Angel Reddy
Luis Conejo Alpizar and Julian Peterson
Foundations in Programming (Python)
26 February 2024

Assignment 07

Introduction

This week's assignment built upon assignment 06 where we created a python program that demonstrates the use of constants, variables, and print statements to display a message about registering a student for a course. This week added even further to the use of classes and separations of concern.

Drafting the Script

To begin, I defined the additional classes that I needed for this assignment. The two classes that I used were Person and Student. The Person class included variables for the student's first name and last name. I used the constructor __init__ for student_first_name and student_last_name to be able to assign these variables to the class Person. I also added script for error handling in case the user inputs values incorrectly. Additionally, I added the class Student that inherits from the class Person. I used the same constructor __init__ to allow for inheritance to occur. I also added course_name to this class as the student's information will also include the course.

```python
class Student(Person):
    """
    A class representing student data. Inherits from Person class. Adds input for course.

    Properties:
    - student_first_name (str): The student's first name.
    - student_last_name (str): The student's last name.
    - course_name (str): The student's course.

    """

    def __init__(self, student_first_name: str = "", student_last_name: str = "", course_name: str = ""):

        super().__init__(student_first_name=student_first_name,student_last_name=student_last_name)
        self.course_name = course_name

    @property
    def course_name(self):
        return self.__course_name.strip()

    @course_name.setter
    def course_name(self, value: str):
        if value.replace(' ', '').isalnum() or value == "":
            self.__course_name = value
        else:
            raise ValueError("The course name should only contain letters and numbers.")

    def __str__(self):
        return f"{self.student_first_name},{self.student_last_name},{self.course_name}"
```

Figure 1. Example of class. I added the class Student that inherited information from the class Person. The above image features an example of the script I used to be able to add a new class that replaced some of my script.

The addition of classes was the largest change that I made to my script. These classes required that I alter some other parts of the script. For example, I had to add a variable called

each_student in the definition read_data_from_file. I noticed that this was needed in order to be able to identify what was written in the existing JSON file. The variable each_student was used throughout the script wherever I would use student_first_name, student_last_name, and student_course_name. This allowed the program to read the list and display the contents to the user.

```
@classmethod
def write_data_to_file(cls, file_name: str, student_data: list):
    """
     Writes data to json file from list.

    """
    file = None
    try:
        list_of_dict_data: list = []

        # Add Student objects to Json compatible list of dictionaries.
        for each_student in student_data:
            student_json: dict \
                = {"FirstName": each_student.student_first_name,
                   "LastName": each_student.student_last_name,
                   "CourseName": each_student.course_name}
            list_of_dict_data.append(student_json)

        file = open(file_name, "w")
        json.dump(list_of_dict_data, file)
        file.close()

        # Present the current data
        print()
        print("-" * 50)
        print("The file contains: ")
        for each_student in list_of_dict_data:
            print(
                f'{each_student["FirstName"]}, '
                f'{each_student["LastName"]}, '
                f'{each_student["CourseName"]}'
            )
        print("-" * 50,)
```

Figure 2. Adding objects to the dictionary. To be able to use the two new classes that I introduced, I needed to convert the JSON dictionary objects to student objects. This allowed me to reuse each_student and add it to my variables to be able to read the JSON file.

Running the Script and Saving to GitHub

I was able to run this script with no issue! I used the starter script provided in the module as this was easier for me to understand what I needed to add. Also, I looked at examples online of creating objects as I kept getting an error stating that the program was unable to read the JSON file. However, after numerous rewrites, I was able to get it to run successfully in PyCharm, IDLE, and the Terminal. I also was able to save it to GitHub here: https://github.com/chreddy1020/IntroToProg-Python-Mod07.

Reflection and Summary

This assignment was also difficult for me. I needed to copy over a great deal of script from other sources and adjust it to fit my script. That being said, I understood the flow of the script and what was happening for each choice on the menu. I was able to make the proper adjustments to be able to get my program running. However, this time I struggled to understand some lines of the script. For example, I am still unsure as to why I needed to include the object each_student on every instance of student_first_name and student_last_name. Is there a way to

make this more efficient? I am still exploring and learning which is very good! At times though, I still struggle to understand the why behind what I am writing.