Matt Fevergeon
8/17/2025
IT FDN 110 A
Assignment 0
My Git Link

Assignment07 Summary

Program and Week 7 Overview

Similar to last week, I've been staying in VScode instead of Pycharm. Some major learnings this week involved adding a new extension to help format the color theme of VScode similar to Pycharm to help make watching the videos and implementing the demos easier to swap between. I was also able to get VScode connected with Github so I can directly upload similar to what was demonstrated and asked for pycharm.

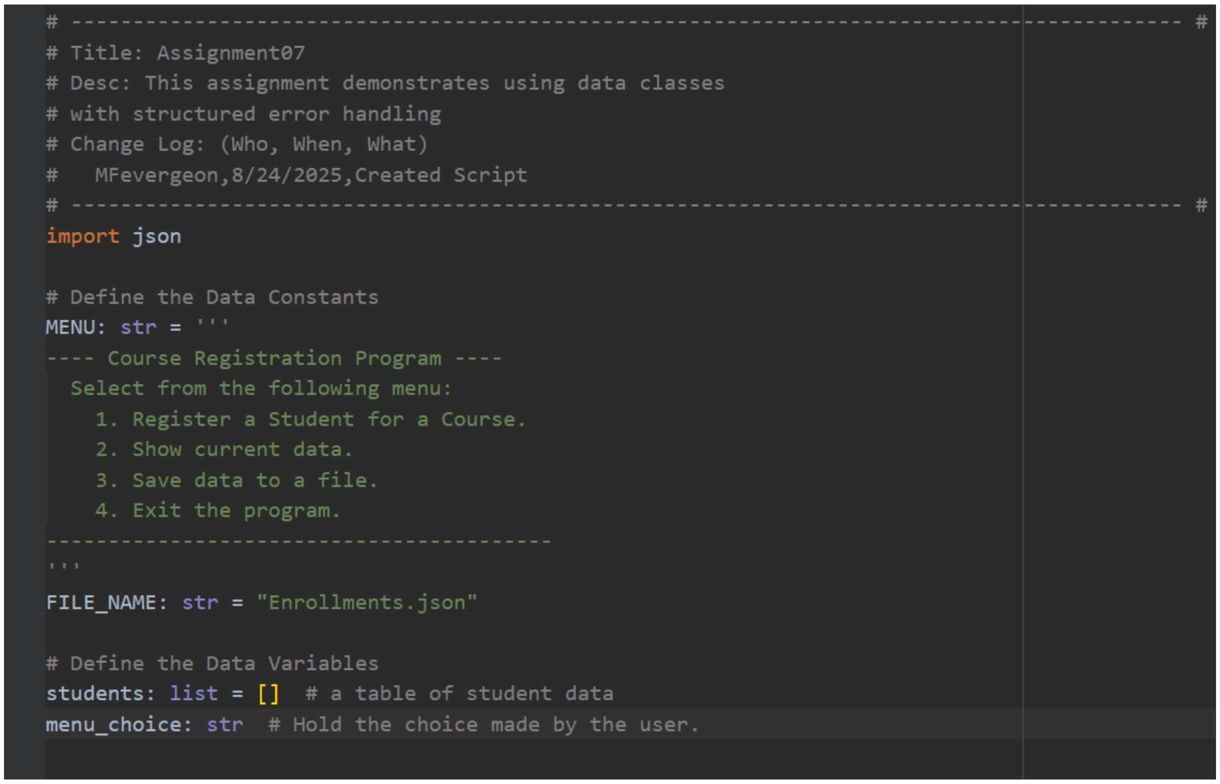
I have a similar sentiment to last week, in that there's a lot of content to absorb each week. I heavily relied on Copilot AI to help me understand what labs were doing, for debugging and finding errors in my assignment code, and debugging. I've been using it frequently to clarify concepts.

One other difficulty this week (and last) was that some of the demos had some differences to how the homework was formatted and written, and how the lab starter files were formatted.

Script Overview and Acceptance Criteria

The script, named Assignment07.py, is structured to fulfill all requirements for a course registration program. The script header includes the required text and has been updated with the author's name and the current date, ensuring clear authorship and version tracking.

The program defines two constants: MENU, which contains the exact menu text as specified, and FILE_NAME, which is set to "Enrollments.json". These constants are used throughout the program and remain unchanged, providing a stable reference for menu display and file operations. The script initializes the variables menu_choice as an empty string and students as an empty list, establishing a clean starting state for user interaction and data storage (see Figure 1 for the header, constants, and variable definition).



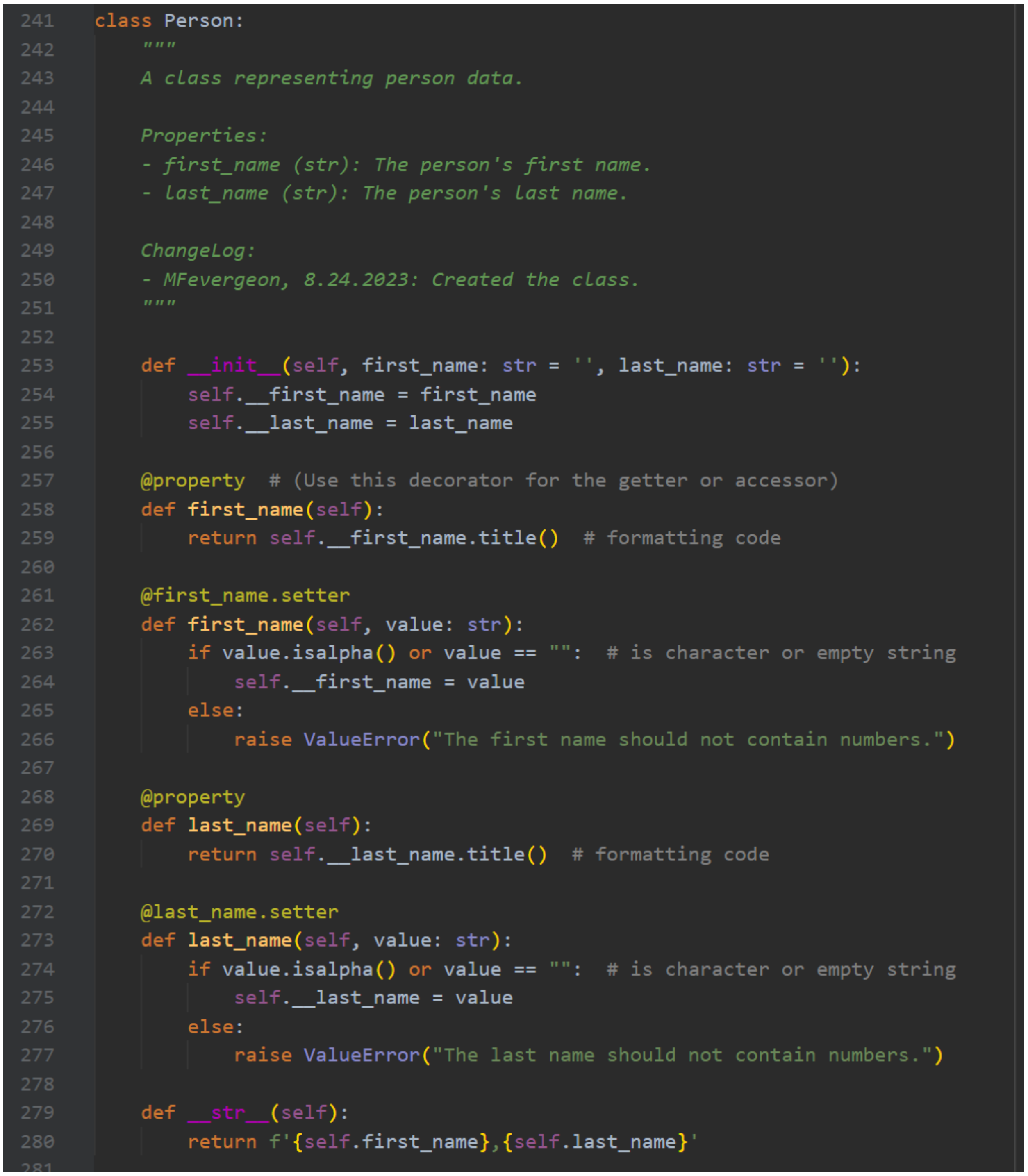
```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
# with structured error handling
# Change Log: (Who, When, What)
#   MFevergeon,8/24/2025,Created Script
# ------------------------------------------------------------------------------------------ #
import json

# 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.
```

Figure 1: Header, Constants and Variables.

A note about the row character max length: while 79 character limits have been suggested, with the improvement overtime of wider resolutions, I have not stuck to this limit. The starter file also comments dashes out to Column 9, so I updated my visual ruler in VSCode to stay within that range. The code is very readable with any modern monitor or screen.

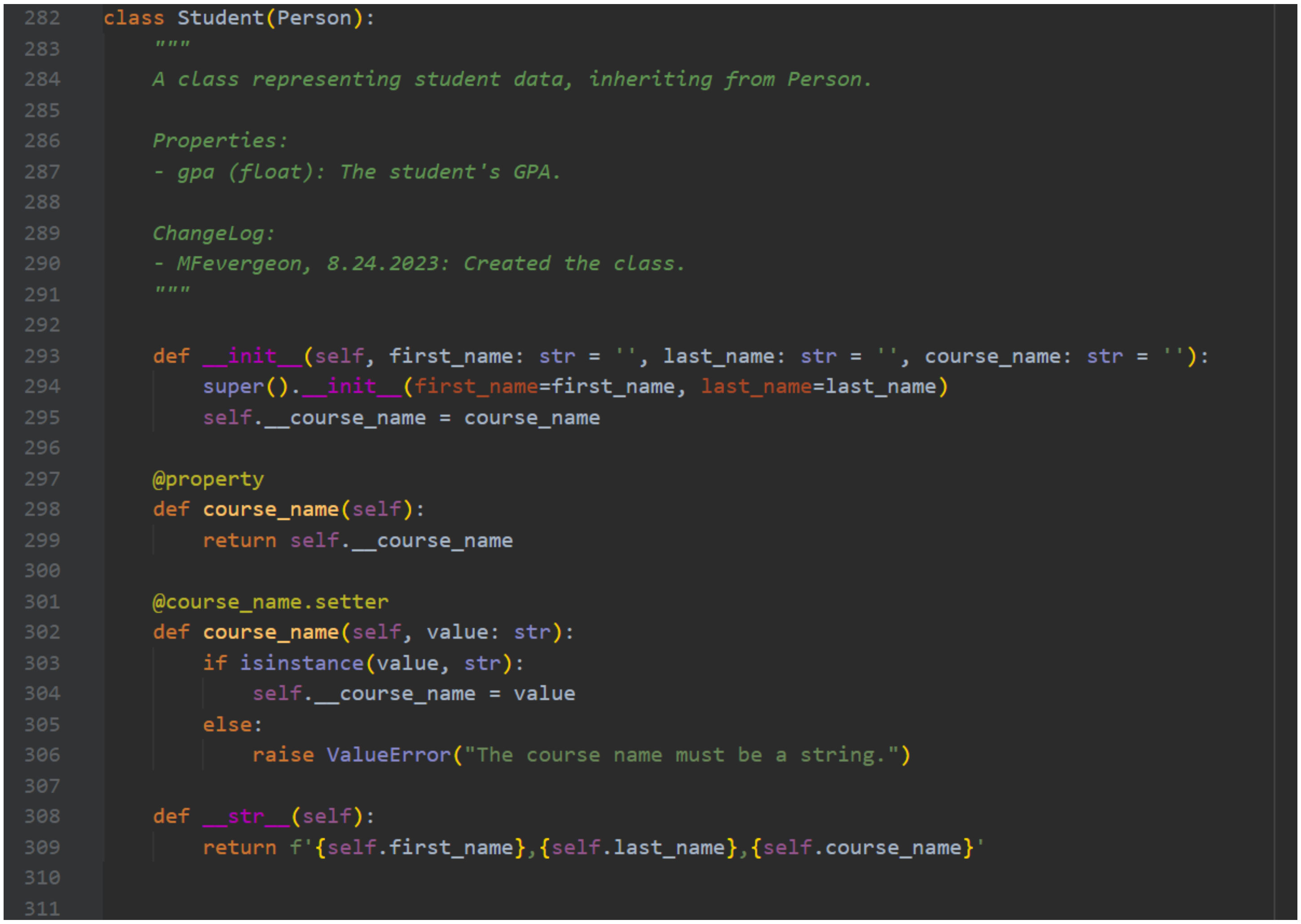
Class Definitions

Four classes are defined: FileProcessor, IO, Person, and Student. FileProcessor and IO class were defined as part of the starter.py file so will not be shown in the screen shot figures. Each class includes a descriptive docstring, and the Student and Person classes provide properties for first name, last name, and course name, all defaulting to empty strings. These properties include simple validation logic to ensure that names do not contain numbers and that course names are strings. Both data classes implement a str method to extract and display comma-separated data, supporting clear and consistent output (see Figure 2 and 3 for class and property definitions for Person and Student Classes).



```python
class Person:
    """
    A class representing person data.

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

    ChangeLog:
    - MFevergeon, 8.24.2023: Created the class.
    """

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

    @property  # (Use this decorator for the getter or accessor)
    def first_name(self):
        return self.__first_name.title()  # formatting code

    @first_name.setter
    def first_name(self, value: str):
        if value.isalpha() or value == "":  # is character or empty string
            self.__first_name = value
        else:
            raise ValueError("The first name should not contain numbers.")

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

    @last_name.setter
    def last_name(self, value: str):
        if value.isalpha() or value == "":  # is character or empty string
            self.__last_name = value
        else:
            raise ValueError("The last name should not contain numbers.")

    def __str__(self):
        return f'{self.first_name},{self.last_name}'
```

Figure 2 – Person Class



```python
class Student(Person):
    """
    A class representing student data, inheriting from Person.

    Properties:
    - gpa (float): The student's GPA.

    ChangeLog:
    - MFevergeon, 8.24.2023: Created the 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

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

    @course_name.setter
    def course_name(self, value: str):
        if isinstance(value, str):
            self.__course_name = value
        else:
            raise ValueError("The course name must be a string.")

    def __str__(self):
        return f'{self.first_name},{self.last_name},{self.course_name}'
```

Figure 3 – Student Class

Functions

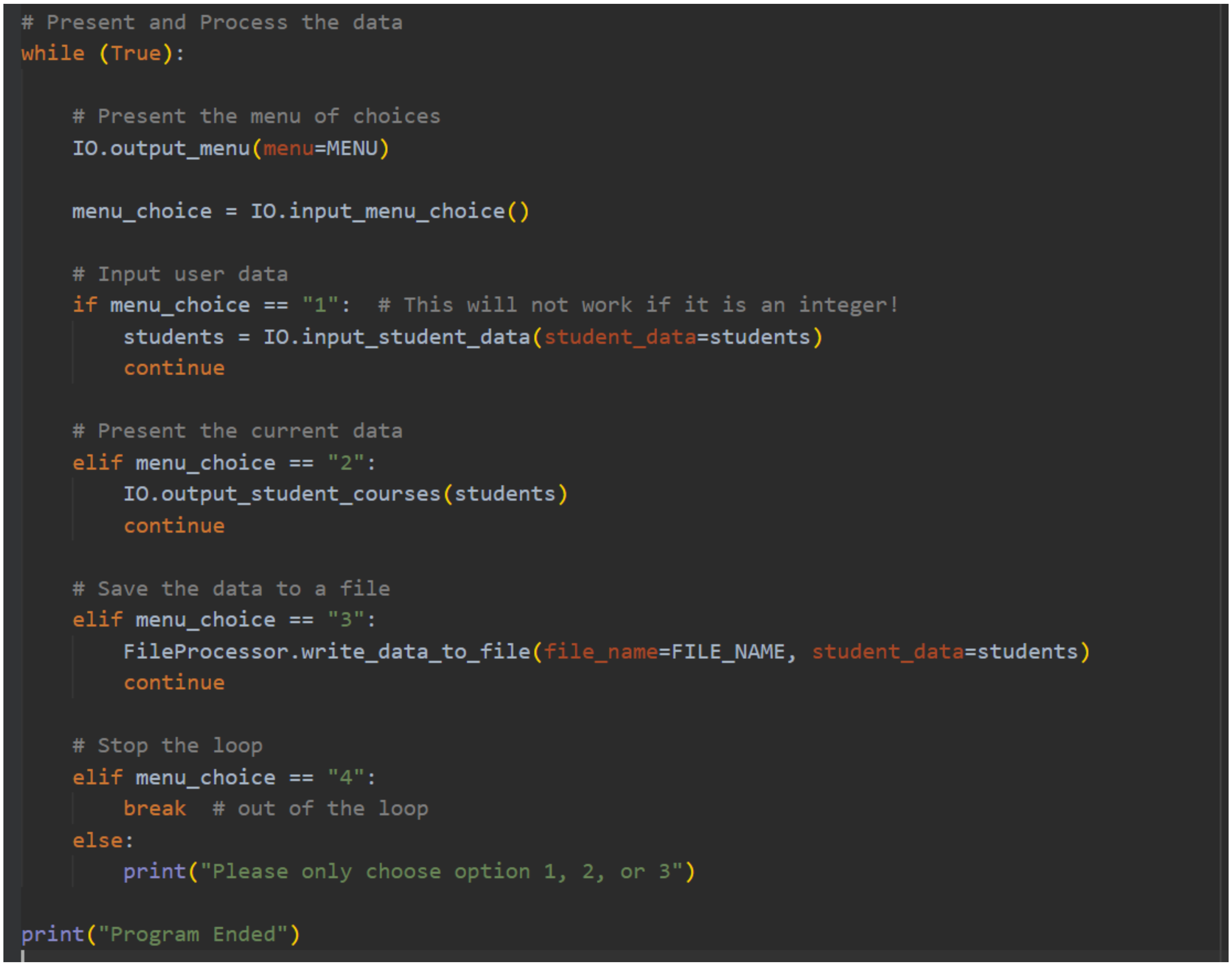
All functions in the script are documented with descriptive docstrings. Functions that include exception handling use the output_error_messages() function to provide structured and user-friendly error messages. Non-instance methods in the FileProcessor and IO classes are correctly decorated with @staticmethod, adhering to best practices for utility functions. The program includes all required functions with the specified names and parameters, such as output_error_messages, output_menu, input_menu_choice, output_student_courses, input_student_data, read_data_from_file, and write_data_to_file (see Figure 3 for example of function definitions and decorators).

Input/Output

For input and output, when the user selects menu choice 1, the program prompts for the student's first name, last name, and course name using the input() function via the IO.Input_student_data method, storing these values in the appropriate variables. The collected data is added to the students list as Student objects, ensuring a two-dimensional list of Student rows. On menu choice 2, the program prints each Student object, displaying the data as a string of comma-separated values, as required (see Appendix for all test results). This uses the student objects again rather than a dictionary.

Input/Output

In terms of processing, the program automatically reads the contents of "Enrollments.json" at startup using json.load(), loading the data into a list of dictionaries and then converting each dictionary into a Student object. When the user selects menu choice 3, the program opens "Enrollments.json" in write mode, converts the list of Student objects into a list of dictionaries, and writes this list to the file using json.dump(), ensuring the file is a valid JSON array taking advantage of the "indent=4" argument. The file is then properly closed. Selecting menu choice 4 ends the program as expected (see Figure 7 for file reading and writing code).



```
# Present and Process the data
while (True):

    # Present the menu of choices
    IO.output_menu(menu=MENU)

    menu_choice = IO.input_menu_choice()

    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        students = IO.input_student_data(student_data=students)
        continue

    # Present the current data
    elif menu_choice == "2":
        IO.output_student_courses(students)
        continue

    # Save the data to a file
    elif menu_choice == "3":
        FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
        continue

    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop
    else:
        print("Please only choose option 1, 2, or 3")

print("Program Ended")
```

Figure 4 – Main While Loop

Structured Error Handling

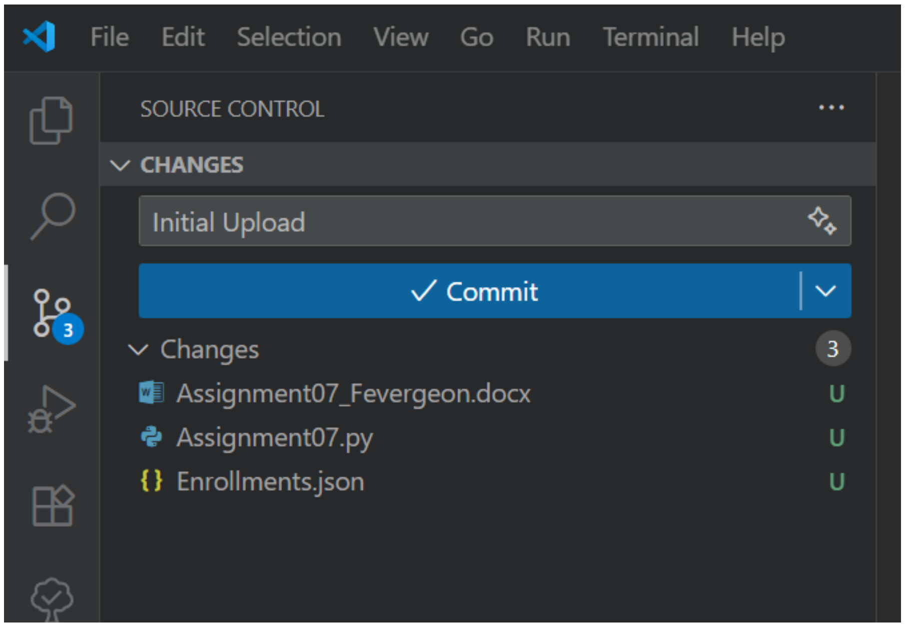
Structured error handling is present throughout the script. Errors encountered when reading the file, entering first or last names, or writing to the file are all caught and reported using the output_error_messages() function, providing both user-friendly and technical feedback. Examples of the newly added error handling for this week can be found in figure 2 and 3 which show the new Person and Student Classes.

Testing

The program has been tested to ensure it accepts user input for student registration, displays the entered data, and saves it to a JSON file. It supports multiple registrations and displays, and allows saving multiple records to the file. The script runs correctly in both PyCharm and from the console or terminal, confirming its portability and reliability (See Appendix for Test results).

Source Control

The script file and a copy of this summary document are loaded onto my GitHub repository, which is linked at the top of the page as well as posted in the discussion forum on canvas. Rather than use Pycharm, as I mentioned earlier, I linked VSCode to Git to manage the uploads. I have a workspace that provides me access to all of the class modules. In order to individually manage just assignment07, I created a new local folder location, and cloned the Module 07 repository there and made local copies of my Script, json file and summary document. In practice, I would likely change the repository structure to have a single repository for the Python Course, and sub folders or projects for each module. Figure 5 shows what the commit to Git page looks like in VSCode.



Initial Upload

Assignment07_Fevergeon.docx U
Assignment07.py U
Enrollments.json U

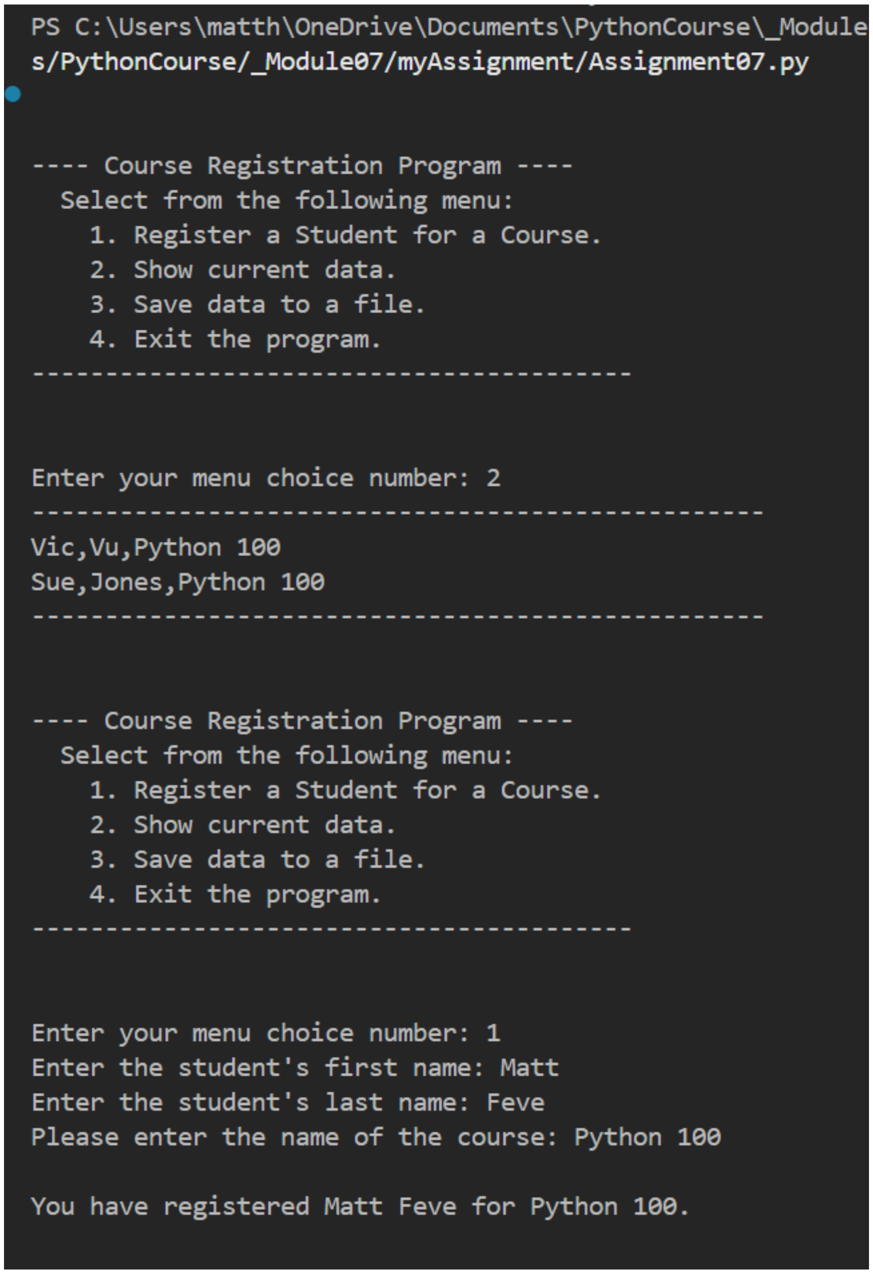
Figure 5 – VSCode Commit

Conclusion

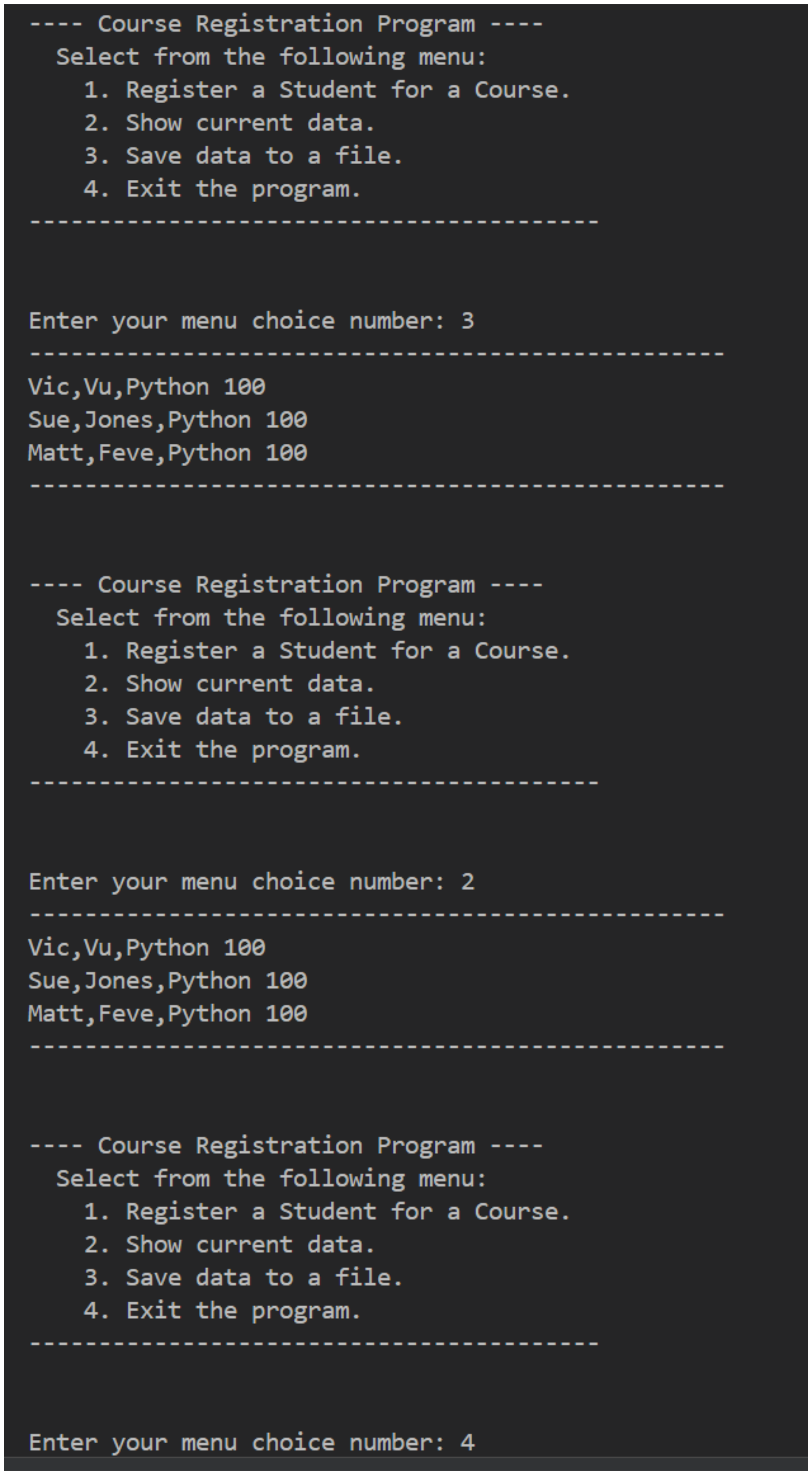
The script meets all functional, structural, and error-handling requirements, and is formatted and documented according to the assignment's specifications.

The original Assignment07-Starter.py script was structured to use basic data handling with dictionaries and included placeholder TODOs for creating and using classes. This week's Assignment file has been significantly enhanced to use object-oriented programming principles. The final script introduces and implements the Person and Student classes, with Student inheriting from Person. These classes encapsulate student data and provide property validation and custom string formatting. The input and output logic has been updated so that, instead of working with dictionaries, the program now creates and manipulates Student objects, storing them in the students list. When reading from or writing to the JSON file, the script converts between Student objects and dictionaries, ensuring compatibility with JSON serialization while maintaining the benefits of class-based data management in memory. Additionally, the output functions now leverage the str methods of these classes for clean, comma-separated display of student data. Overall, the transition from the starter script to the completed version demonstrates a shift from procedural to object-oriented design, resulting in more robust, maintainable, and extensible code.

Appendix A – VSCode Terminal Results



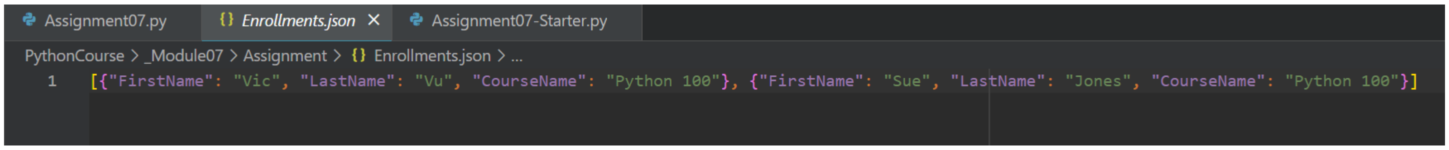
```
PS C:\Users\matth\OneDrive\Documents\PythonCourse\_Module
s/PythonCourse/_Module07/myAssignment/Assignment07.py


---- 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: 2
--------------------------------------------------
Vic,Vu,Python 100
Sue,Jones,Python 100
--------------------------------------------------



---- 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: Matt
Enter the student's last name: Feve
Please enter the name of the course: Python 100

You have registered Matt Feve for Python 100.
```



```
---- 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: 3
--------------------------------------------------
Vic,Vu,Python 100
Sue,Jones,Python 100
Matt,Feve,Python 100
--------------------------------------------------


---- 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: 2
--------------------------------------------------
Vic,Vu,Python 100
Sue,Jones,Python 100
Matt,Feve,Python 100
--------------------------------------------------


---- 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: 4
```

Appendix B – Enrollements.json before and after

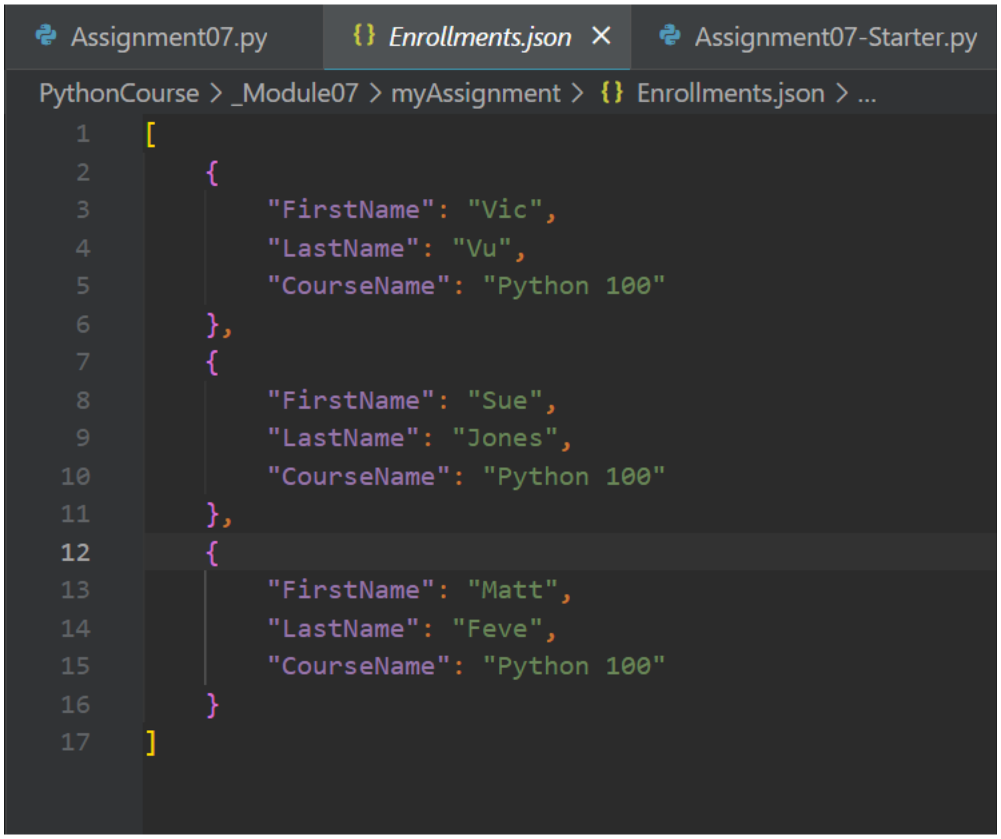
Before:



Assignment07.py
Enrollments.json
Assignment07-Starter.py
PythonCourse > _Module07 > Assignment > Enrollments.json > ...

```json
[{"FirstName": "Vic", "LastName": "Vu", "CourseName": "Python 100"}, {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"}]
```

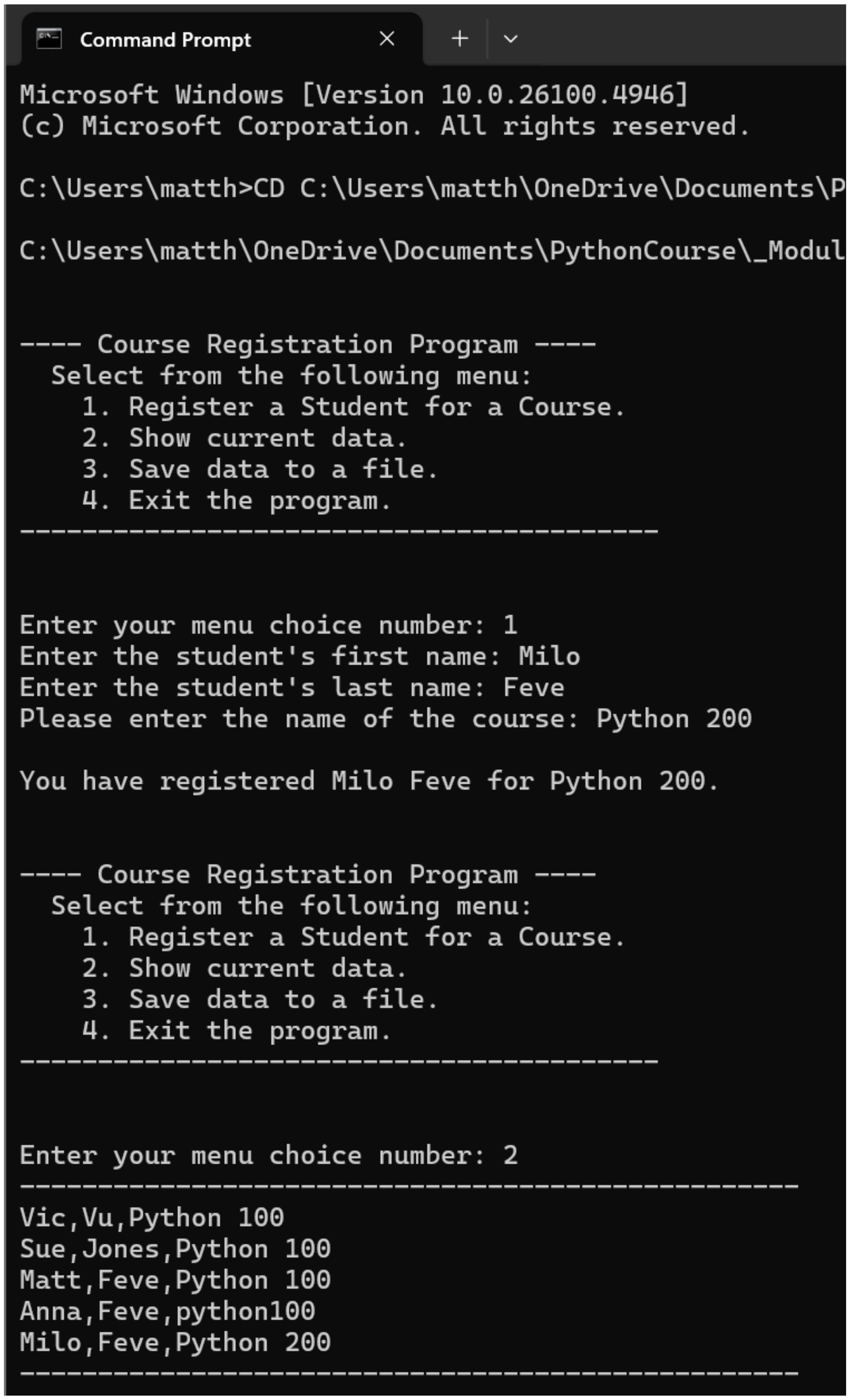
After Program Run:



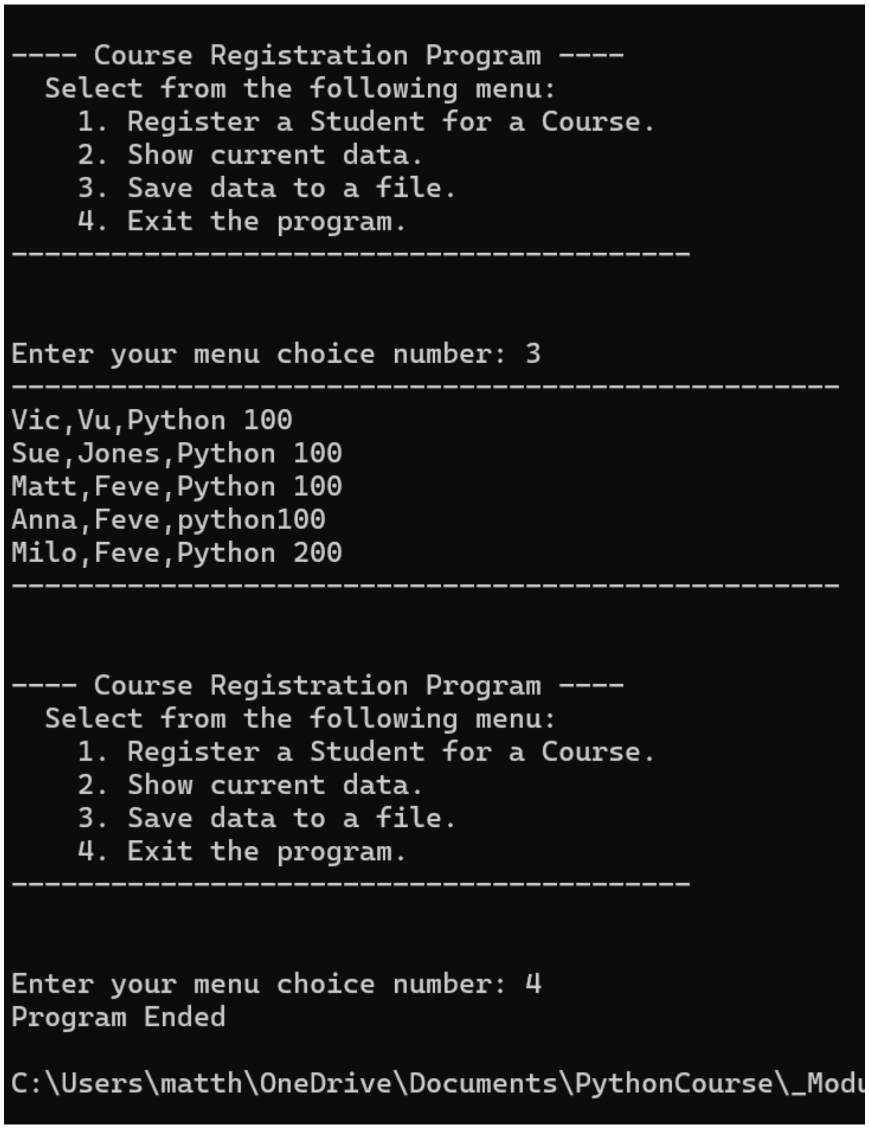
Assignment07.py
Enrollments.json
Assignment07-Starter.py
PythonCourse > _Module07 > myAssignment > Enrollments.json > ...

```json
[
    {
        "FirstName": "Vic",
        "LastName": "Vu",
        "CourseName": "Python 100"
    },
    {
        "FirstName": "Sue",
        "LastName": "Jones",
        "CourseName": "Python 100"
    },
    {
        "FirstName": "Matt",
        "LastName": "Feve",
        "CourseName": "Python 100"
    }
]
```

Appendix C – Command Prompt Terminal Results



```
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\matth>CD C:\Users\matth\OneDrive\Documents\P

C:\Users\matth\OneDrive\Documents\PythonCourse\_Modul


---- 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: Milo
Enter the student's last name: Feve
Please enter the name of the course: Python 200

You have registered Milo Feve for Python 200.


---- 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: 2
--------------------------------------------------
Vic,Vu,Python 100
Sue,Jones,Python 100
Matt,Feve,Python 100
Anna,Feve,python100
Milo,Feve,Python 200
--------------------------------------------------
```



```
---- 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: 3
--------------------------------------------------
Vic,Vu,Python 100
Sue,Jones,Python 100
Matt,Feve,Python 100
Anna,Feve,python100
Milo,Feve,Python 200
--------------------------------------------------



---- 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: 4
Program Ended

C:\Users\matth\OneDrive\Documents\PythonCourse\_Modu
```