Miyles Clark
8/14/2024
Foundations of Programming: Python
Assignment 07
https://github.com/macxuw/IntroToProg-Python-Mod07/blob/main/Assignment07.py

Data Classes

Introduction

Assignment 07 continues building upon a student registration program by adding data classes. The first class is a Person class, holding only names, while a Student subclass takes a first and last name in addition to a course name. The while loop remains primarily unchanged: first reading and appending a JSON file, then presenting the following menu:

Enter new data.
Display student data.
Write student data to a JSON file.
Exit the program.

The program will accept multiple course registrations, present all data in the students list, and write data back to a JSON file. Error handling was not altered.

Adding Code

The task required for Assignment 07 was to add necessary data classes and modify the read_data_from_file and write_data_to_file functions to convert the JSON data into the Student class and vice versa.

Data Classes

The Person class holds a person's first and last name, along with any input validation code. The ___init__ constructor initializes the function's variables as private attributes, using the self keyword to identify object instances (Figure 1).

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

Figure 1: Initialize variables in a class

Once initialized, getters and setters can apply formatting and error validation, with the latter setting the private variable to a value (Figure 2).

```
@first_name.setter
  def first_name(self, value: str):
    if value.isalpha() or len(value) < 2:
      self.__first_name = value
    else:
      raise ValueError("First name should only contain letters and be at least 2 characters.")
```

Figure 2: Setter definition for first_name

The exact process is followed for every variable in the class: initialize, set, get. For the child class Student(Person), the __init__ function uses the Person object variables, first and last name, and adds the course name (Figure 3).

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

Figure 3: Calling parent variables and adding a child variable

Other data classes can call upon the Person class, for example for teachers or administration staff.

Lists and Collections

Class object instances do not exist in external files. As such, Python code must be used to assign dictionary keys to object instances. A local variable acts as a go-between, first assigning the row values to dictionary keys (Figure 4) and then, when saving, associating the key to a specific class variable path (Figure 5).

```
for row in file_dict:
student_data.append(Student(row["FirstName"],
        row["LastName"],
        row["CourseName"])
  )
```

Figure 4: Identify conversion to load file

```
for student in student_data:
        file_dict.append({"FirstName": student.first_name,
                    "LastName": student.last_name,
                    "CourseName": student.course_name}
                                )
```

Figure 5: Associate class variable names to dictionary key

The variable file_dict is assigned when the file is loaded (Figure 6):

```
file = open(file_name, "r")
file_dict = json.load(file)
file.close()
```

Figure 6: Code to load existing data into the student_data dictionary

Once loaded into memory, a user can add one or more students and display current data. The verification displayed after every registration calls the class name and the object variable (Figure 7).

```
print(f"{student.first_name} {student.last_name} is enrolled in
        {student.course_name}")
```

Figure 7: Registration verification code

When writing the data to the JSON file, the student_data dictionary is iterated back to file_dict (Figure 8):

```
file = open(file_name, "w")
json.dump(file_dict, file, indent=1)
```

Figure 8: Save the students list to JSON file

An indent level set to 1 positions each key and associated value on a separate line (Figure 9).

```
{
 "FirstName": "Stu",
 "LastName": "Hust",
 "CourseName": "English Lit"
}
```

Figure 9: Formatted JSON data

Disability Accessibility

Ease of use is important to consider when building a program. Using .strip() at the end of user inputs ensures extra whitespace is stripped, increasing data uniformity (Figure 10).

```
course_name = input("Please enter the \033[1mcourse name\033[0m:
        ").strip()
```

Figure 10: Using .strip() for data uniformity

In addition to bolded phrases on the student inputs, the ANSI code encloses the individual numbers in the menu_choice validation statement to further highlight valid data inputs (Figure 11).

```
menu_choice = input("Enter your menu choice number: ").strip()
      if menu_choice not in ("1", "2", "3", "4"):
        raise Exception("Please choose \033[1m1\033[0m,
                     \033[1m2\033[0m, \033[1m3\033[0m, or
                     \033[1m4\033[0m")
```

Figure 11: Highlighting requested output

Fixing an Error

The program defines two data variables in the main body of the script, one for menu_choice and another for the list of students. When tested, the results yielded a custom object rather than the student list. The writing to file and display functions were affected differently; however, only the write function failed to work. The students list was declared incorrectly (Figure 12), and the list should have been set to an empty string rather than setting the Student class as an object (Figure 13).

```
students: list = [Student]
```

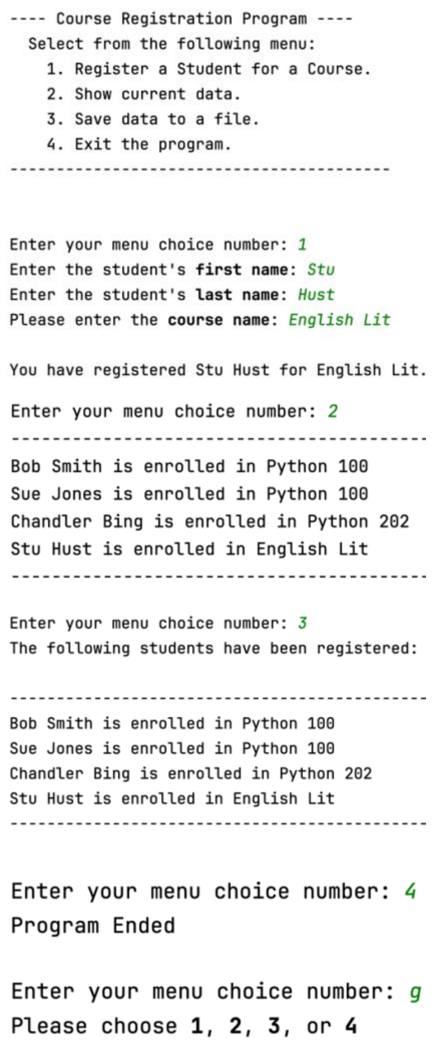
Figure 12: Incorrectly declared Student object in list

```
students: list[Student] = []
```

Figure 13: Declare an empty string that will hold Student class data

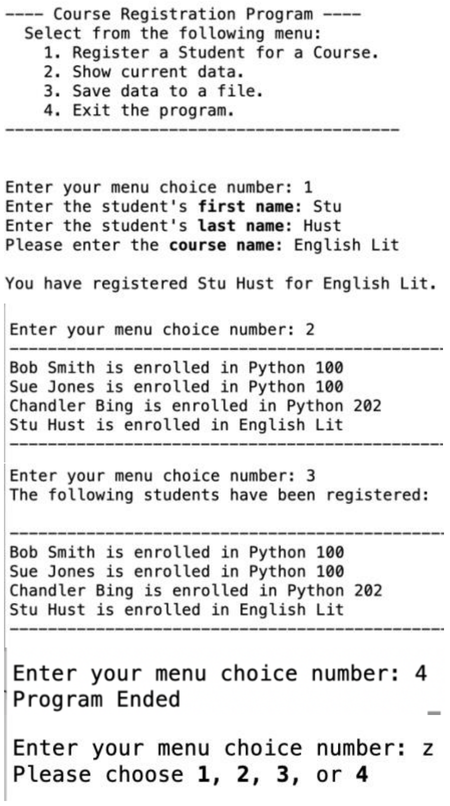
Results

Testing my program in PyCharm (Figure 14) and Terminal (Figure 15) returned the correct outcome, whether a prompt for data, a print statement, or an end to the program.



```
---- 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: 1
Enter the student's first name: Stu
Enter the student's last name: Hust
Please enter the course name: English Lit

You have registered Stu Hust for English Lit.

Enter your menu choice number: 2
-----------------------------------------
Bob Smith is enrolled in Python 100
Sue Jones is enrolled in Python 100
Chandler Bing is enrolled in Python 202
Stu Hust is enrolled in English Lit
-----------------------------------------

Enter your menu choice number: 3
The following students have been registered:

---------------------------------------------
Bob Smith is enrolled in Python 100
Sue Jones is enrolled in Python 100
Chandler Bing is enrolled in Python 202
Stu Hust is enrolled in English Lit
---------------------------------------------


Enter your menu choice number: 4
Program Ended

Enter your menu choice number: g
Please choose 1, 2, 3, or 4
```

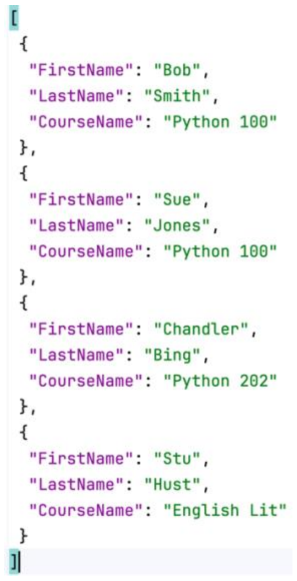
Figure 14: Program output in PyCharm



```
---- 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: 1
Enter the student's first name: Stu
Enter the student's last name: Hust
Please enter the course name: English Lit

You have registered Stu Hust for English Lit.

Enter your menu choice number: 2
---------------------------------------------
Bob Smith is enrolled in Python 100
Sue Jones is enrolled in Python 100
Chandler Bing is enrolled in Python 202
Stu Hust is enrolled in English Lit
---------------------------------------------

Enter your menu choice number: 3
The following students have been registered:

---------------------------------------------
Bob Smith is enrolled in Python 100
Sue Jones is enrolled in Python 100
Chandler Bing is enrolled in Python 202
Stu Hust is enrolled in English Lit
---------------------------------------------

Enter your menu choice number: 4
Program Ended

Enter your menu choice number: z
Please choose 1, 2, 3, or 4
```

Figure 15: Program output in Terminal

The JSON file for this assignment returns a student's first and last name and a course name. The data is listed next to the corresponding dictionary key (Figure 16).



```
[
 {
  "FirstName": "Bob",
  "LastName": "Smith",
  "CourseName": "Python 100"
 },
 {
  "FirstName": "Sue",
  "LastName": "Jones",
  "CourseName": "Python 100"
 },
 {
  "FirstName": "Chandler",
  "LastName": "Bing",
  "CourseName": "Python 202"
 },
 {
  "FirstName": "Stu",
  "LastName": "Hust",
  "CourseName": "English Lit"
 }
]
```

Figure 16: JSON file data

Summary

Creating data classes further encapsulates a program to prevent errors while simplifying the front-end code. In earlier assignments, up to a dozen global variables were declared before starting the body of the code; with the addition of data classes, the script declared two data constants before and two variables after delineating layers of concern and their associated classes and functions. Data classes allow programmers to create a hierarchy of classes with inherited attributes. In this assignment, the Student class inherited the first and last name attributes from the Person class and added the student's course name.