Assignment 7: Classes and Objects

Rucha P. Nimbalkar

University of Washington

IT FDN 110: Fall 2024

Prof. Randall Root

Nov. 27, 2024

Abstract

Classes help in organizing the code. Constructors and Class methods help in implementing the Object-Oriented Programming principles like abstraction and encapsulation. Data classes play a vital role in the abstraction and encapsulation of data.

Keywords: Classes, Constructors, Methods.

Assignment 7: Classes and Objects

Assignment 7 is the seventh coding assignment of the IT Fundamentals (IT FDN 110 A) course I am taking at University of Washington. The goal of this assignment is to help me understand the usage of objects. Furthermore, I will be re-factoring my code using constructors and class methods in the assignment. This the link to my GitHub account for Assignment 07.

Reflection

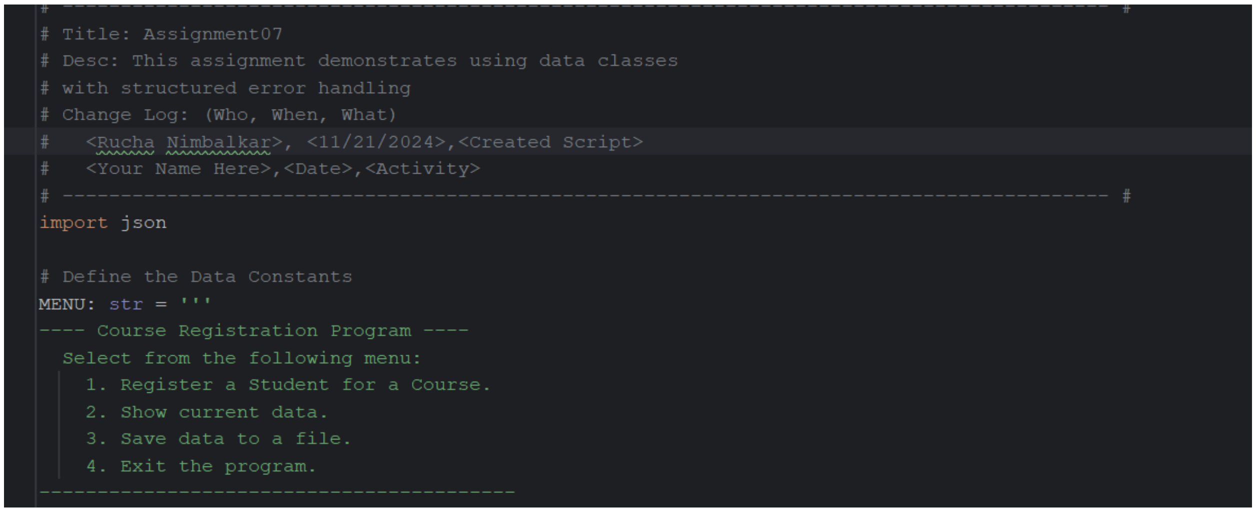
I used classes in Module 06 of this course. We used class methods (or functions) to perform file operations. In Assignment 06, I re-organized the code to include classes and class methods that performed the same operations that were written in the previous assignment. I learnt about code reusability and modularity in Assignment 06. In this assignment (07), I closely applied the Separation of Patterns (SoC) using the data, processing, and presentation sections. According to Professor Root, " Classes are designed to focus on either data, processing, or presentation(input or output)" (2024). In Module 07, I learnt about Object Oriented Programming (OOP) and its feature like isolation, reusability and abstraction while creating more detailed Classes. In this assignment, I used three types of classes, and each class had the purpose of managing data, or processing data in the file and to the file and presentation of data on the console. Professor Root states that, "While processing and presentation classes usually have methods, data Classes typically have Attributes, Constructors, Properties, in addition to Methods." (2024).

Program Summary

I started assignment 07 using the starter file provided in module 07 files, and I updated the header script with my name in it.

Figure 1

Assignment 07 Script Header is updated with my name

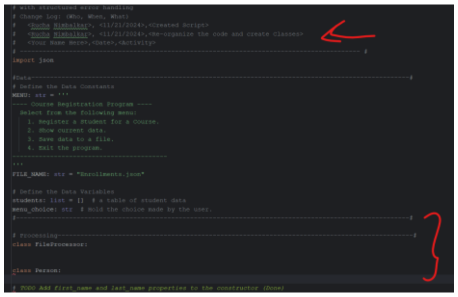


```
# ---------------------------------------------------------------------------------------------- #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
# with structured error handling
# Change Log: (Who, When, What)
#   <Rucha Nimbalkar>, <11/21/2024>,<Created Script>
#   <Your Name Here>,<Date>,<Activity>
# ---------------------------------------------------------------------------------------------- #
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.
-----------------------------------------
```

After that I re-organized the code with some sections using comments and added the two classes "FileProcessor" and "Person" as shown in figure 2.

Figure 2

Assignment 07 Added two classes "FileProcessor" and "Person"

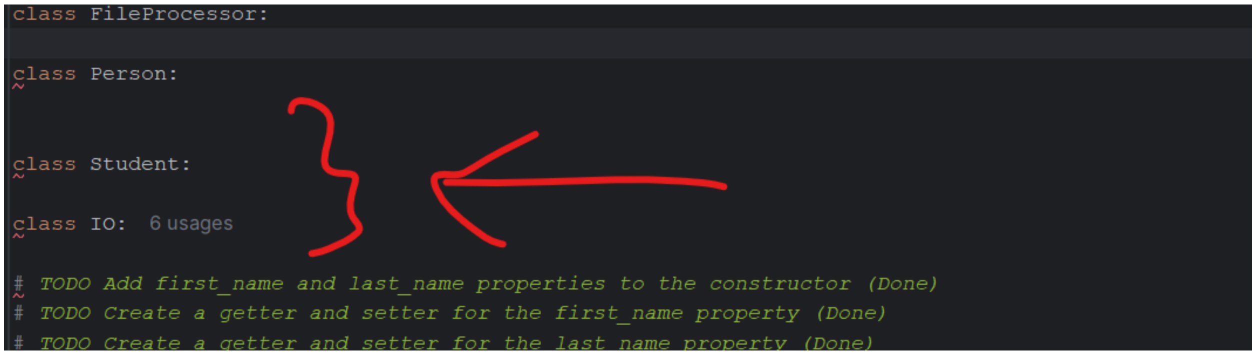


```
# with structured error handling
# Change Log: (Who, When, What)
#   <Rucha Nimbalkar>, <11/21/2024>,<Created Script>
#   <Rucha Nimbalkar>, <11/21/2024>,<Re-organize the code and create Classes>
#   <Your Name Here>,<Date>,<Activity>
# ---------------------------------------------------------------------------------------------- #
import json

#Data------------------------------------------------------------------------------------------#
# 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.
#----------------------------------------------------------------------------------------------#

# Processing-----------------------------------------------------------------------------------#
class FileProcessor:


class Person:

# TODO Add first_name and last_name properties to the constructor (Done)
```

As I was learning the module notes provided by Professor root, and working on my code, I realized that my code would have three classes for different purposes. The “Student” class would store student data. The “Person” class would store person data. The “FileProcessor” class would process the file data, and the “IO” class would present data on the console. So, I added two more classes (“Student” and “IO” to my code).

Figure 3

Assignment 07 Added two additional classes “Student” and “IO”

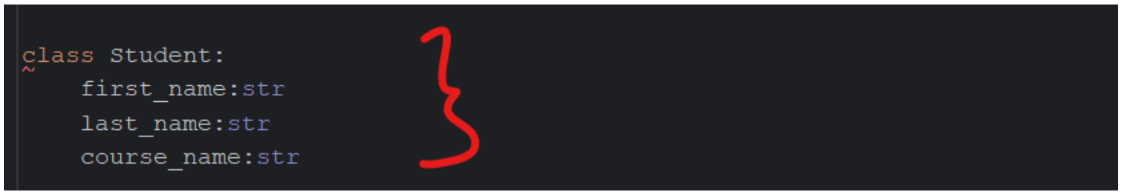


```
class FileProcessor:

class Person:


class Student:

class IO:   6 usages

# TODO Add first_name and last_name properties to the constructor (Done)
# TODO Create a getter and setter for the first_name property (Done)
# TODO Create a getter and setter for the last_name property (Done)
```

I added the attributes first_name, last_name and course_name to the “Student” class.

Figure 4

Assignment 07 Added attributes to the class “Student”

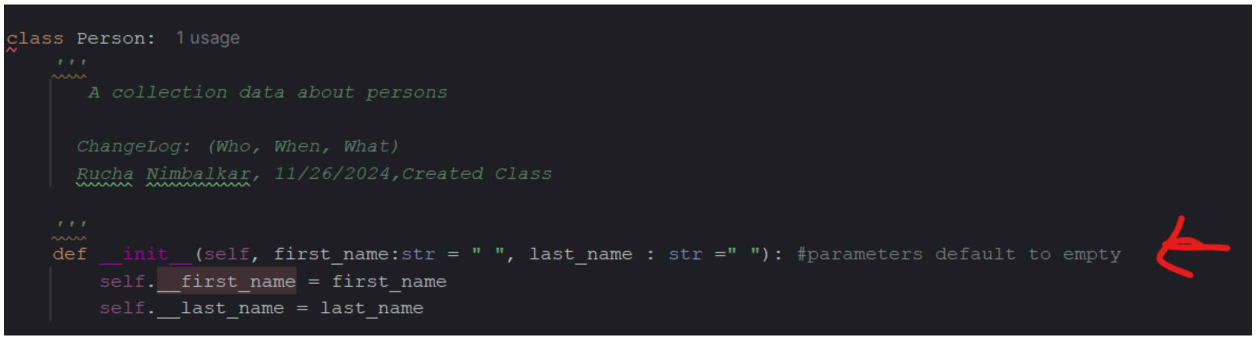


```
class Student:
    first_name:str
    last_name:str
    course_name:str
```

After finishing the reading and practice of notes and code samples provided for this module, I started working on my code. I decided to work on the Person Class first by adding constructor, and then inheriting it in the class Student as shown in figure 5.

Figure 5

Assignment 07 Added constructor for Person class and inherited it in the Student class

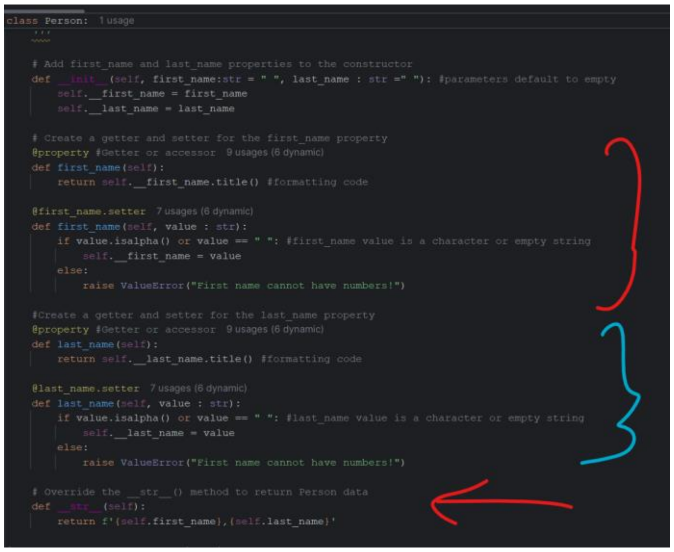


```
class Person:  1 usage
    '''
      A collection data about persons

     ChangeLog: (Who, When, What)
     Rucha Nimbalkar, 11/26/2024,Created Class

    '''
    def __init__(self, first_name:str = " ", last_name : str =" "): #parameters default to empty
        self.__first_name = first_name
        self.__last_name = last_name
```

I added a property for first_name and last_name in the class Person as shown in figure 6. I also added the method to override the __str__() to return Person data.

Figure 6

Assignment 07 Added property for first_name and last_name in the class Person

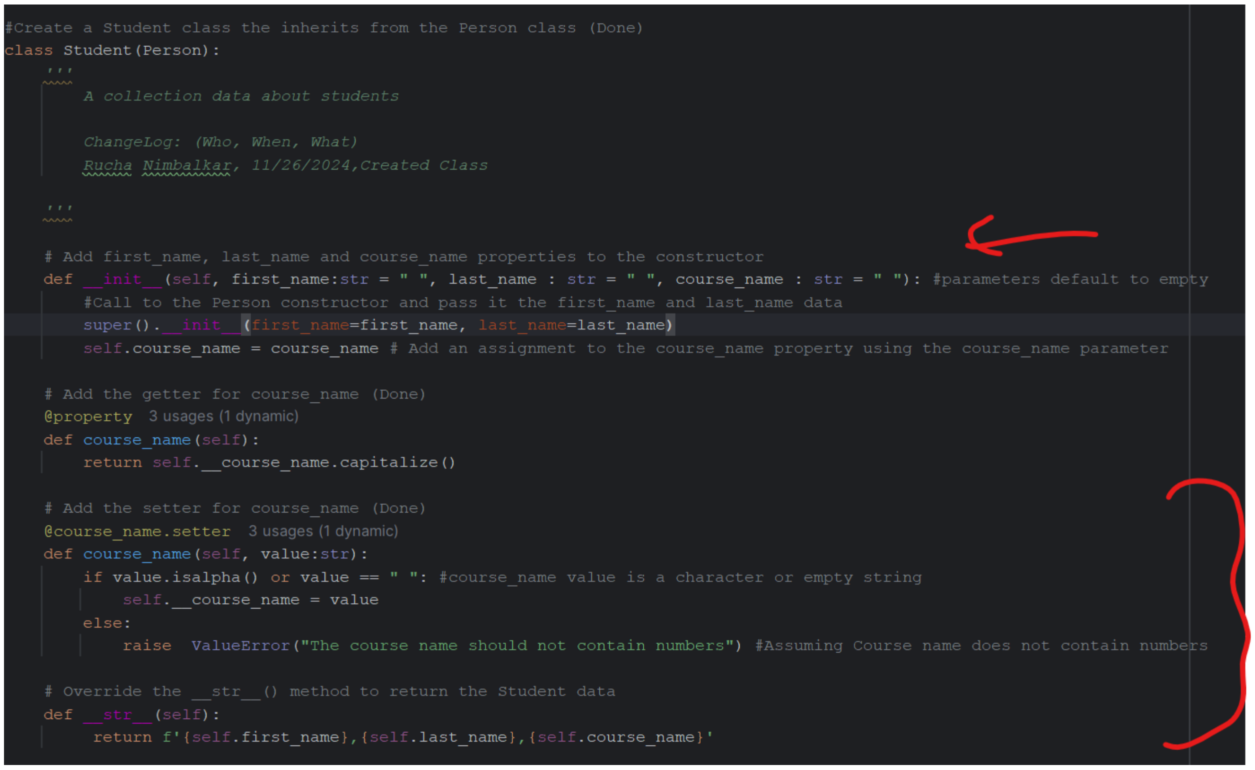


```
class Person:  1 usage
    '''

    # Add first_name and last_name properties to the constructor
    def __init__(self, first_name:str = " ", last_name : str =" "): #parameters default to empty
        self.__first_name = first_name
        self.__last_name = last_name

    # Create a getter and setter for the first_name property
    @property #Getter or accessor  9 usages (6 dynamic)
    def first_name(self):
        return self.__first_name.title() #formatting code

    @first_name.setter  7 usages (6 dynamic)
    def first_name(self, value : str):
        if value.isalpha() or value == " ": #first_name value is a character or empty string
            self.__first_name = value
        else:
            raise ValueError("First name cannot have numbers!")

    #Create a getter and setter for the last_name property
    @property #Getter or accessor  9 usages (6 dynamic)
    def last_name(self):
        return self.__last_name.title() #formatting code

    @last_name.setter  7 usages (6 dynamic)
    def last_name(self, value : str):
        if value.isalpha() or value == " ": #last_name value is a character or empty string
            self.__last_name = value
        else:
            raise ValueError("First name cannot have numbers!")

    # Override the __str__() method to return Person data
    def __str__(self):
        return f'{self.first_name},{self.last_name}'
```

After looking at the Todo comments and carefully looking at the starter file, I realized that the class definitions for FileProcessor and IO were already provided in the starter file, so I removed the two classes I had added earlier (FileProcessor and IO) as they were duplicate. I removed the attributes from the Student class as I learnt I would be using constructor and setter for assigning values.

I followed along with the Todo comments. I added constructor for student class and then I added getter and setter for course_name in the Class student. I also added the method to override the __str__() method to return student data as shown in figure 7.

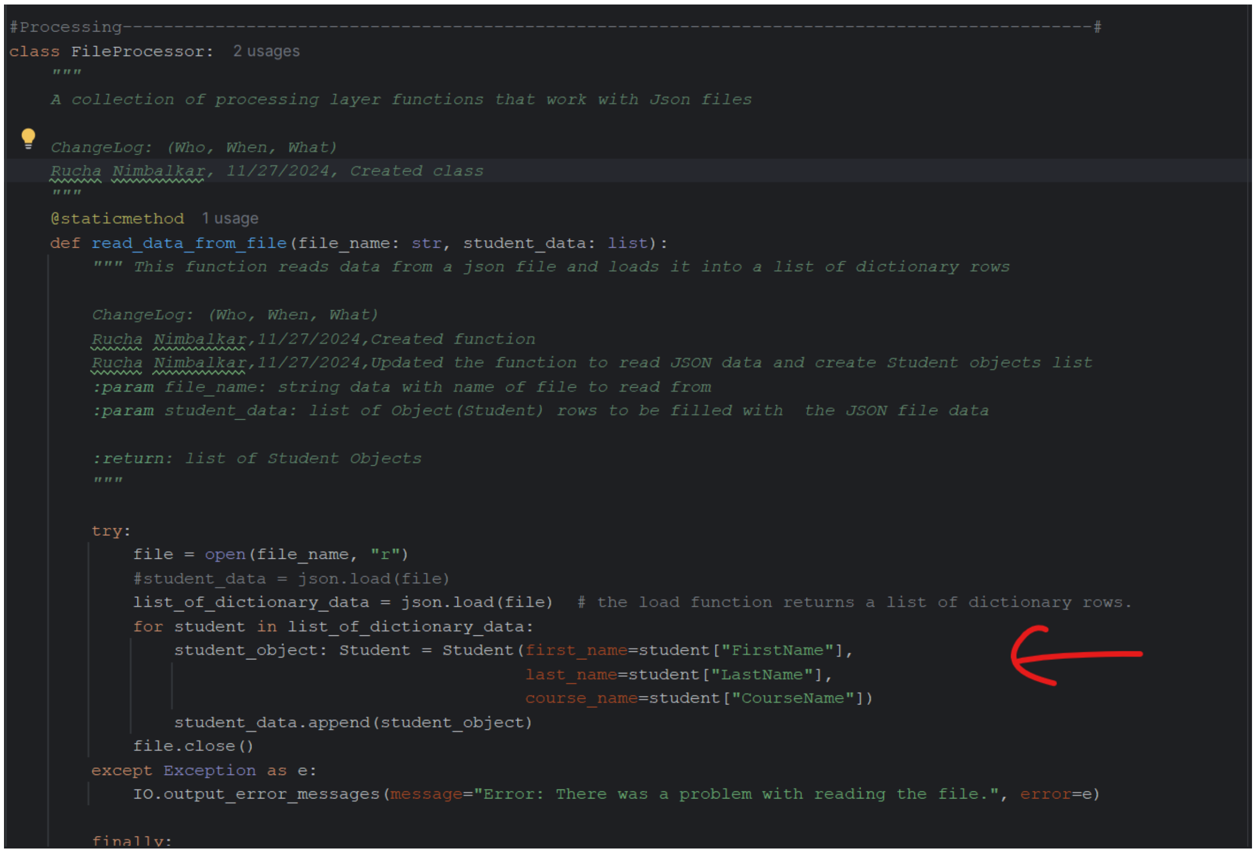
Figure 7 *Assignment 07 Class Student with constructor method, property methods and overriding methods*



```
#Create a Student class the inherits from the Person class (Done)
class Student(Person):
    '''
        A collection data about students

        ChangeLog: (Who, When, What)
        Rucha Nimbalkar, 11/26/2024,Created Class

    '''
    # Add first_name, last_name and course_name properties to the constructor
    def __init__(self, first_name:str = " ", last_name : str = " ", course_name : str = " "): #parameters default to empty
        #Call to the Person constructor and pass it the first_name and last_name data
        super().__init__(first_name=first_name, last_name=last_name)
        self.course_name = course_name # Add an assignment to the course_name property using the course_name parameter

    # Add the getter for course_name (Done)
    @property  3 usages (1 dynamic)
    def course_name(self):
        return self.__course_name.capitalize()

    # Add the setter for course_name (Done)
    @course_name.setter  3 usages (1 dynamic)
    def course_name(self, value:str):
        if value.isalpha() or value == " ": #course_name value is a character or empty string
            self.__course_name = value
        else:
            raise  ValueError("The course name should not contain numbers") #Assuming Course name does not contain numbers

    # Override the __str__() method to return the Student data
    def __str__(self):
         return f'{self.first_name},{self.last_name},{self.course_name}'
```

Since the class FileProcessor was already provided, it just needed to be adjusted to have rows of objects (Students) in the list (student_data) when we read from the file (Enrollments.json) in the read_data_from_file() class FileProcessor method. It was similar to the activities done on Lab 3 of this module. I added the for loop which created a Student object using the data read from the JSON file and appended the object in the student_data list at each iteration as shown in figure 8.

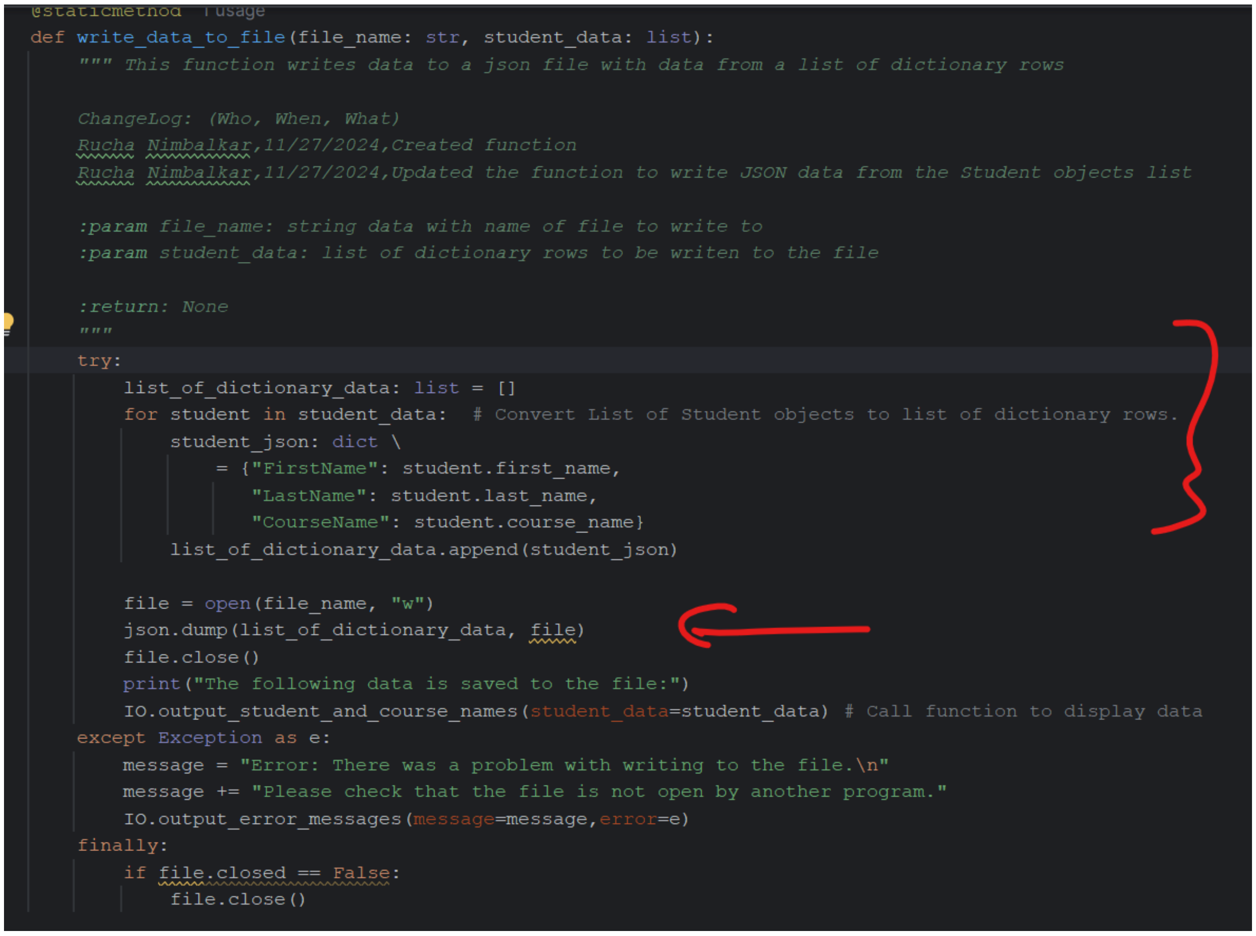
Figure 8 *Assignment 07 Class FileProcessor method read_data_from_file() updated to read from JSON format and store in a list of Student Objects*



```
#Processing-----------------------------------------------------------------------------------------------#
class FileProcessor:  2 usages
    """
    A collection of processing layer functions that work with Json files

    ChangeLog: (Who, When, What)
    Rucha Nimbalkar, 11/27/2024, Created class
    """
    @staticmethod  1 usage
    def read_data_from_file(file_name: str, student_data: list):
        """ This function reads data from a json file and loads it into a list of dictionary rows

        ChangeLog: (Who, When, What)
        Rucha Nimbalkar,11/27/2024,Created function
        Rucha Nimbalkar,11/27/2024,Updated the function to read JSON data and create Student objects list
        :param file_name: string data with name of file to read from
        :param student_data: list of Object(Student) rows to be filled with  the JSON file data

        :return: list of Student Objects
        """

        try:
            file = open(file_name, "r")
            #student_data = json.load(file)
            list_of_dictionary_data = json.load(file)  # the load function returns a list of dictionary rows.
            for student in list_of_dictionary_data:
                student_object: Student = Student(first_name=student["FirstName"],
                                                  last_name=student["LastName"],
                                                  course_name=student["CourseName"])
                student_data.append(student_object)
            file.close()
        except Exception as e:
            IO.output_error_messages(message="Error: There was a problem with reading the file.", error=e)

        finally:
```

The next thing I worked on was updating the method write_data_to_file() in the class FileProcessor. When we write in the JSON file, we are reading the data from the list of objects (Students) and writing it in dictionary format as shown in figure 9.

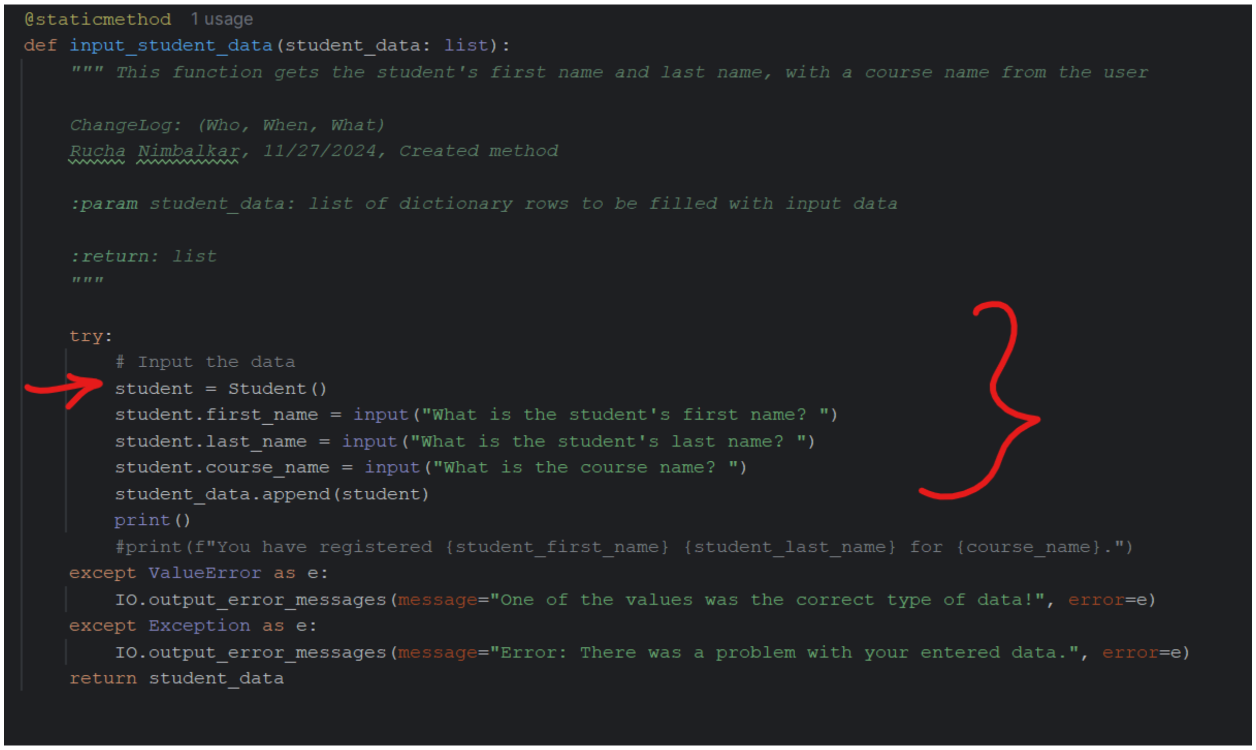
Figure 9 *Assignment 07 Class FileProcessor method write_data_to_file() updated to write in dictionary format in the JSON file*



```
@staticmethod  1 usage
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)
    Rucha Nimbalkar,11/27/2024,Created function
    Rucha Nimbalkar,11/27/2024,Updated the function to write JSON data from the Student objects list

    :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.course_name}
            list_of_dictionary_data.append(student_json)

        file = open(file_name, "w")
        json.dump(list_of_dictionary_data, file)
        file.close()
        print("The following data is saved to the file:")
        IO.output_student_and_course_names(student_data=student_data) # Call function to display data
    except Exception as e:
        message = "Error: There was a problem with writing to the file.\n"
        message += "Please check that the file is not open by another program."
        IO.output_error_messages(message=message,error=e)
    finally:
        if file.closed == False:
            file.close()
```

After working on the data and processing sections of this program, I moved on to the presentation layer. Class IO is the presentation section. I realized that when accepting student data, it will be stored and validated in the Student object form. Hence, I updated the input_student_data() method in the IO class as shown in figure 10.

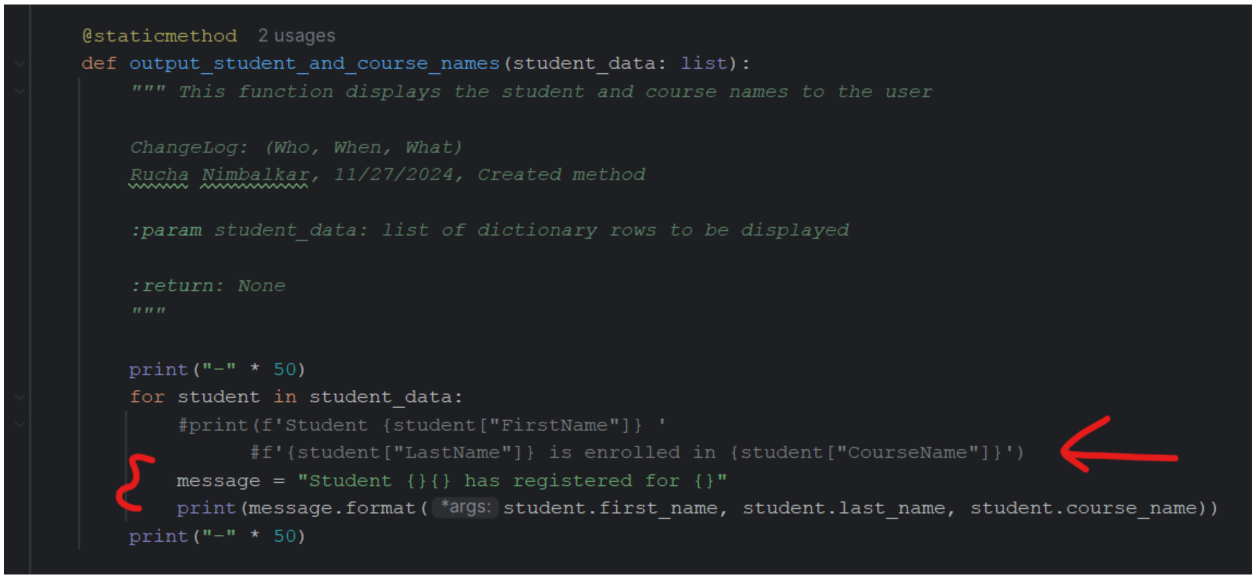
Figure 10 *Assignment 07 Class IO method input_student_data() updated to accept data and validate using class Student's setter property and constructor*



```
@staticmethod  1 usage
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)
    Rucha Nimbalkar, 11/27/2024, Created method

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

    :return: list
    """

    try:
        # Input the data
        student = Student()
        student.first_name = input("What is the student's first name? ")
        student.last_name = input("What is the student's last name? ")
        student.course_name = input("What is the course name? ")
        student_data.append(student)
        print()
        #print(f"You have registered {student_first_name} {student_last_name} for {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)
    return student_data
```

When showing the current data, we will be accessing the list of Student objects student_data. Therefore, I updated the method output_student_and_course_names() in the IO class.

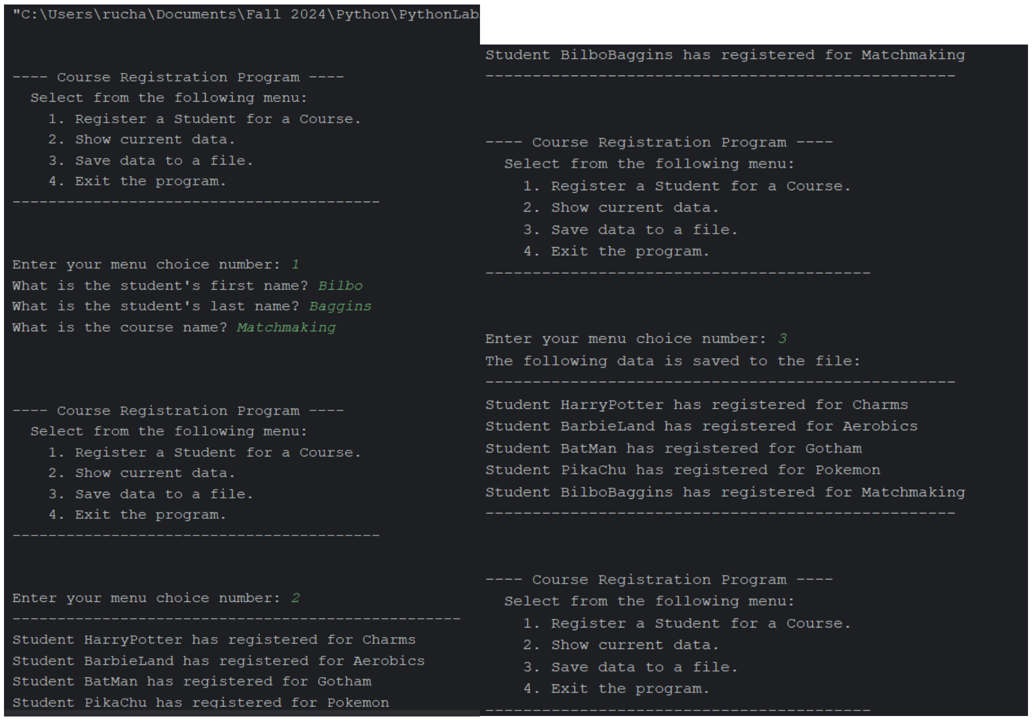
Figure 11 *Assignment 07 Class IO method output_student_and_course_names() updated to display student and course name from the Student Objects list*



```
@staticmethod  2 usages
def output_student_and_course_names(student_data: list):
    """ This function displays the student and course names to the user

    ChangeLog: (Who, When, What)
    Rucha Nimbalkar, 11/27/2024, Created method

    :param student_data: list of dictionary rows to be displayed

    :return: None
    """

    print("-" * 50)
    for student in student_data:
        #print(f'Student {student["FirstName"]} '
              #f'{student["LastName"]} is enrolled in {student["CourseName"]}')
        message = "Student {}{} has registered for {}"
        print(message.format( *args: student.first_name, student.last_name, student.course_name))
    print("-" * 50)
```

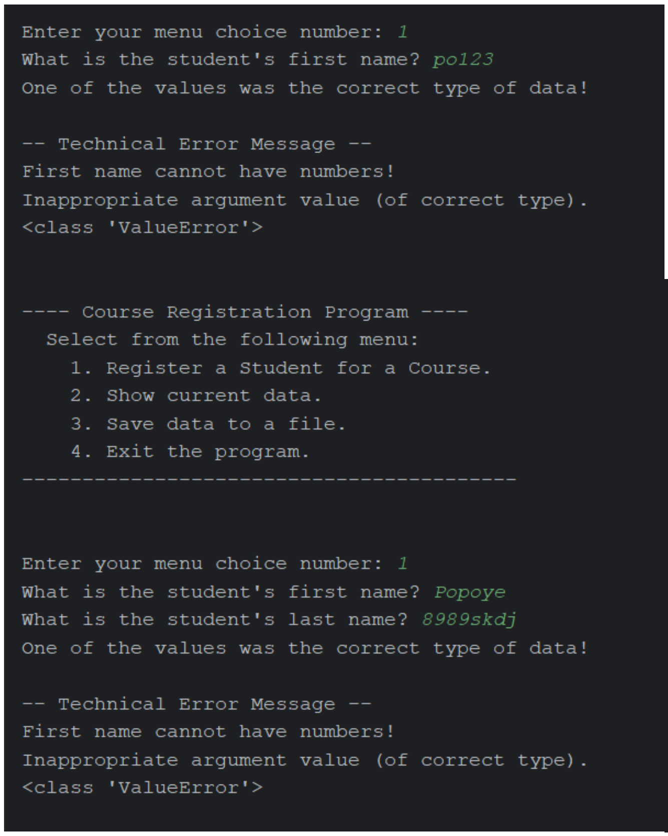
I ran the code, and I got the output shown in figure 12.

Figure 12 *Assignment 07 Output*

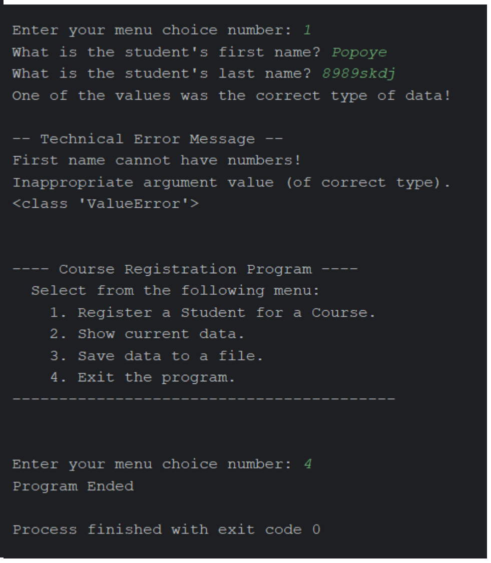


```
"C:\Users\rucha\Documents\Fall 2024\Python\PythonLab


---- 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
What is the student's first name? Bilbo
What is the student's last name? Baggins
What is the course name? Matchmaking


---- 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 HarryPotter has registered for Charms
Student BarbieLand has registered for Aerobics
Student BatMan has registered for Gotham
Student PikaChu has registered for Pokemon
```

```
Student BilboBaggins has registered for Matchmaking
--------------------------------------------------


---- 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
The following data is saved to the file:
--------------------------------------------------
Student HarryPotter has registered for Charms
Student BarbieLand has registered for Aerobics
Student BatMan has registered for Gotham
Student PikaChu has registered for Pokemon
Student BilboBaggins has registered for Matchmaking
--------------------------------------------------


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



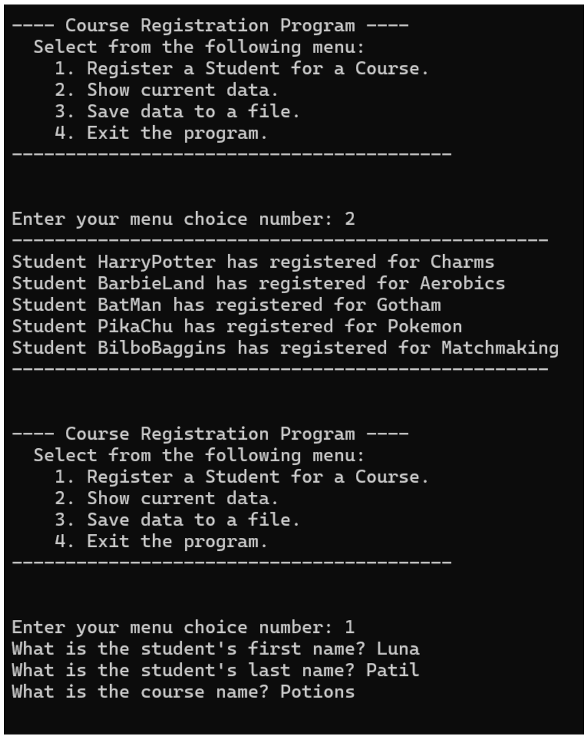
```
Enter your menu choice number: 1
What is the student's first name? po123
One of the values was the correct type of data!

-- Technical Error Message --
First name cannot have 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
What is the student's first name? Popoye
What is the student's last name? 8989skdj
One of the values was the correct type of data!

-- Technical Error Message --
First name cannot have numbers!
Inappropriate argument value (of correct type).
<class 'ValueError'>
```



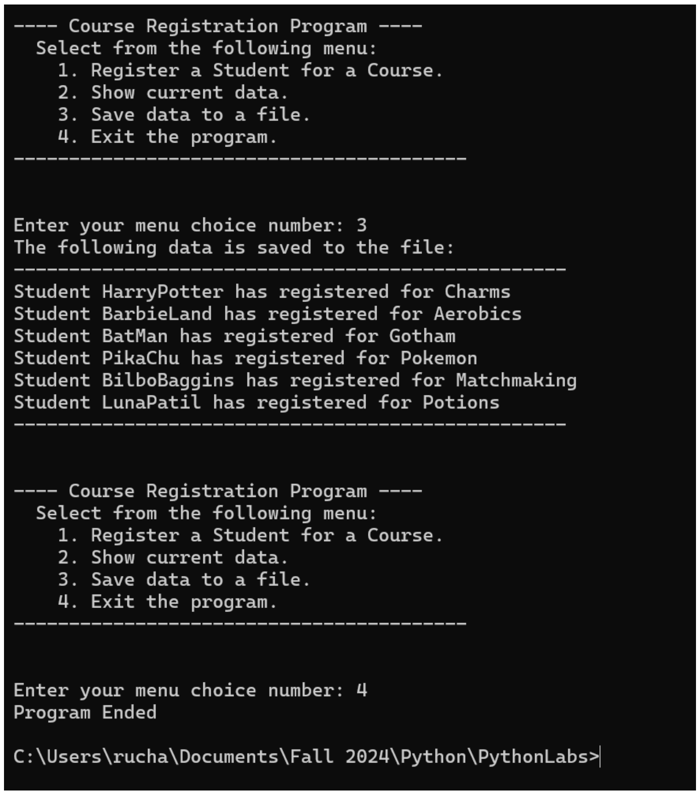
```
Enter your menu choice number: 1
What is the student's first name? Popoye
What is the student's last name? 8989skdj
One of the values was the correct type of data!

-- Technical Error Message --
First name cannot have 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: 4
Program Ended

Process finished with exit code 0
```

I ran the code in command terminal and got the output shown in figure 13. After looking at both the outputs I noticed that there was no space between the first same and last name, so I went back to my code and added some space in the message variable in the output_student_and_course_names() method. After making that change I ran the code again and I got the desired output (figure 14).

Figure 13 *Assignment 07 Output in command terminal*

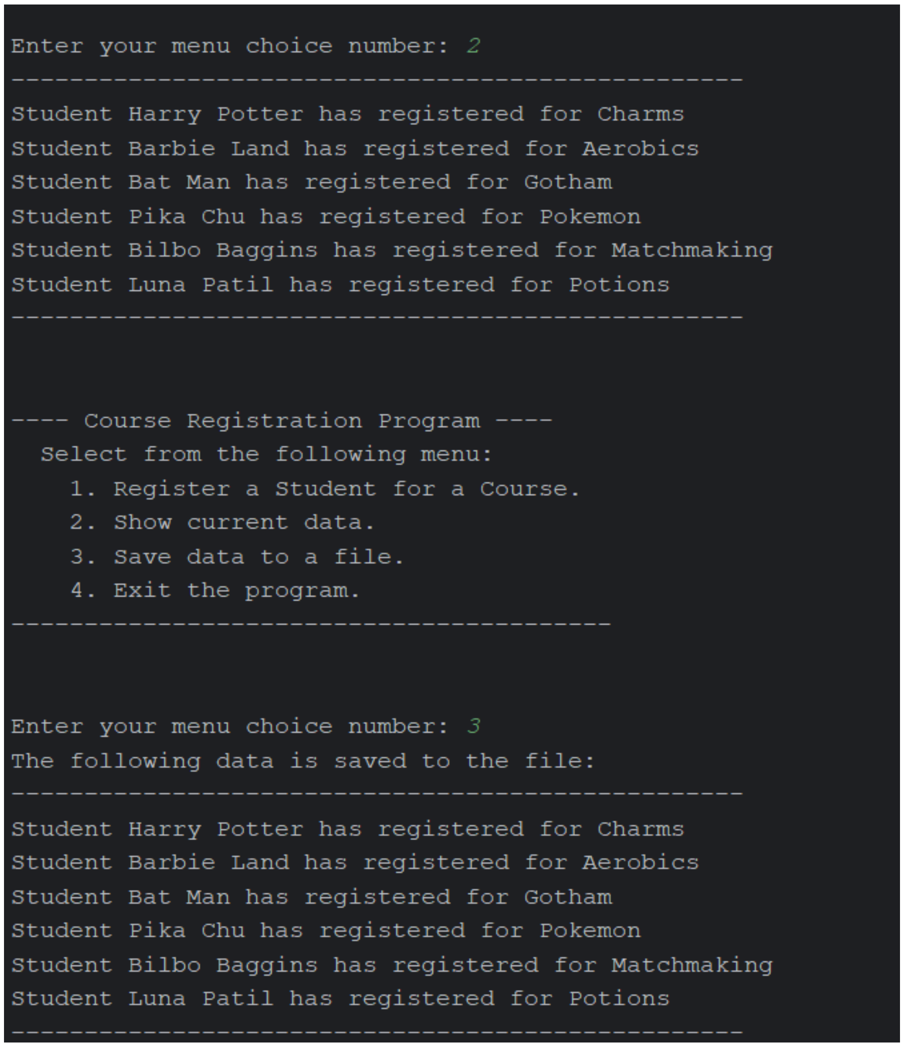


```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


Enter your menu choice number: 2
--------------------------------------------------
Student HarryPotter has registered for Charms
Student BarbieLand has registered for Aerobics
Student BatMan has registered for Gotham
Student PikaChu has registered for Pokemon
Student BilboBaggins has registered for Matchmaking
--------------------------------------------------


---- 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
What is the student's first name? Luna
What is the student's last name? Patil
What is the course name? Potions
```



```
---- 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
The following data is saved to the file:
--------------------------------------------------
Student HarryPotter has registered for Charms
Student BarbieLand has registered for Aerobics
Student BatMan has registered for Gotham
Student PikaChu has registered for Pokemon
Student BilboBaggins has registered for Matchmaking
Student LunaPatil has registered for Potions
--------------------------------------------------


---- 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\rucha\Documents\Fall 2024\Python\PythonLabs>
```

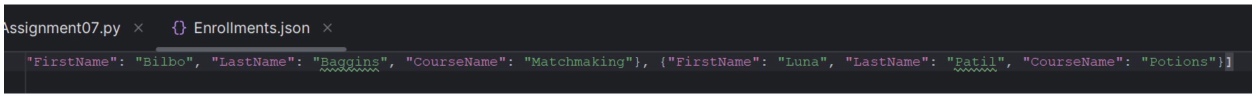
Figure 14 *Assignment 07 Output after adding some space in the message variable in the output_student_and_course_names() method*



```
Enter your menu choice number: 2
--------------------------------------------------
Student Harry Potter has registered for Charms
Student Barbie Land has registered for Aerobics
Student Bat Man has registered for Gotham
Student Pika Chu has registered for Pokemon
Student Bilbo Baggins has registered for Matchmaking
Student Luna Patil has registered for Potions
--------------------------------------------------



---- 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
The following data is saved to the file:
--------------------------------------------------
Student Harry Potter has registered for Charms
Student Barbie Land has registered for Aerobics
Student Bat Man has registered for Gotham
Student Pika Chu has registered for Pokemon
Student Bilbo Baggins has registered for Matchmaking
Student Luna Patil has registered for Potions
--------------------------------------------------
```

To verify whether data was saved to the file, I opened the file and confirmed it as shown in figure 15.

Figure 14 *Assignment 07 Enrollments.json file*



```
"FirstName": "Bilbo", "LastName": "Baggins", "CourseName": "Matchmaking"}, {"FirstName": "Luna", "LastName": "Patil", "CourseName": "Potions"}]
```

Conclusion

This assignment helped me understand that classes are a better way to encapsulate data, and they help in isolating the data from being manipulated without authorization. I also learnt that I could create a list of Objects of a certain type, and I can iterate the list using a for loop.

References

Randall, R.(n.d.). *IT Fundamentals 110 A* [MOOC]. University of Washington. Foundations of Programming (Python) - UW Professional & Continuing Education