are essentially 'blueprints' we can use as a template when creating our objects. An object is created using the constructor of the class.

Creating Classes

For the sake of brevity, we copy the classes or 'blueprint' from the previous assignment, but we add in the Student & Person class.

Figure 1 – we keep the same data constants and data variables and view our menu for the user.

```
# Define the Data Constants
MENU: str = '''
---- 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.
-----------------------------------------
'''
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables
students: list = []  # a table of student data
menu_choice: str  # Hold the choice made by the user.
```

Creating Person class

```
class Person:
```

Figure 2 – we add first and last name properties to the constructor

```
def __init__(self, first_name: str = "", last_name: str = ""):
    self.__last_name = None
    self.first_name = first_name
    self.last_name = last_name
```

Figure 3 – we create a getter and setter for first name and last name

```
@property
def first_name(self):
    return self.__first_name.title()

@first_name.setter
def first_name(self, value: str):
    if value.isalpha() or value == "":
        self.__first_name = value
    else:
        raise ValueError("First name cannot contain numbers.")


@property
def last_name(self):
    return self.__last_name.title()

@last_name.setter
def last_name(self, value: str):
    if value.isalpha() or value == "":
        self.__last_name = value
    else:
        raise ValueError("Last name cannot contain numbers.")
```

Creating Student class – similar to Person class that inherits from Person class

```
class Student(Person):
```

Figure 1 – call to the Person constructor and pass in first and last name and also course name

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

# Here, we add an assignment to the course_name property using course_name
parameter
       self.course_name = course_name
```

Figure 2 – add the getter and setter for course_name

```
@property
def course_name(self):
    return self.__course_name.title()

@course_name.setter
def course_name(self,value: str):
    self.__course_name=value
```

Conclusion

The rest of the script remains the same from module 6; that is, we still have our FileProcessor class to read and write our JSON file. We have our IO class for our input and output methods. Most importantly, we ensure we have our try catches in between to ensure invalid input is captured. As mentioned before, classes are simply a notion that acts as a template or blueprint for creating objects in Python; e.g., a Person object, a Student object, and so forth. Classes are the fundamental building blocks of Python because it helps us define an object within the program's code, representing a group of data and functions.