Giovanna Torres

3/12/2025

IT FDN 110 A Wi 25: Foundations Of Programming: Python

Assignment 7

GitHub Link: https://github.com/giozoar/IntroToProg-Python-Mod07

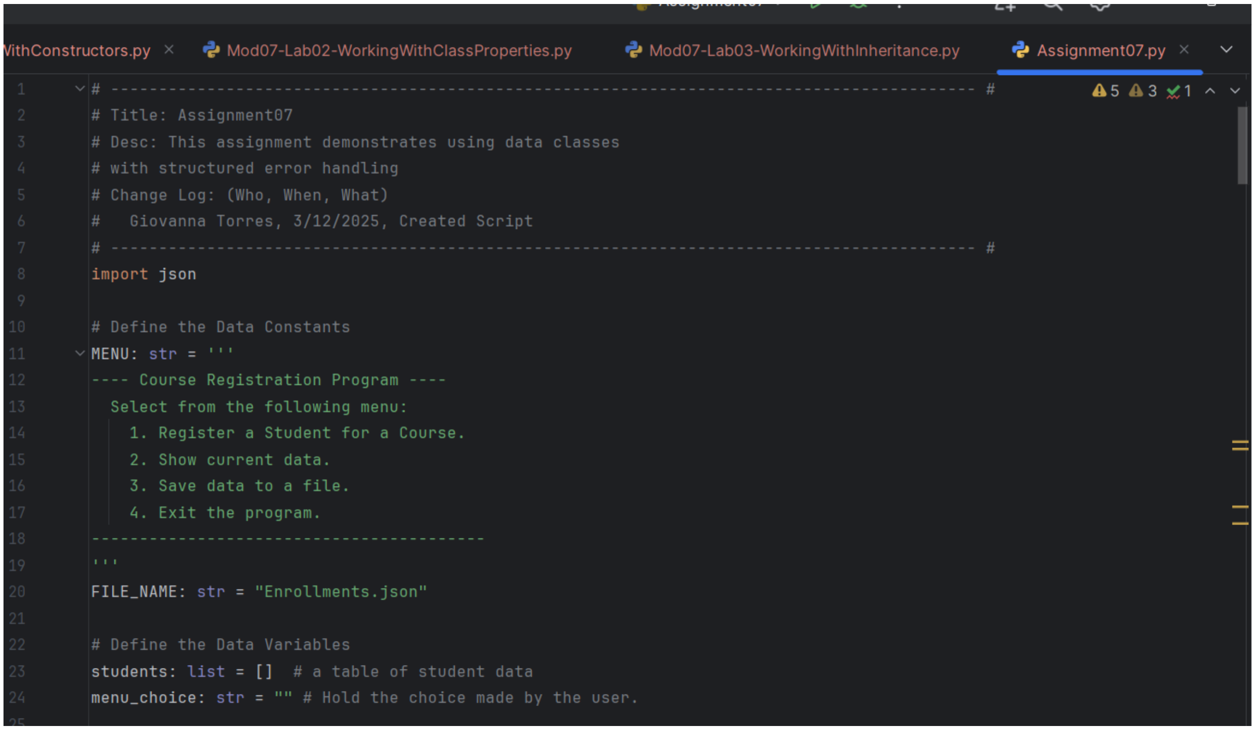
Classes and Objects

Introduction

This week as a part of my Foundations of Programming course, I learned extensions of built-in functions and classes, which allow me to organize my script in a logical, modular fashion and increase maintainability. I also learned about inheritance, properties, and getting and setting functions. These built-in methods for Python help me easily define behaviors for my script.

Creating the Script

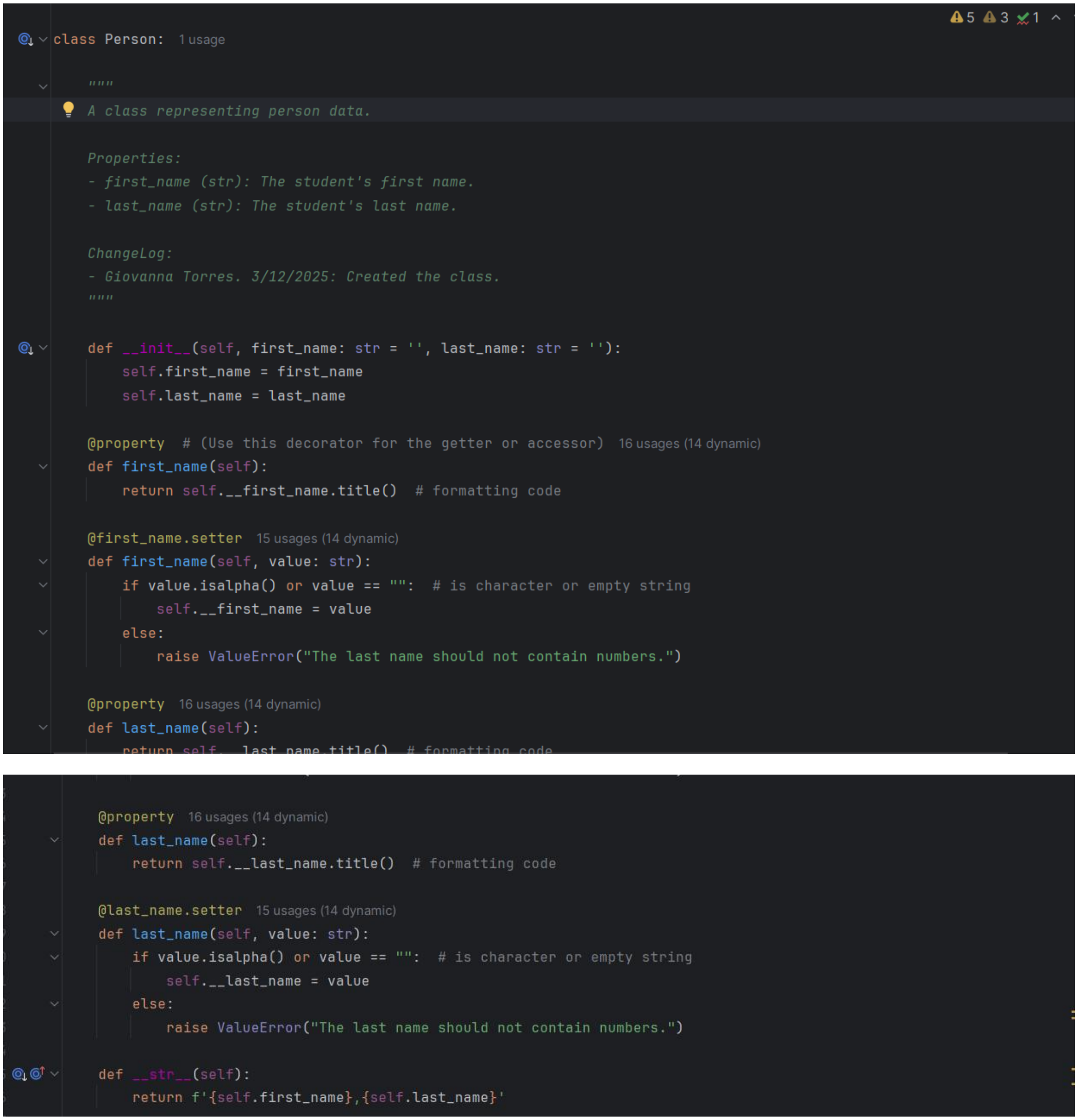
After reading the acceptance criteria described in the Mod07-Assignment file, I began to work on my script using the PyCharm Community Edition IDE. I reused the header and base code from the Assignment07-Starter.py file included in the module materials to display the necessary information.


WithConstructors.py
Mod07-Lab02-WorkingWithClassProperties.py
Mod07-Lab03-WorkingWithInheritance.py
Assignment07.py
5
3
1

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
# with structured error handling
# Change Log: (Who, When, What)
#   Giovanna Torres, 3/12/2025, 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 - Script Header and constants

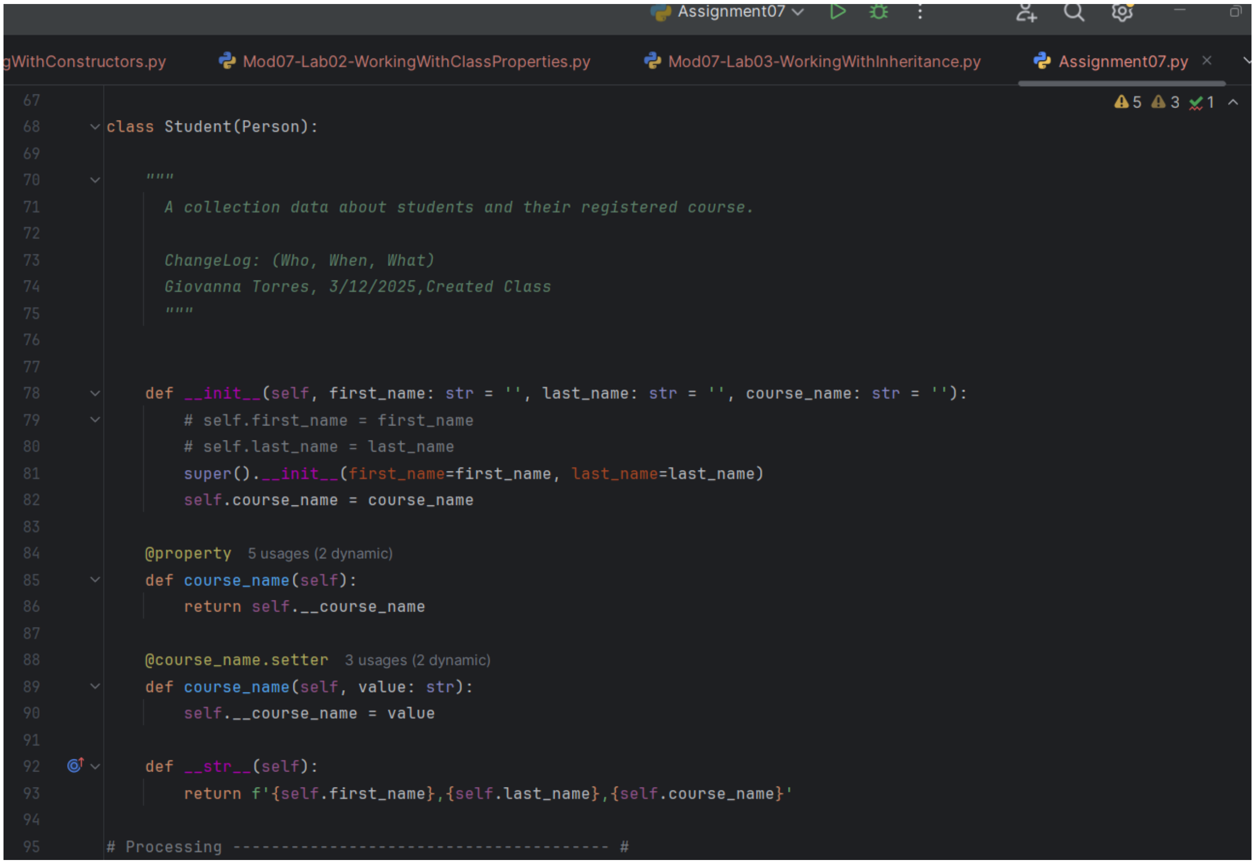
After verifying the constants, I then created the two ‘parent’ and ‘child’ classes ‘ Person’ and ‘Student’, respectively. ‘Person’ holds the high-level identifying characteristics and class structure, and ‘Student’ holds specific information in addition to that of the ‘Person’ class. ‘Student’ will therefore inherit the structure from ‘Person’. I additionally created ‘getting’ and ‘setting’ properties for the class attributes of first_name, last_name, and course_name.



```
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:
    - Giovanna Torres. 3/12/2025: 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)  16 usages (14 dynamic)
    def first_name(self):
        return self.__first_name.title()  # formatting code

    @first_name.setter  15 usages (14 dynamic)
    def first_name(self, value: str):
        if value.isalpha() or value == "":  # is character or empty string
            self.__first_name = value
        else:
            raise ValueError("The last name should not contain numbers.")

    @property  16 usages (14 dynamic)
    def last_name(self):
        return self.__last_name.title()  # formatting code
```

```
    @property  16 usages (14 dynamic)
    def last_name(self):
        return self.__last_name.title()  # formatting code

    @last_name.setter  15 usages (14 dynamic)
    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}'
```

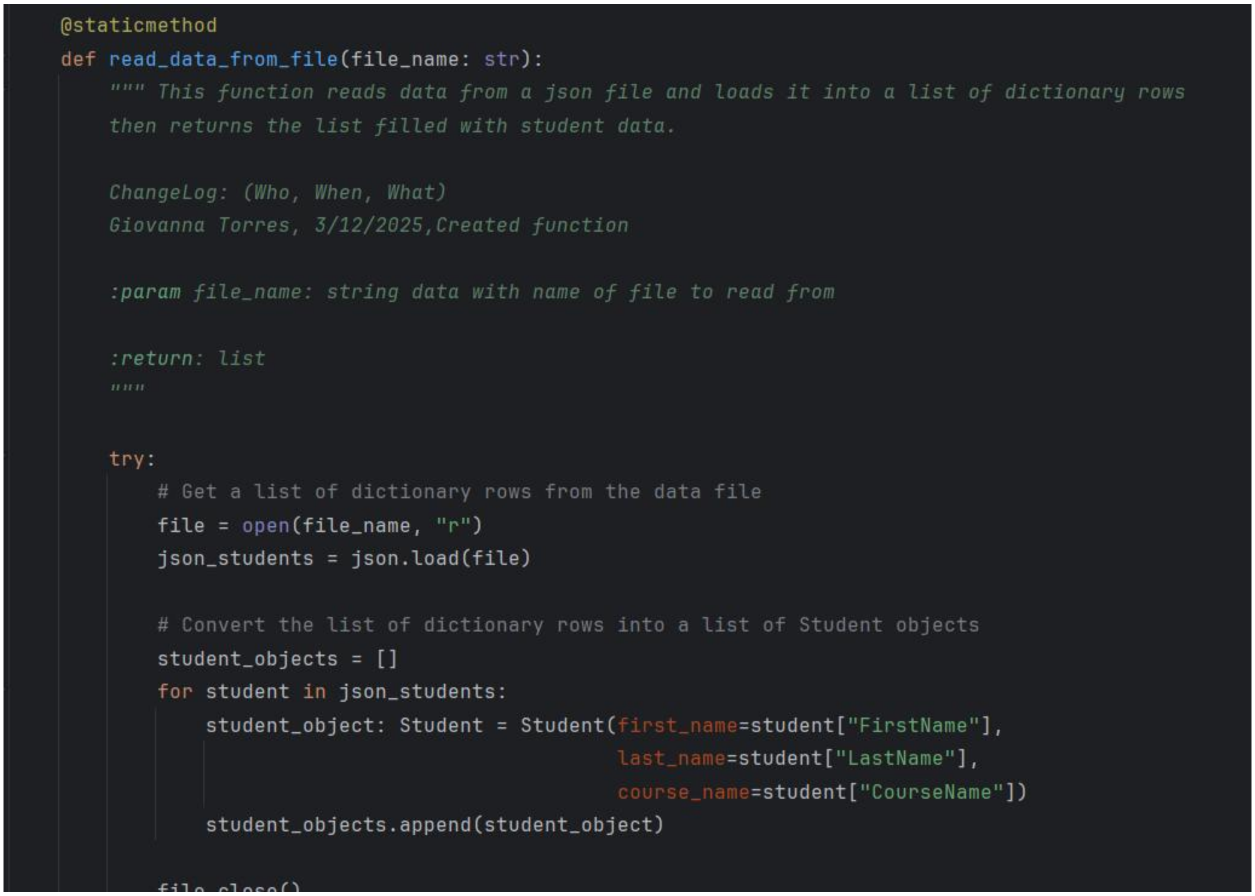
Figure 2 – ‘Person’ parent class



```python

class Student(Person):

    """
      A collection data about students and their registered course.

      ChangeLog: (Who, When, What)
      Giovanna Torres, 3/12/2025,Created Class
      """


    def __init__(self, first_name: str = '', last_name: str = '', course_name: str = ''):
        # self.first_name = first_name
        # self.last_name = last_name
        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):
        self.__course_name = value

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

# Processing --------------------------------------- #
```

Figure 3 – 'Student' child class

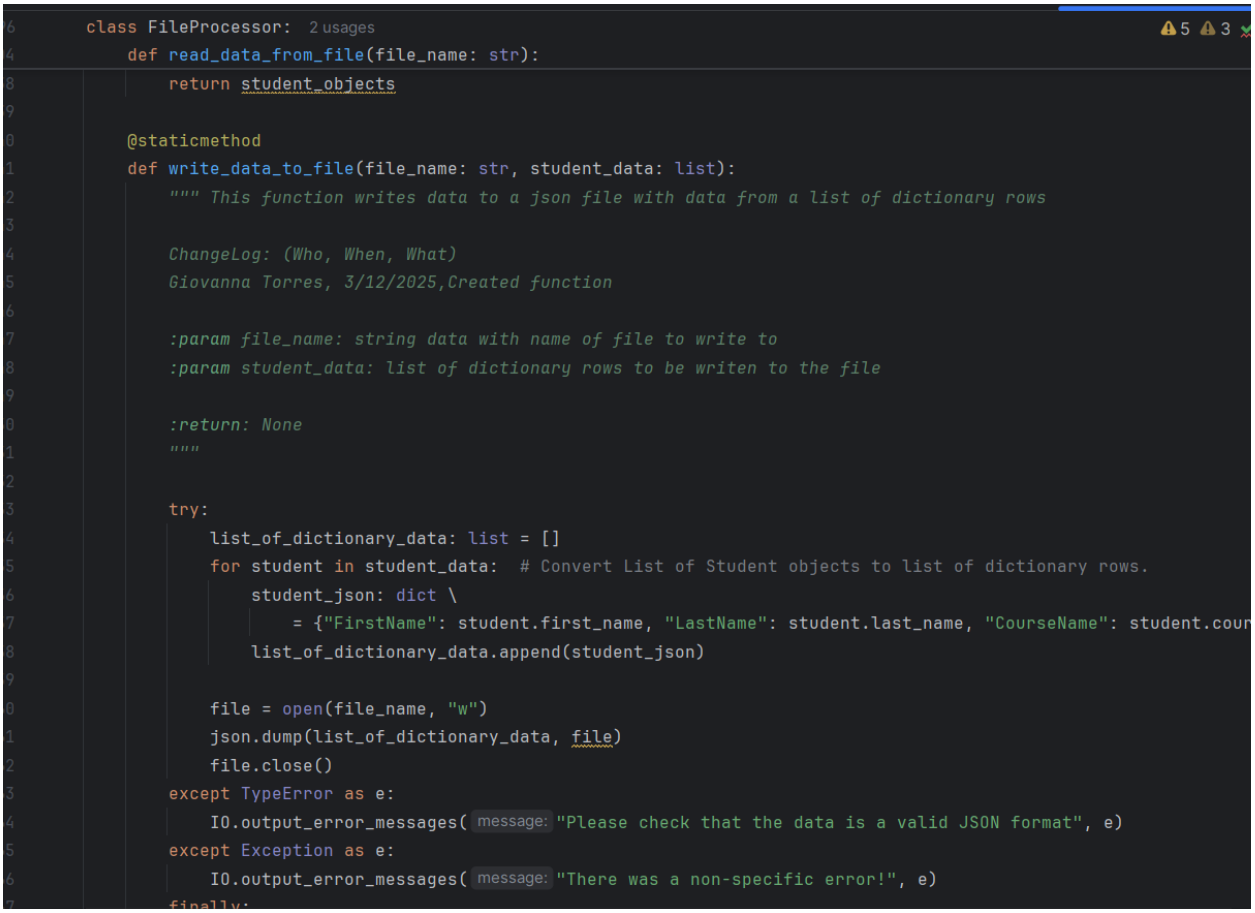
After creating my new classes, I then began modifying my existing 'IO' and 'FileProcessing' classes to use the 'Student' class attributes to read, write, and create new data. The script was reformatted to read JSON data from a file, then convert it into an object instance of the class and append to a list of objects.



```
    @staticmethod
    def read_data_from_file(file_name: str):
        """ This function reads data from a json file and loads it into a list of dictionary rows
        then returns the list filled with student data.

        ChangeLog: (Who, When, What)
        Giovanna Torres, 3/12/2025,Created function

        :param file_name: string data with name of file to read from

        :return: list
        """

        try:
            # Get a list of dictionary rows from the data file
            file = open(file_name, "r")
            json_students = json.load(file)

            # Convert the list of dictionary rows into a list of Student objects
            student_objects = []
            for student in json_students:
                student_object: Student = Student(first_name=student["FirstName"],
                                                  last_name=student["LastName"],
                                                  course_name=student["CourseName"])
                student_objects.append(student_object)

            file.close()
```

Figure 4 – read_data_from_file method in the FileProcessing class modified using the 'Student' class to create an object instance.

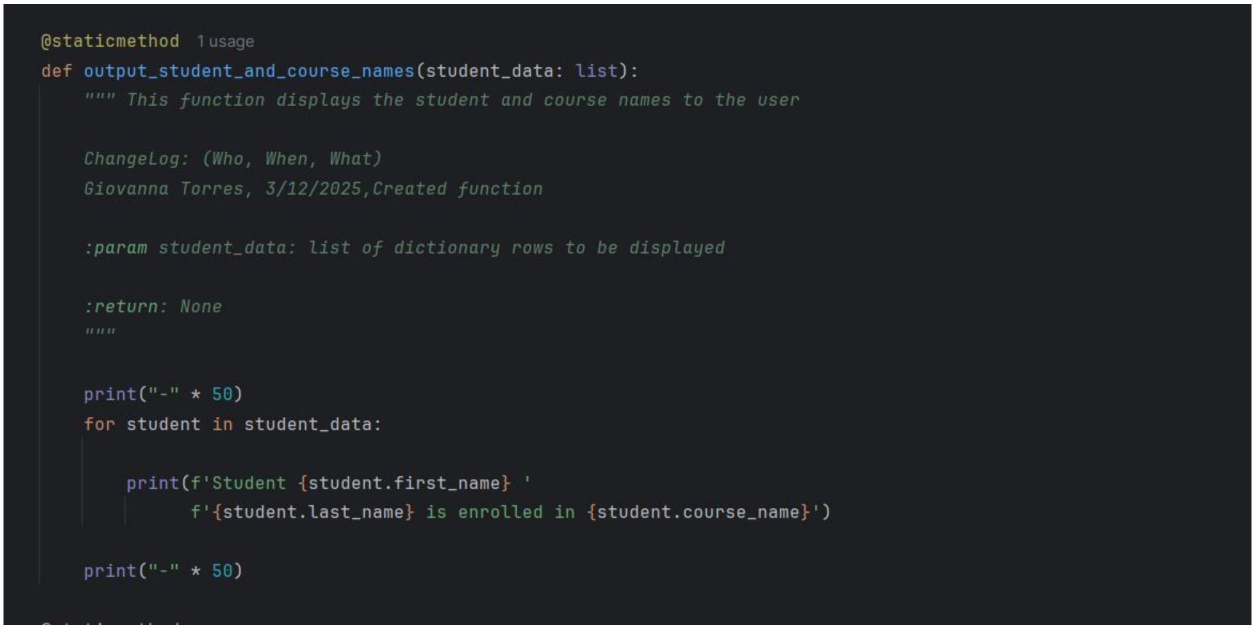
I then modified the write_data_to_file method to convert the object data into a JSON format to write to a JSON file.



```
class FileProcessor:  2 usages
    def read_data_from_file(file_name: str):
        return student_objects

    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
        """ This function writes data to a json file with data from a list of dictionary rows

        ChangeLog: (Who, When, What)
        Giovanna Torres, 3/12/2025,Created function

        :param file_name: string data with name of file to write to
        :param student_data: list of dictionary rows to be writen to the file

        :return: None
        """

        try:
            list_of_dictionary_data: list = []
            for student in student_data:  # Convert List of Student objects to list of dictionary rows.
                student_json: dict \
                    = {"FirstName": student.first_name, "LastName": student.last_name, "CourseName": student.cour
                list_of_dictionary_data.append(student_json)

            file = open(file_name, "w")
            json.dump(list_of_dictionary_data, file)
            file.close()
        except TypeError as e:
            IO.output_error_messages( message: "Please check that the data is a valid JSON format", e)
        except Exception as e:
            IO.output_error_messages( message: "There was a non-specific error!", e)
        finally:
```

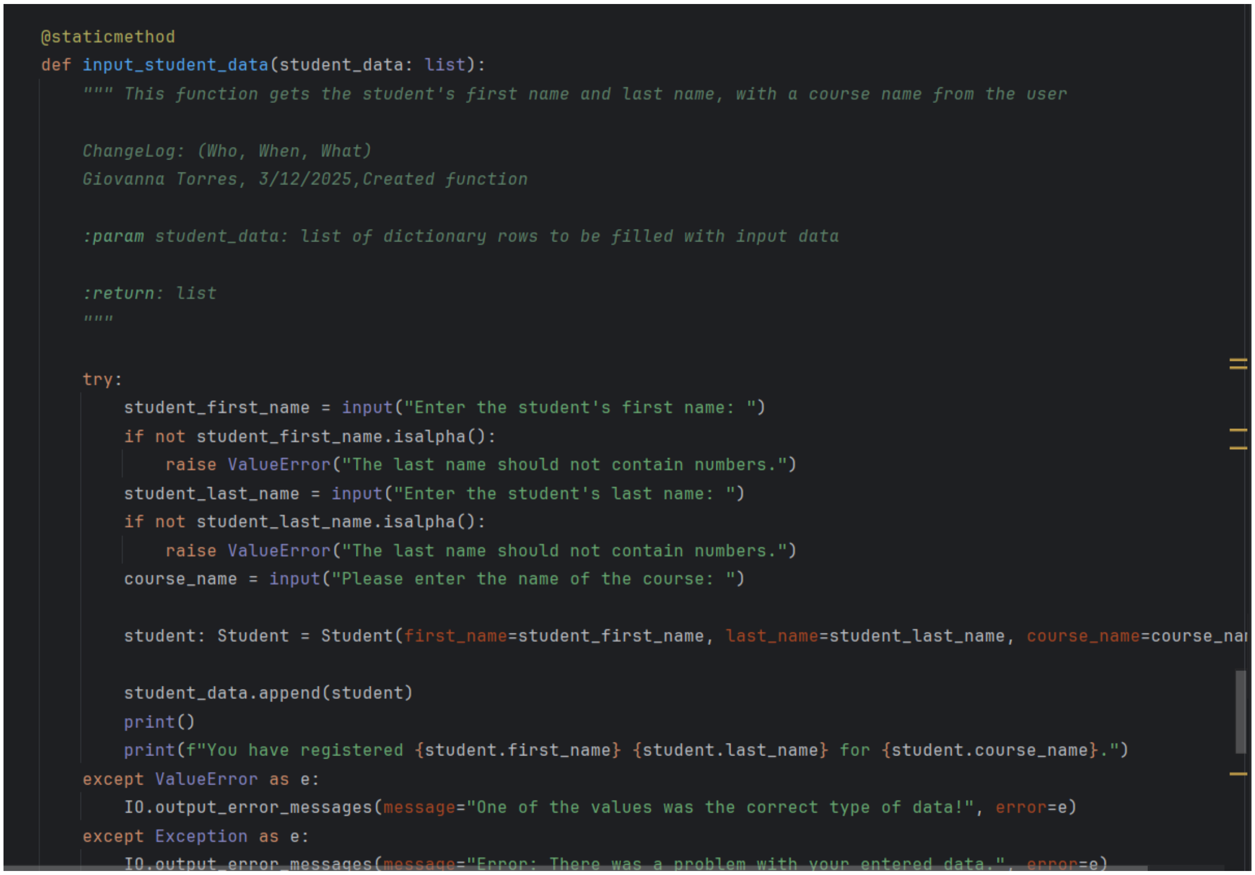
Figure 5 - Converting object data to JSON to write to file.

For my input and output of student data, I changed the statements to use the student class format to display the appropriate data from the list of objects, and append new data to the object list.



```python
@staticmethod  1 usage
def output_student_and_course_names(student_data: list):
    """ This function displays the student and course names to the user

    ChangeLog: (Who, When, What)
    Giovanna Torres, 3/12/2025,Created function

    :param student_data: list of dictionary rows to be displayed

    :return: None
    """

    print("-" * 50)
    for student in student_data:

        print(f'Student {student.first_name} '
              f'{student.last_name} is enrolled in {student.course_name}')

    print("-" * 50)
```

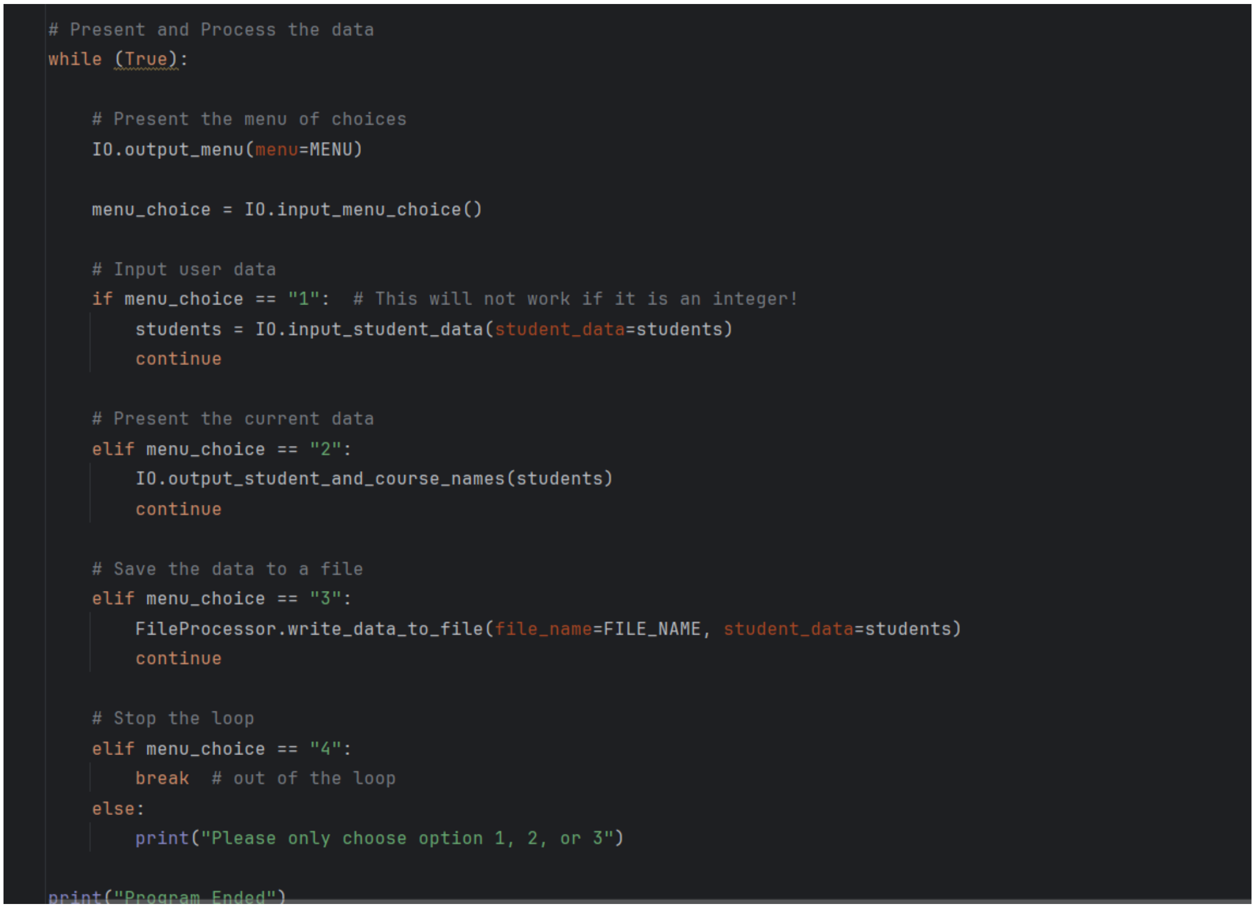
Figure 6 - Formatting the output data with class structure.



```python
@staticmethod
def input_student_data(student_data: list):
    """ This function gets the student's first name and last name, with a course name from the user

    ChangeLog: (Who, When, What)
    Giovanna Torres, 3/12/2025,Created function

    :param student_data: list of dictionary rows to be filled with input data

    :return: list
    """

    try:
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("The last name should not contain numbers.")
        student_last_name = input("Enter the student's last name: ")
        if not student_last_name.isalpha():
            raise ValueError("The last name should not contain numbers.")
        course_name = input("Please enter the name of the course: ")

        student: Student = Student(first_name=student_first_name, last_name=student_last_name, course_name=course_na

        student_data.append(student)
        print()
        print(f"You have registered {student.first_name} {student.last_name} for {student.course_name}.")
    except ValueError as e:
        IO.output_error_messages(message="One of the values was the correct type of data!", error=e)
    except Exception as e:
        IO.output_error_messages(message="Error: There was a problem with your entered data.", error=e)
```

Figure 7 - Creating an object from the 'Student' class with user input.

Finally, I verified that no changes were needed in the main script body, since it used the same file IO and processing classes to display, therefore proving that the main script does not need to concern itself with the details of how the data is organized, but with how it should act on the data.



```
# 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_and_course_names(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")
```

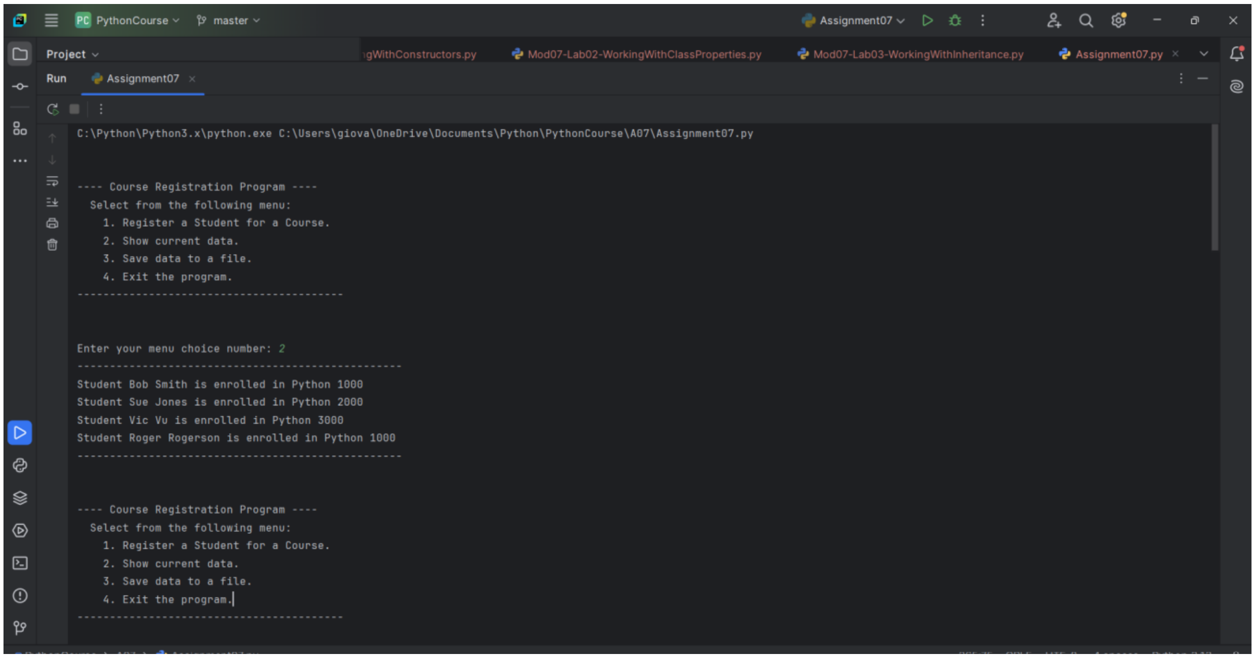
Figure 8 - Reused main body from module 6. No changes were made to accommodate new classes.

I then proceeded to test my code, discussed in the next section.

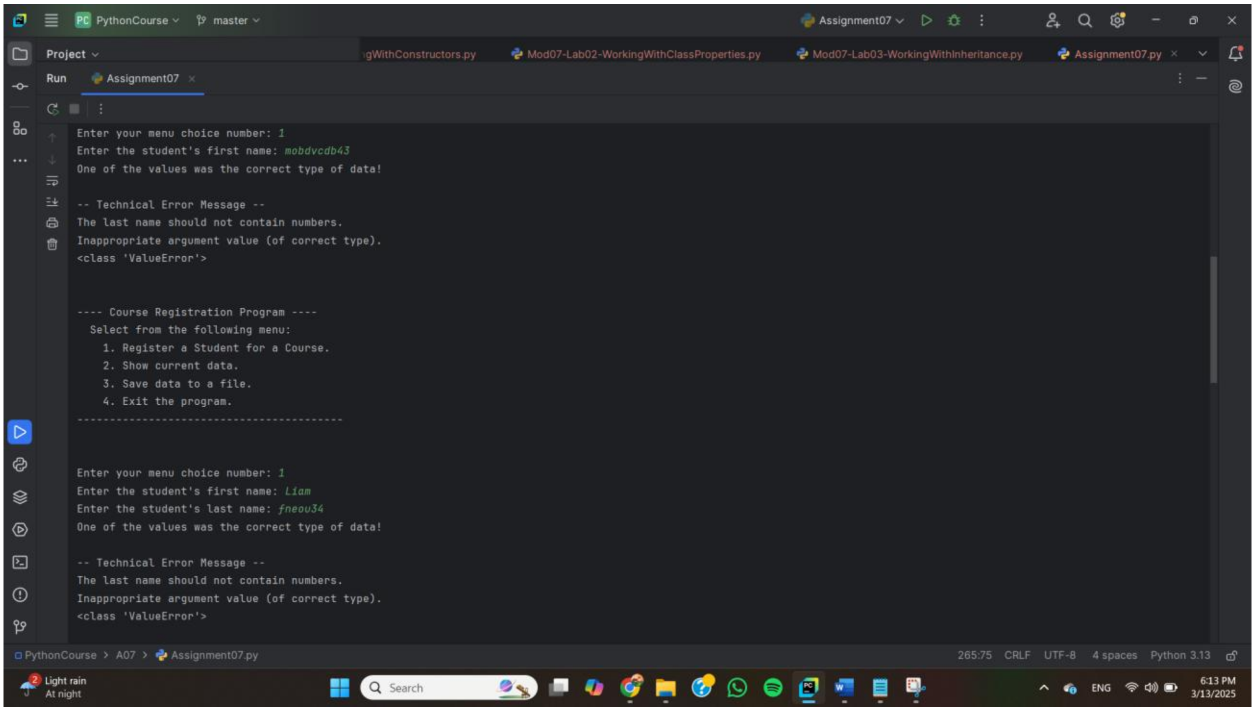
Testing the Script

After setting up my initial 'Enrollments.json' data file with data from module 6 and running the script in the IDE to ensure that the correct outputs were being displayed and created in the directory, I proceeded to run the script in the Command Prompt terminal window. I changed the directory over to my 'A07' file within my PythonCourse directory and ran the script.

I tested multiple cases, including adding multiple names, and adding characters that weren't the numerical in the student name fields, and 1-4 menu option values. As I expected, the script would prompt the user to select another option if the input was invalid. It would also display multiple entries of student registrations and write them to the file as expected. The outputs were the same in either window, as seen below.

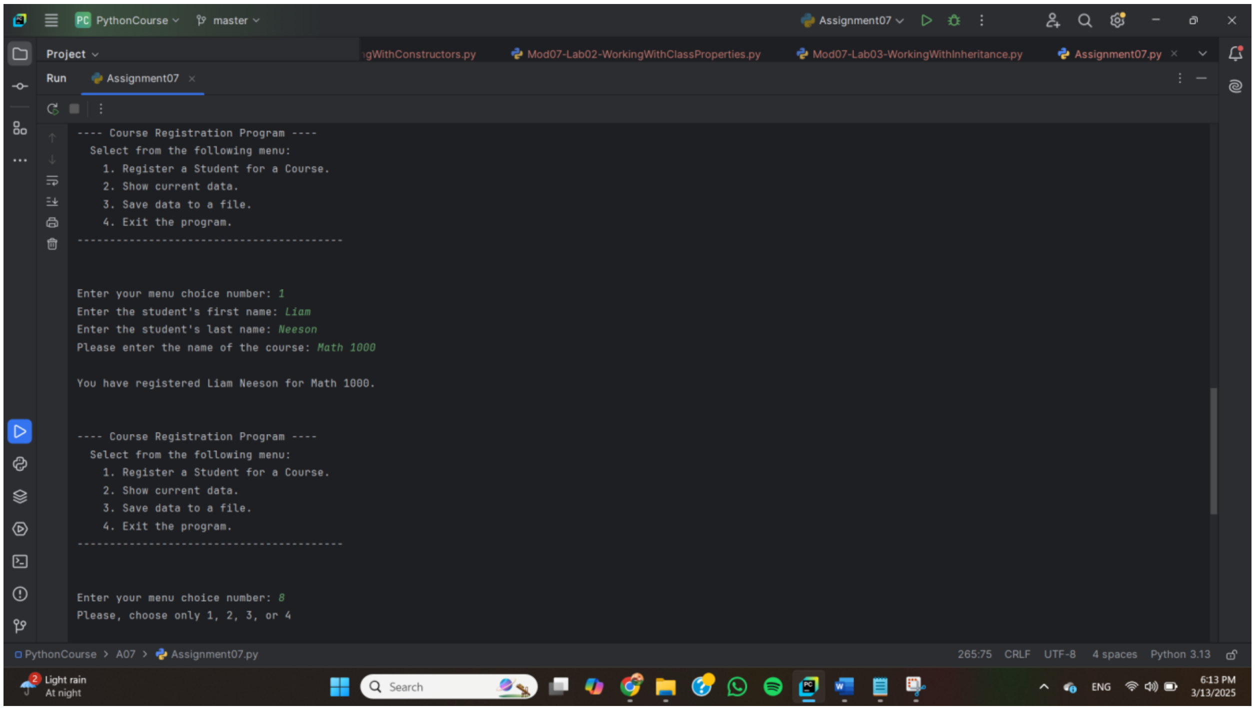


```
C:\Python\Python3.x\python.exe C:\Users\giova\OneDrive\Documents\Python\PythonCourse\A07\Assignment07.py


---- 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
--------------------------------------------------
Student Bob Smith is enrolled in Python 1000
Student Sue Jones is enrolled in Python 2000
Student Vic Vu is enrolled in Python 3000
Student Roger Rogerson is enrolled in Python 1000
--------------------------------------------------



---- 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.
-----------------------------------------
```



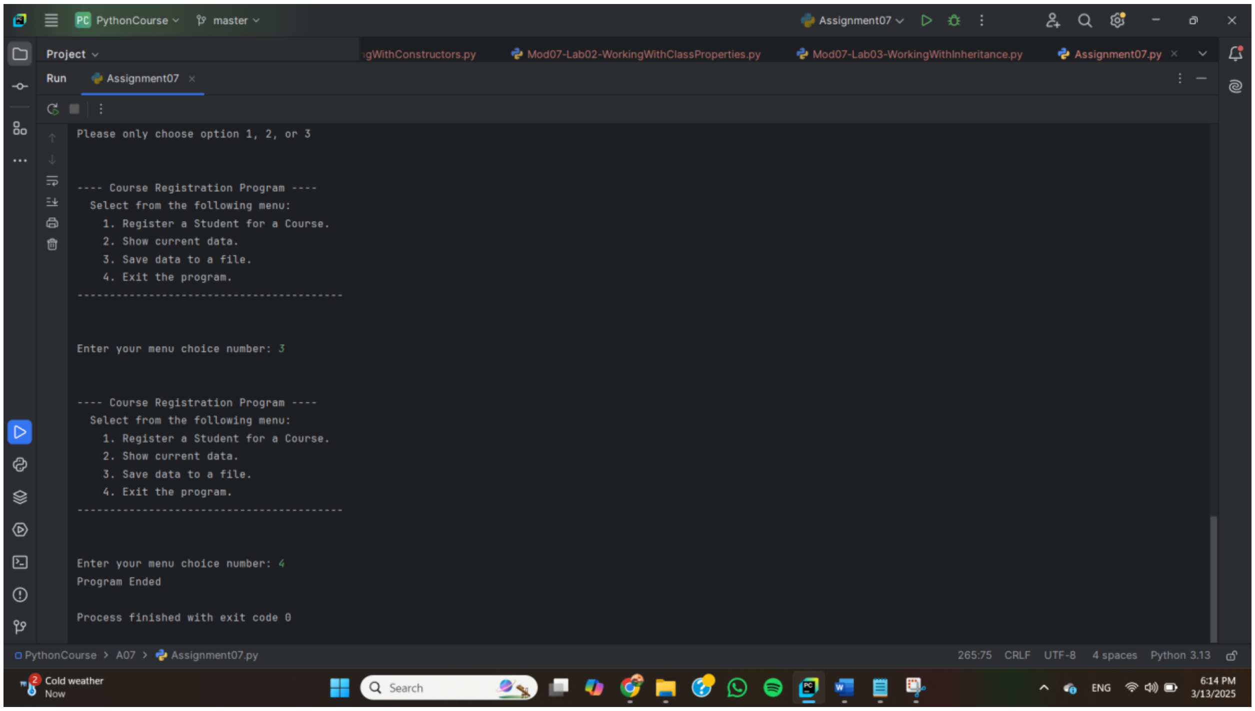
PythonCourse
master
Assignment07
Project
gWithConstructors.py
Mod07-Lab02-WorkingWithClassProperties.py
Mod07-Lab03-WorkingWithInheritance.py
Assignment07.py
Run
Assignment07
PythonCourse > A07 > Assignment07.py
265:75 CRLF UTF-8 4 spaces Python 3.13
Light rain
At night
Search
ENG
6:13 PM
3/13/2025

```
Enter your menu choice number: 1
Enter the student's first name: mobdvcdb43
One of the values was the correct type of data!

-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>



---- 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: Liam
Enter the student's last name: fneou34
One of the values was the correct type of data!

-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>
```



PythonCourse
master
Assignment07
Project
gWithConstructors.py
Mod07-Lab02-WorkingWithClassProperties.py
Mod07-Lab03-WorkingWithInheritance.py
Assignment07.py
Run
Assignment07
PythonCourse > A07 > Assignment07.py
265:75 CRLF UTF-8 4 spaces Python 3.13
Light rain
At night
Search
ENG
6:13 PM
3/13/2025

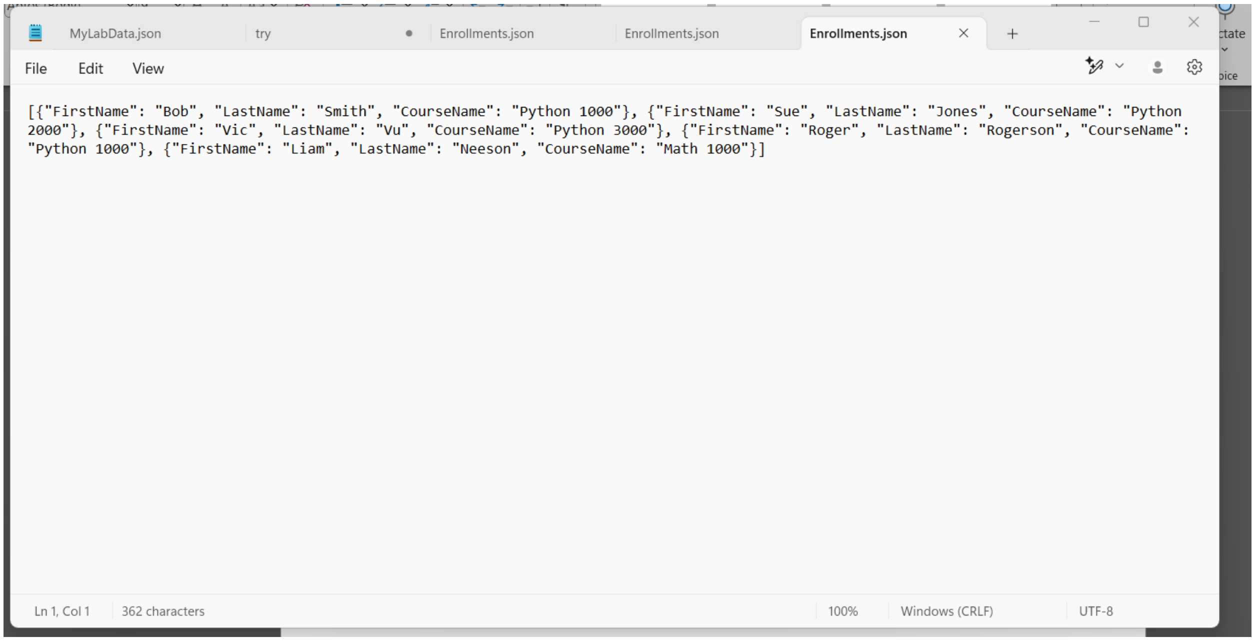
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------


Enter your menu choice number: 1
Enter the student's first name: Liam
Enter the student's last name: Neeson
Please enter the name of the course: Math 1000

You have registered Liam Neeson for Math 1000.


---- 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: 8
Please, choose only 1, 2, 3, or 4
```



PythonCourse
master
Assignment07
Project
gWithConstructors.py
Mod07-Lab02-WorkingWithClassProperties.py
Mod07-Lab03-WorkingWithInheritance.py
Assignment07.py
Run
Assignment07
PythonCourse > A07 > Assignment07.py
265:75 CRLF UTF-8 4 spaces Python 3.13
Cold weather
Now
Search
ENG
6:14 PM
3/13/2025

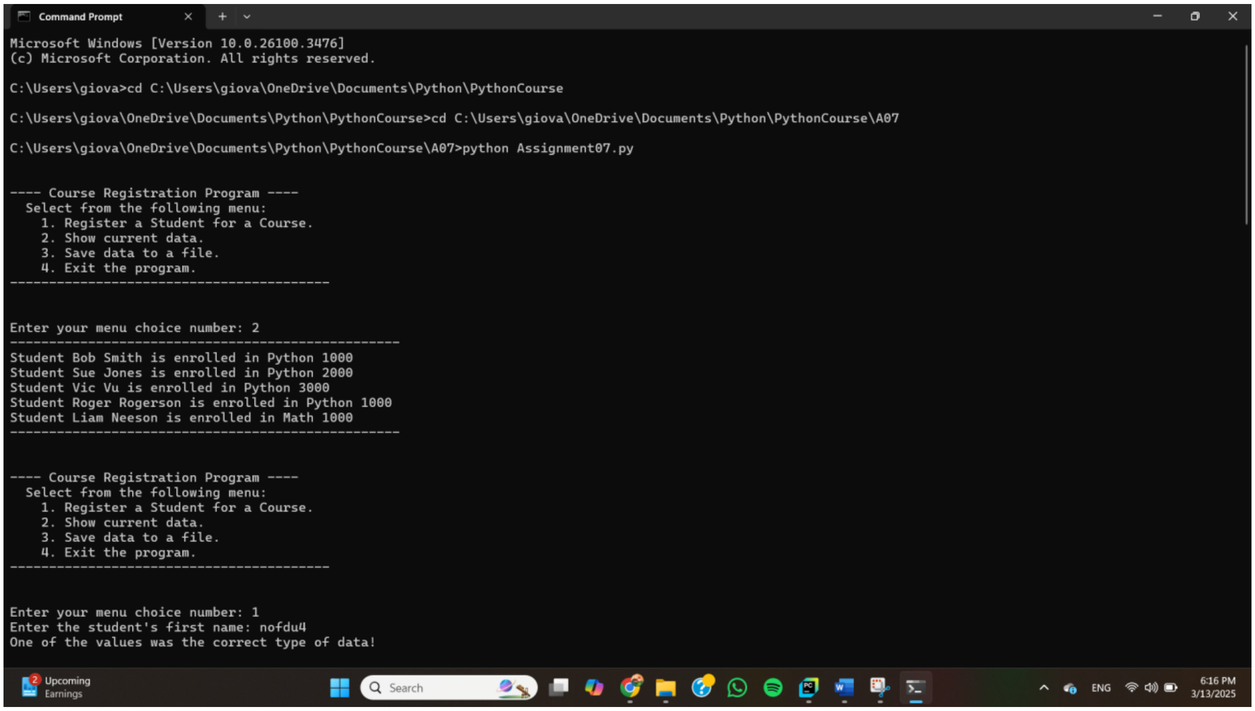
```
Please only choose option 1, 2, or 3


---- 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


---- 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

Process finished with exit code 0
```

Figure 9 - Testing inputs in PyCharm IDE.

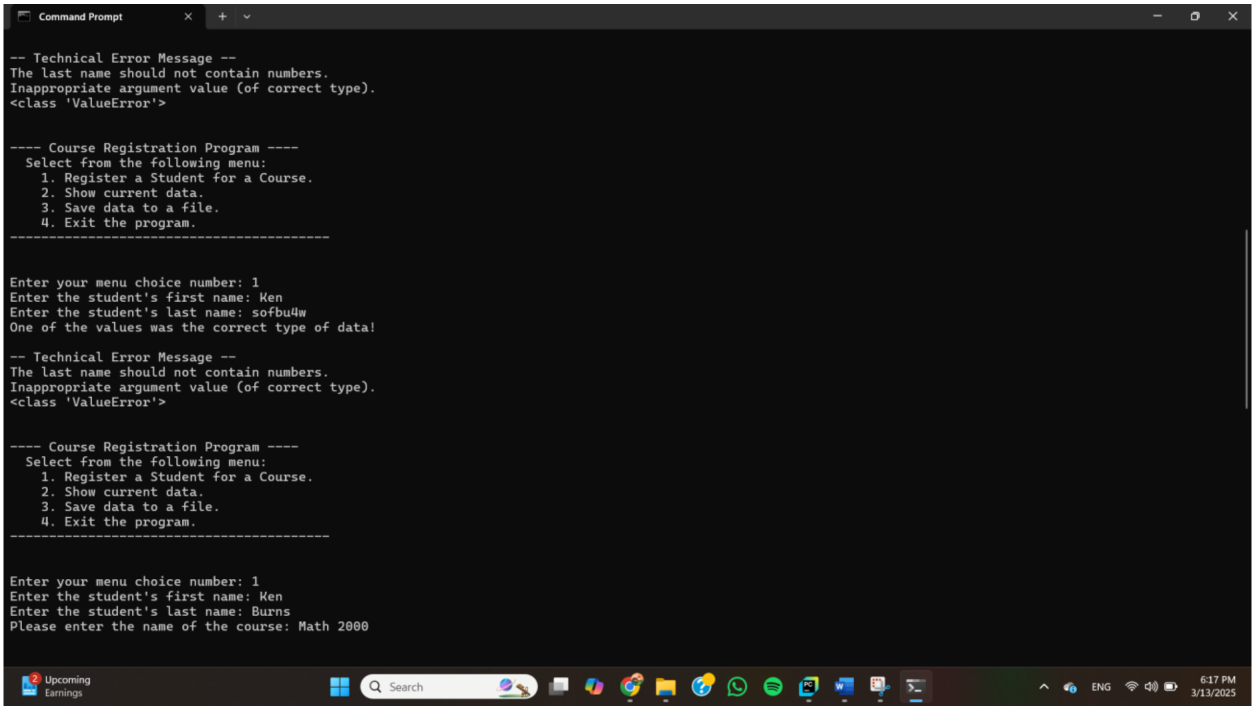


```
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 1000"}, {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python
2000"}, {"FirstName": "Vic", "LastName": "Vu", "CourseName": "Python 3000"}, {"FirstName": "Roger", "LastName": "Rogerson", "CourseName":
"Python 1000"}, {"FirstName": "Liam", "LastName": "Neeson", "CourseName": "Math 1000"}]
```

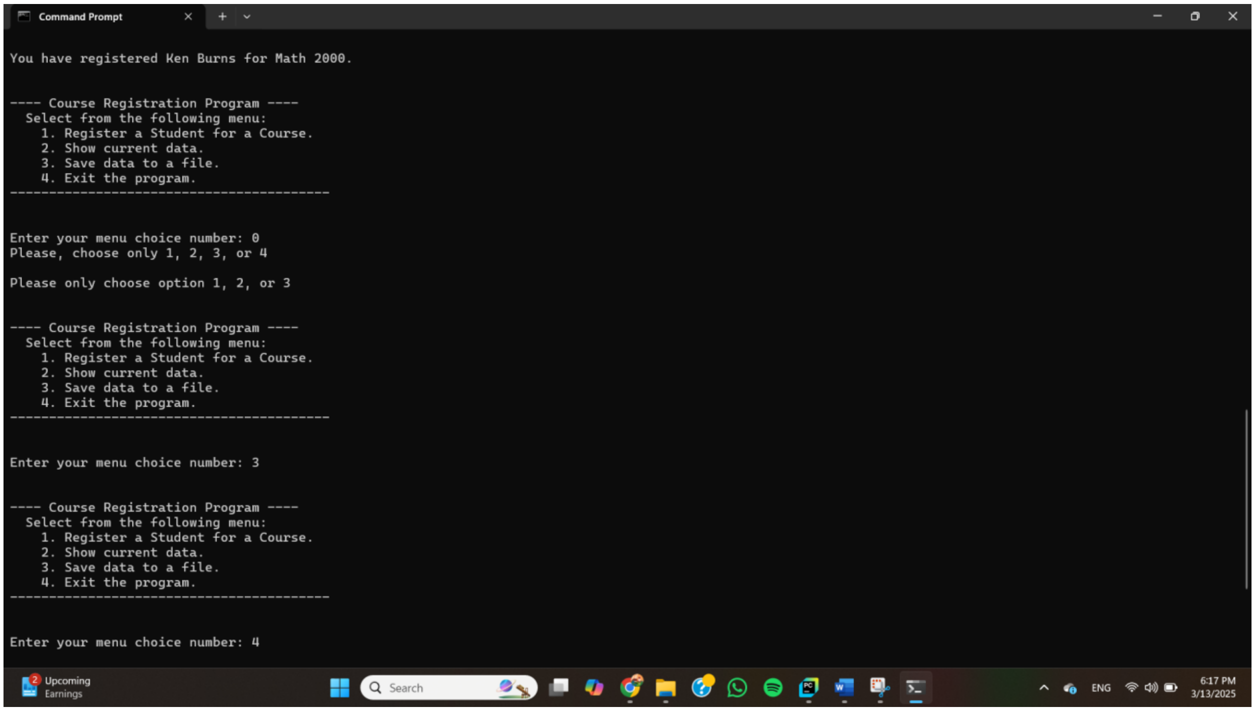
Figure 10 - Outputs in JSON file.



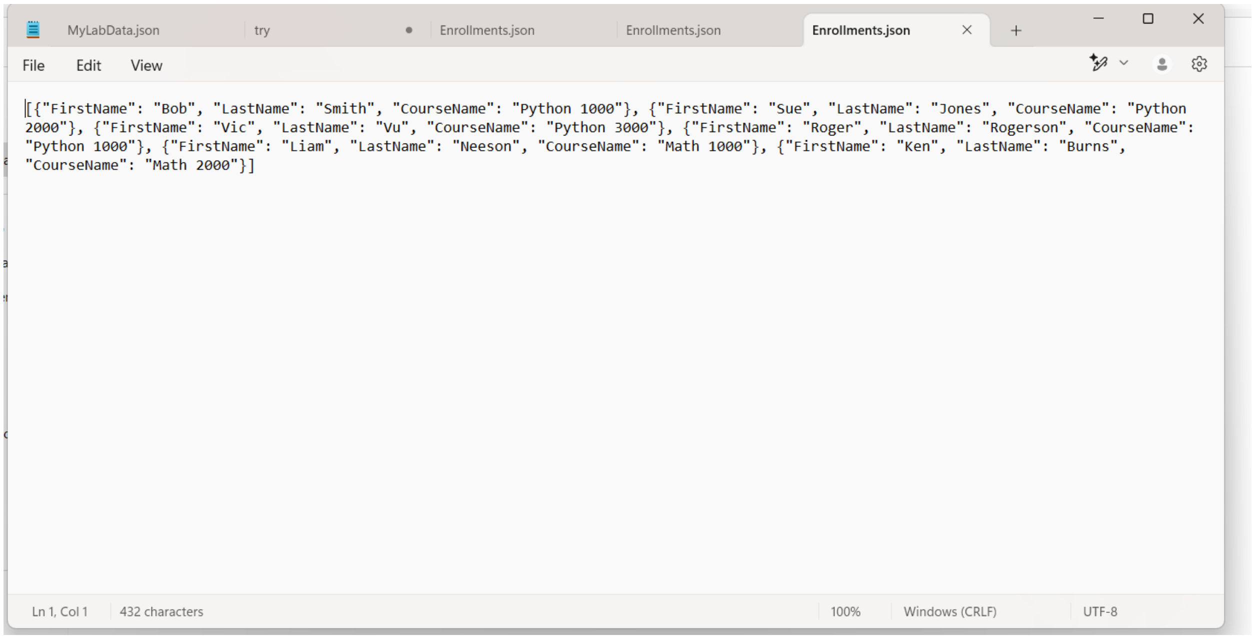
```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\giova>cd C:\Users\giova\OneDrive\Documents\Python\PythonCourse

C:\Users\giova\OneDrive\Documents\Python\PythonCourse>cd C:\Users\giova\OneDrive\Documents\Python\PythonCourse\A07

C:\Users\giova\OneDrive\Documents\Python\PythonCourse\A07>python 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
--------------------------------------------------
Student Bob Smith is enrolled in Python 1000
Student Sue Jones is enrolled in Python 2000
Student Vic Vu is enrolled in Python 3000
Student Roger Rogerson is enrolled in Python 1000
Student Liam Neeson is enrolled in Math 1000
--------------------------------------------------


---- 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: nofdu4
One of the values was the correct type of data!
```



```
-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>


---- 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: Ken
Enter the student's last name: sofbu4w
One of the values was the correct type of data!

-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>


---- 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: Ken
Enter the student's last name: Burns
Please enter the name of the course: Math 2000
```



```
You have registered Ken Burns for Math 2000.


---- 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: 0
Please, choose only 1, 2, 3, or 4

Please only choose option 1, 2, or 3


---- 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


---- 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
```



```
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 1000"}, {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python
2000"}, {"FirstName": "Vic", "LastName": "Vu", "CourseName": "Python 3000"}, {"FirstName": "Roger", "LastName": "Rogerson", "CourseName":
"Python 1000"}, {"FirstName": "Liam", "LastName": "Neeson", "CourseName": "Math 1000"}, {"FirstName": "Ken", "LastName": "Burns",
"CourseName": "Math 2000"}]
```

Figure 11 - Testing in Command Prompt window.

Summary

During this assignment, I learned about using classes and functions to organize my data using Magic Methods such as inheritance and private properties. The biggest gap I found was understanding the behavior of ‘private’ properties, which limit the ability to change them outside of the class. I look forward to learning more about separation of concerns and Python Magic methods for class inheritance, and well as additional use cases for ‘getting’ and ‘setting’ functions.