Paulo Roberto Barreto Alves

08/27/2025

Foundations of Python Programming

Assignment 07

https://github.com/pagh59258/IntroToProg-Python-Mod07

Classes and Objects

Introduction

On module 07, I acquired knowledge on some Python programming tools and techniques. With this new knowledge, I created a Python script very similar to the one I have developed in Module 06, but adding some extra techniques, such as: **the use of classes and objects.**

Reading Python 100 Module 07 notes.doc file

- Statements, Functions and Classes
  - Objects vs. Classes
- Data Classes vs. Processing Classes
- Data Classes Components
  - Attributes
  - Constructors
  - The Self Keyword
- Adding Data Validation
  - Private Attributes
  - Properties
  - Abstractions and Encapsulation
- Inherited Code
  - Python's Magic Methods
  - Overriding Methods
  - The Advantages of Inheritance
- Git vs GitHub
  - Git
  - GitHub Desktop

Performing Module 07 labs

Throughout the module, I was asked to apply the learned knowledge into some practice labs, such as:

- Mod07-Lab01-Working with Constructors
- Mod07-Lab02-Working with Class Properties
- Mod07-Lab03-Working with Inheritance

Creating a Python script

I created a new file named Assignment07.py, which as the extension indicates, is a Python script. This Python script demonstrates using constants, variables, and print statements to display a message about a student's registration for a course. This program is very similar to Assignment06, but it adds the use of set of data classes.

Acceptance Criteria

Your program must include the following features and code to be accepted as complete:

File Name:

- The file is named Assignment07.py

Script Header:

- The script header includes this text and has been updated with your name and the current date.

Constants:

- The constant **MENU: str** is set to the value:

```
---- 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
-----------------------------------------
```

- The constant **FILE_NAME: str** is set to the value "Enrollments.json"
- Constant values do not change throughout the program.

Variables:

- **menu_choice: str** is set to empty string.
- **students: list** : list is set to and empty list

Classes:

- The program includes a class named FileProcessor.
- The program includes a class named IO.

- The program includes a class named Person.
- The program includes a class named Student.
- All classes include descriptive document strings.

Class Properties:

- The program includes properties for **student_first_name: str** and defaults to an empty string.
- The program includes properties for **student_last_name: str** and defaults to an empty string.
- The program includes properties for **course_name: str** and defaults to an empty string.
- The program's properties must include simple validation code.

Class Methods:

- The program includes a method to extract comma separately data from each data class.

Functions:

- All functions include descriptive document strings.
- All functions the include exception blocks use the output_error_messages() function for handling error messages.
- All non-instance functions use the @staticmethod decorator (The ones in the FileProcessor and IO classes.)
- The program includes functions with the following names and parameters:
    - output_error_messages(message: str, error: Exception = None)
    - output_menu(menu: str)
    - input_menu_choice()
    - output_student_courses(student_data: list)
    - input_student_data(student_data: list)
    - read_data_from_file(file_name: str, student_data: list):
    - write_data_to_file(file_name: str, student_data: list):

Input / Output:

- On menu choice 1, the program prompts the user to enter the student's first name and last name, followed by the course name, using the input() function and stores the inputs in the respective variables.
- Data collected for menu choice 1 is added to the **students** two-dimensional list of Student objects.
- On menu choice 2, the program uses the print() function to show a string of comma-separated values for each row collected in the **students** variable.

Processing

- When the program starts, the contents of the "Enrollments.json" are automatically read into a two-dimensional list of dictionaries rows using the json.load() function. Next, it converts that data into a list of Student object rows. (**Tip:** Make sure to put some starting data into the file or you will get an error!)
- On menu choice 3, the program opens a file named "Enrollments.json" in write mode using the open() function. Next, it converts the data in the **students** variable (a list of Student object rows) into a list of dictionary rows, then writes the list of dictionary data into the file using the json.dump() function. Finally, it closes the file using the close() method.
- On menu choice 4, the program ends.

Error Handling

- The program provides structured error handling when the file is read into the list of dictionary rows.
- The program provides structured error handling when the user enters a first name.
- The program provides structured error handling when the user enters a last name.
- The program provides structured error handling when the dictionary rows are written to the file.

Test:

- The program takes the user's input for a student's first, last name, and course name.
- The program displays the user's input for a student's first, last name, and course name.
- The program saves the user's input for a student's first, last name, and course name to a JSON file. (check this in PyCharm or a simple text editor like Notepad or TextEdit.)
- The program allows users to enter multiple registrations (first name, last name, course name).
- The program allows users to display multiple registrations (first name, last name, course name).
- The program allows users to save multiple registrations to a file (first name, last name, course name).
- The program runs correctly in both **PyCharm and** from the **console or terminal**.

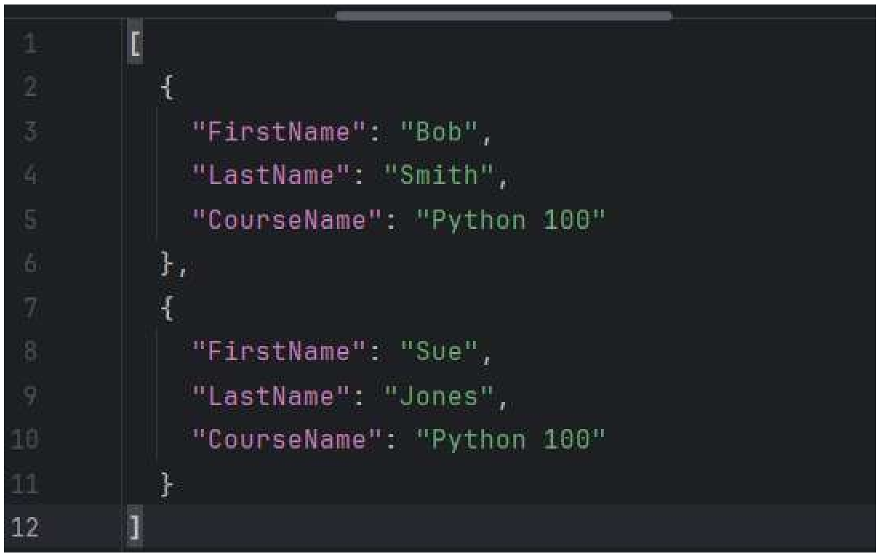
Source Control:

- The script file and the knowledge document are hosted on a GitHub repository.
- A link to the repository is included in the knowledge document.
- A link to the repository is included in the GitHub links forum.

Here are some notes about this Python script:

- To avoid errors, we need to have some data already populated on the "Enrollments.json" file before starting script execution.
- A "While" loop is started
    - Function "IO.input_menu_choice" is invoked to display a menu with 4 options
        - Only accept options are 1, 2, 3 or 4. If any other option is entered, an error is shown and menu is displayed again;
        - Regarding Menu Option "1":
            - "IO.input_student_data" function is invoked
                - Function does a critique to not allow invalid fields:
                    - Student's first name cannot be numeric or blank;
                    - Students' last name cannot be numeric or blank;
                    - Course name cannot be blank.
            - Users are allowed to make several enrollments;
- Regarding Menu Option "2":
    - Function "IO.output_current_student_data" is invoked
        - It displays current student registrations (including the ones not yet saved to the json file)
- Regarding Menu Option "3":
    - Function "FileProcessor.write_data_to_file" is invoked
        - Writes current student registrations to json file
        - Display list of student registrations saved
- Regarding Menu Option "4":
    - Function "IO.output_check_unsaved_student_data" is invoked
        - It checks if there is unsaved data and offer the user the option to save it or not;
        - Program is terminated

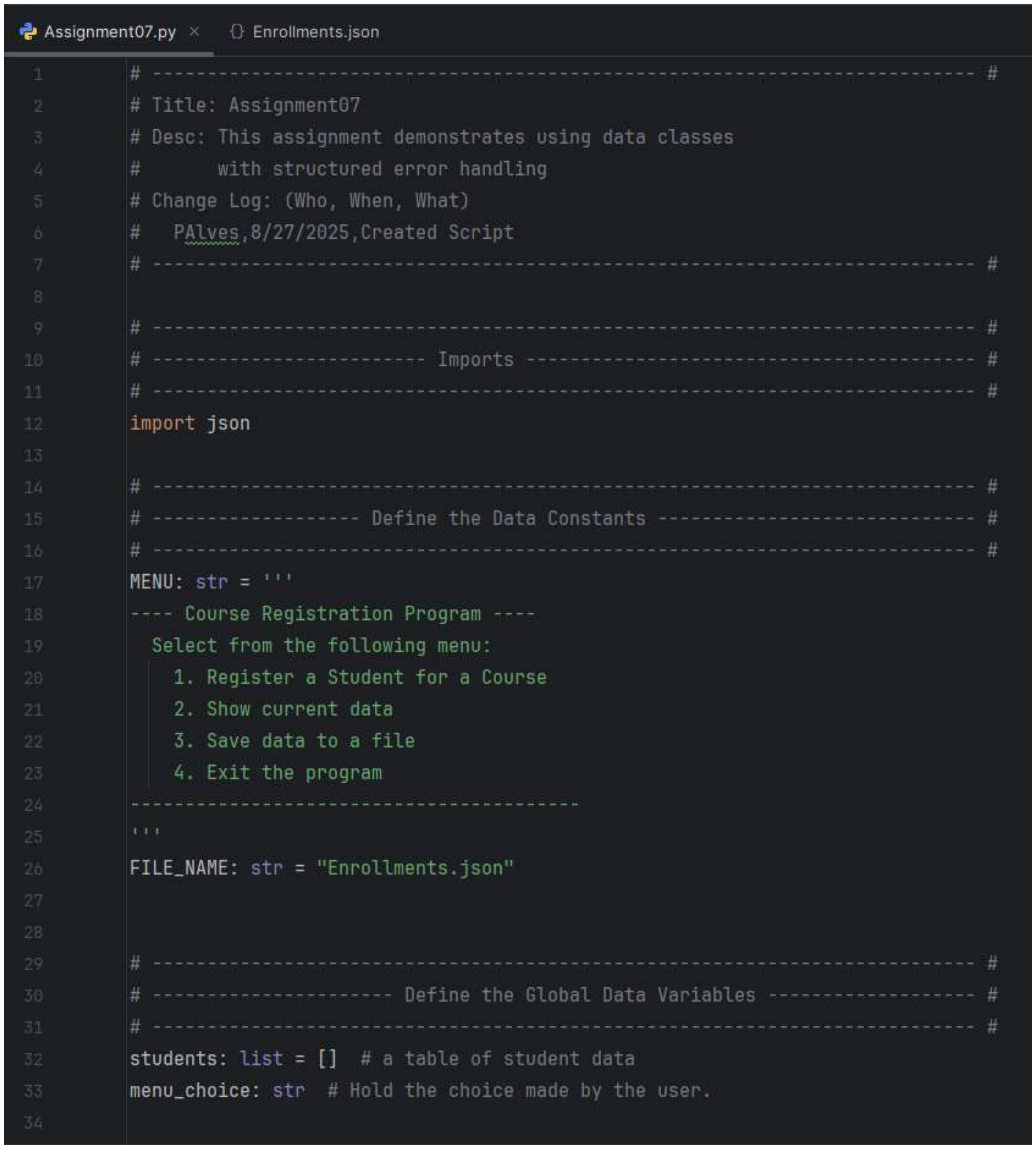
On Figure 1 below, we can see the initial Enrollment.json file (provided inside assignment zip file), must contain some data:



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

Figure 1: Enrollment.json initial file

On Figure 2 below, we can see the basic coding used for Script header, Import and constants & variables definition:

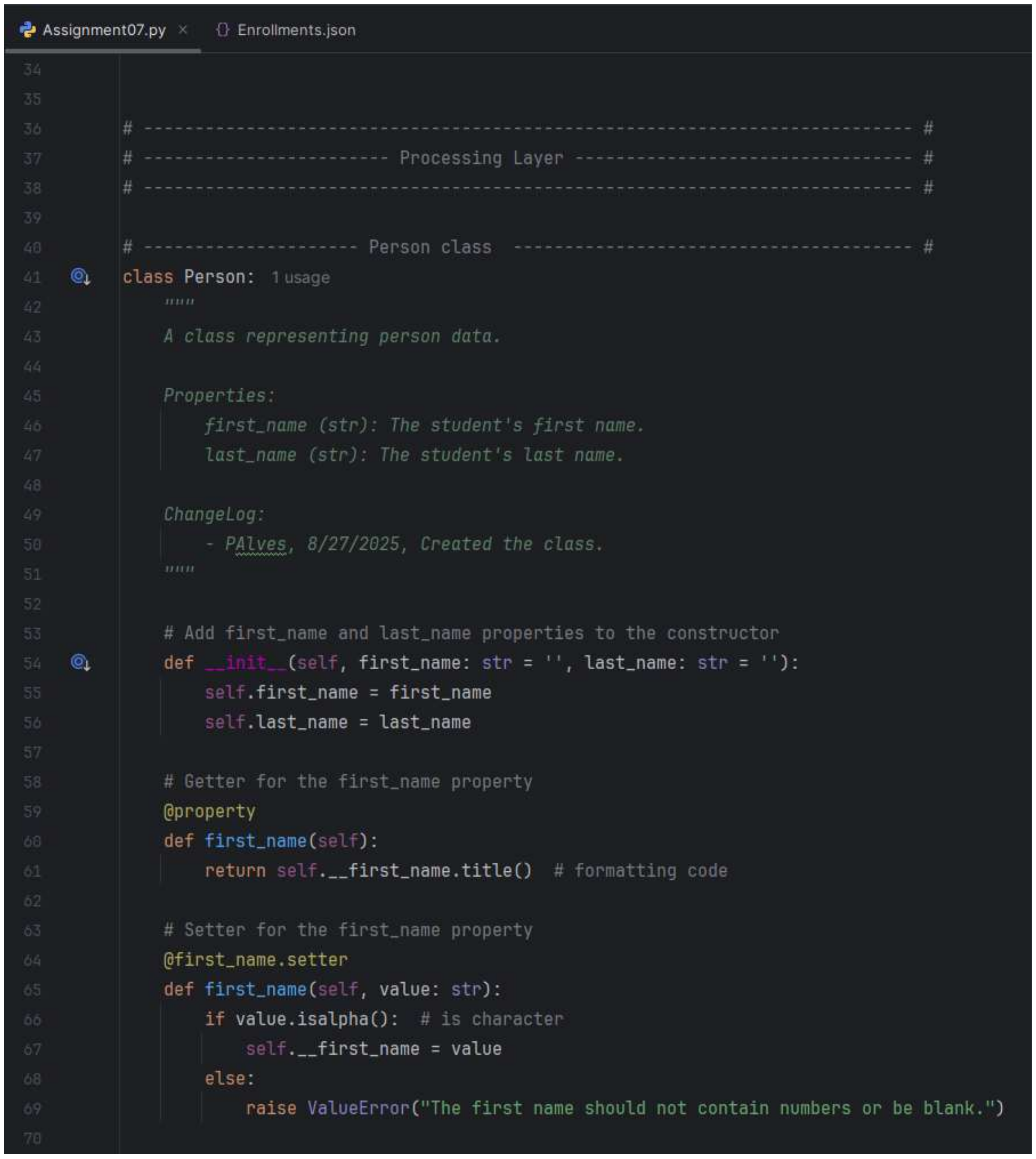


Assignment07.py Enrollments.json

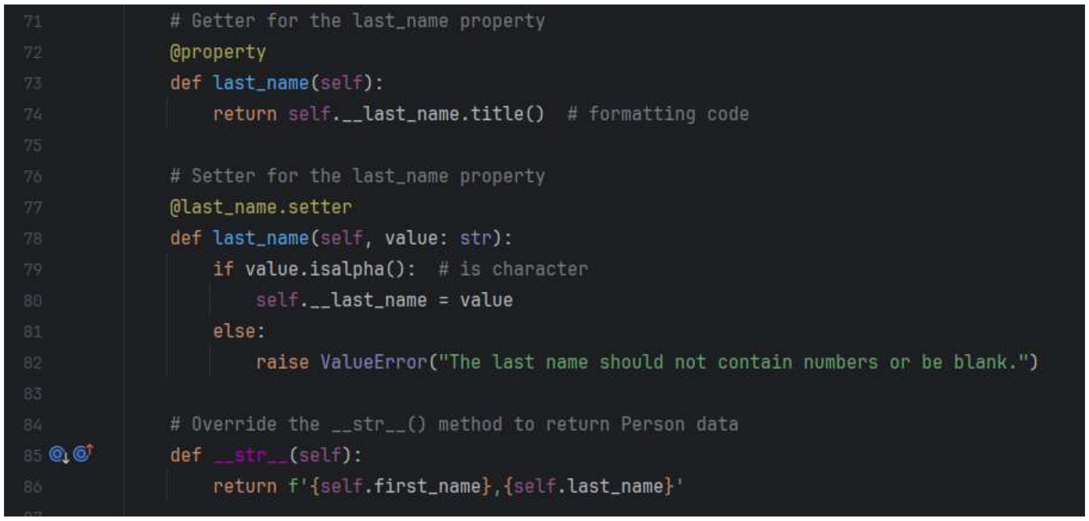
```python
# ------------------------------------------------------------------------ #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
#       with structured error handling
# Change Log: (Who, When, What)
#   PAlves,8/27/2025,Created Script
# ------------------------------------------------------------------------ #

# ------------------------------------------------------------------------ #
# ------------------------- Imports -------------------------------------- #
# ------------------------------------------------------------------------ #
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 Global Data Variables ----------------- #
# ------------------------------------------------------------------------ #
students: list = []  # a table of student data
menu_choice: str  # Hold the choice made by the user.

```

Figure 2: Assignment07.py Python script header, imports, constants and variables

On Figure 3 below, we can see Processing Layer – Person class:



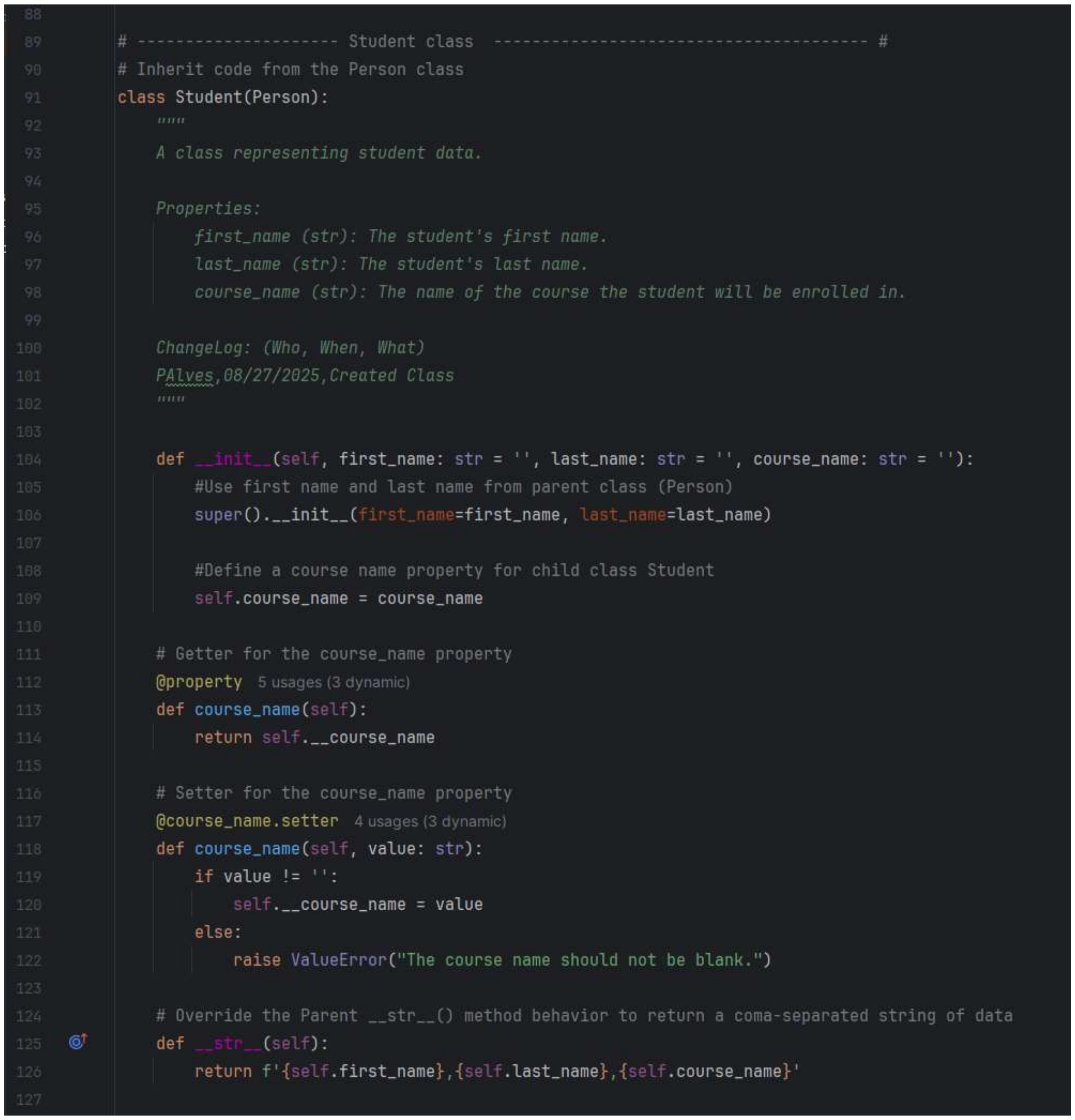
```python


# -------------------------------------------------------------------------- #
# ----------------------- Processing Layer ------------------------------- #
# -------------------------------------------------------------------------- #

# -------------------- Person class  ------------------------------------- #
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:
        - PAlves, 8/27/2025, Created the class.
    """

    # Add first_name and last_name properties to the constructor
    def __init__(self, first_name: str = '', last_name: str = ''):
        self.first_name = first_name
        self.last_name = last_name

    # Getter for the first_name property
    @property
    def first_name(self):
        return self.__first_name.title()  # formatting code

    # Setter for the first_name property
    @first_name.setter
    def first_name(self, value: str):
        if value.isalpha():  # is character
            self.__first_name = value
        else:
            raise ValueError("The first name should not contain numbers or be blank.")

```



```
    # Getter for the last_name property
    @property
    def last_name(self):
        return self.__last_name.title()  # formatting code

    # Setter for the last_name property
    @last_name.setter
    def last_name(self, value: str):
        if value.isalpha():  # is character
            self.__last_name = value
        else:
            raise ValueError("The last name should not contain numbers or be blank.")

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

Figure 3: PyCharm – Assignment07.py Python Person Class

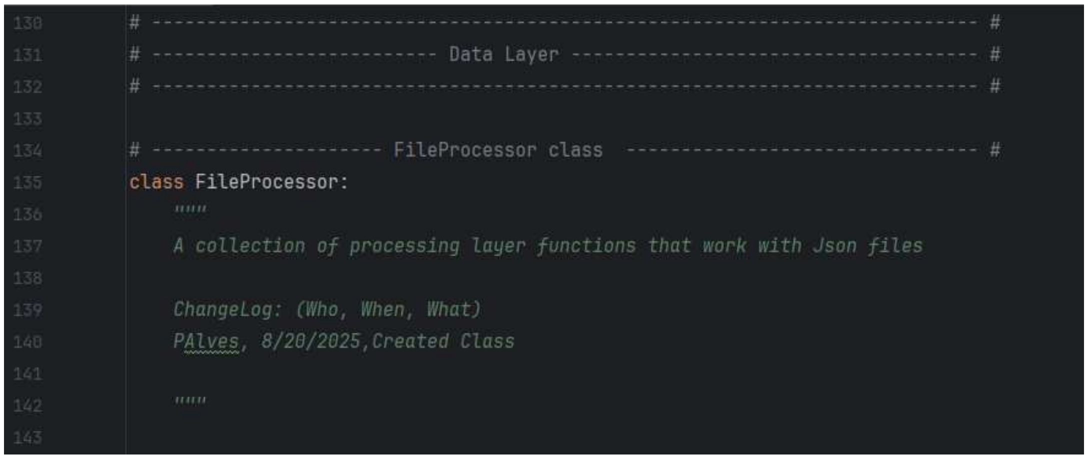
On Figure 4 below, we can see Processing Layer – Student class:



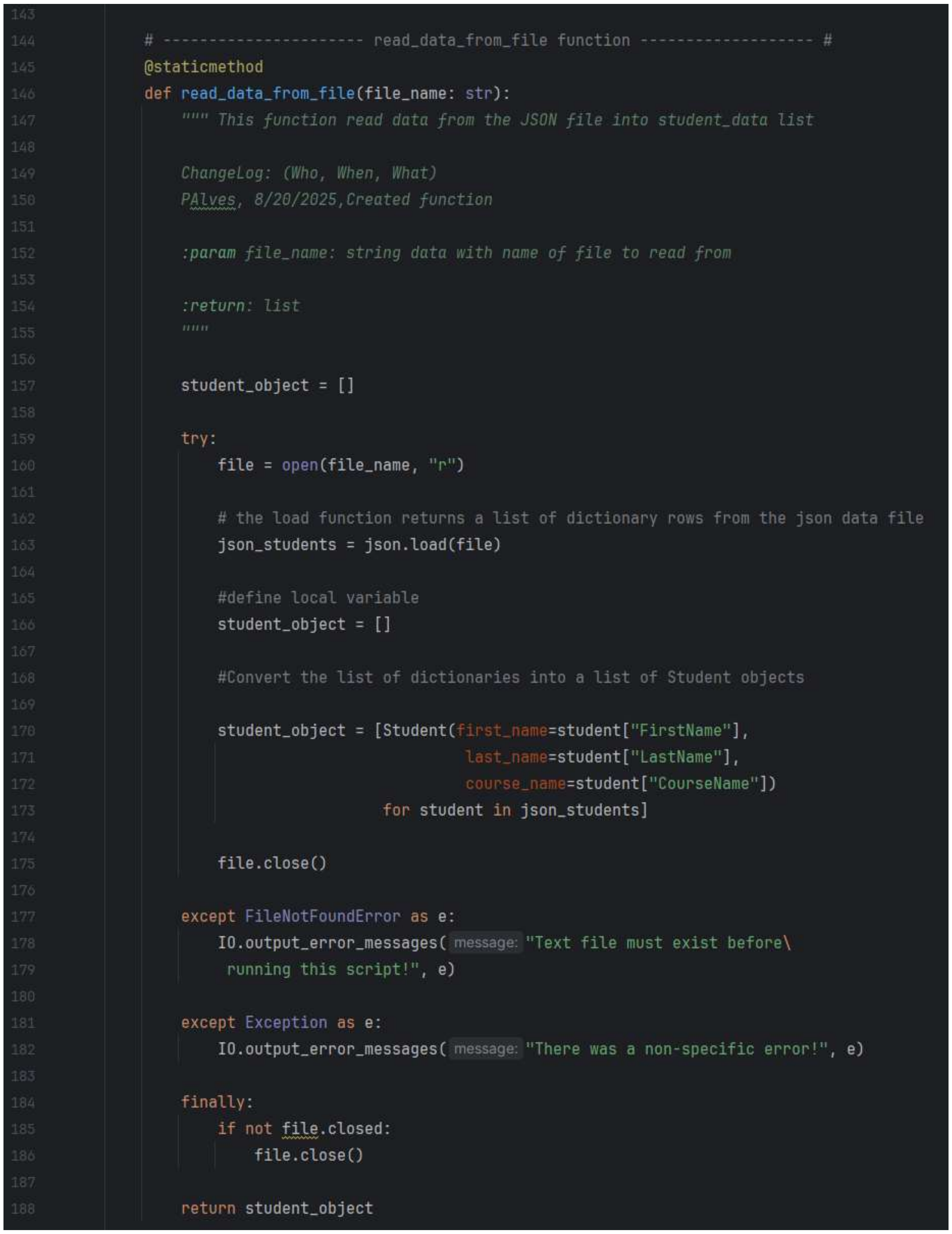
```python

# --------------------- Student class  --------------------------------------- #
# Inherit code from the Person class
class Student(Person):
    """
    A class representing student data.

    Properties:
        first_name (str): The student's first name.
        last_name (str): The student's last name.
        course_name (str): The name of the course the student will be enrolled in.

    ChangeLog: (Who, When, What)
    PAlves,08/27/2025,Created Class
    """

    def __init__(self, first_name: str = '', last_name: str = '', course_name: str = ''):
        #Use first name and last name from parent class (Person)
        super().__init__(first_name=first_name, last_name=last_name)

        #Define a course name property for child class Student
        self.course_name = course_name

    # Getter for the course_name property
    @property  5 usages (3 dynamic)
    def course_name(self):
        return self.__course_name

    # Setter for the course_name property
    @course_name.setter  4 usages (3 dynamic)
    def course_name(self, value: str):
        if value != '':
            self.__course_name = value
        else:
            raise ValueError("The course name should not be blank.")

    # Override the Parent __str__() method behavior to return a coma-separated string of data
    def __str__(self):
        return f'{self.first_name},{self.last_name},{self.course_name}'

```

Figure 4: PyCharm – Assignment07.py Python Student Class

On Figure 5 below, we can see the Data Layer – FileProcessor class – read_data_from_file function:



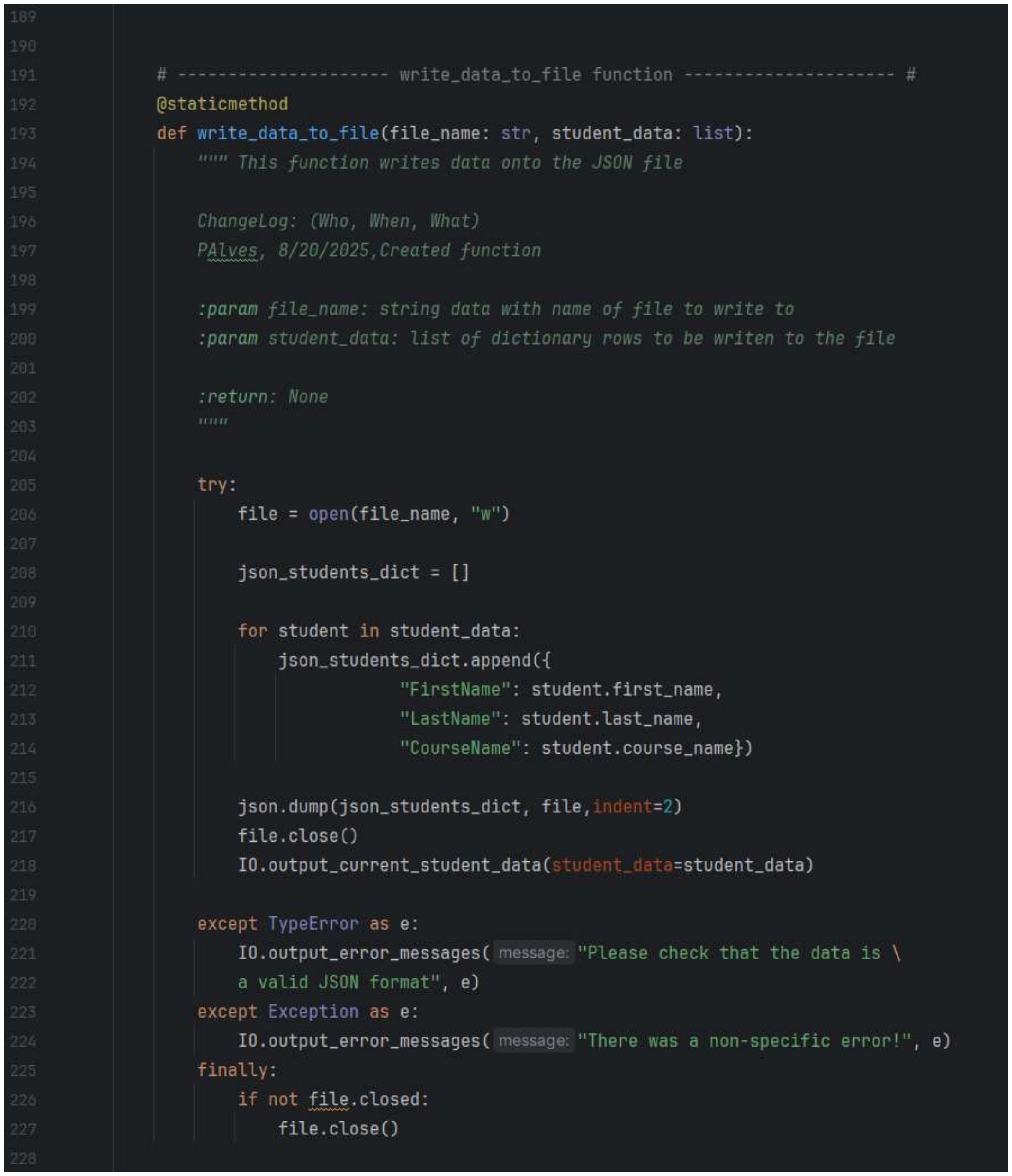
```
# ----------------------------------------------------------------------- #
# ------------------------- Data Layer ----------------------------------- #
# ----------------------------------------------------------------------- #

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

    ChangeLog: (Who, When, What)
    PAlves, 8/20/2025,Created Class

    """

```



```
    # --------------------- read_data_from_file function ------------------- #
    @staticmethod
    def read_data_from_file(file_name: str):
        """ This function read data from the JSON file into student_data list

        ChangeLog: (Who, When, What)
        PAlves, 8/20/2025,Created function

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

        :return: list
        """

        student_object = []

        try:
            file = open(file_name, "r")

            # the load function returns a list of dictionary rows from the json data file
            json_students = json.load(file)

            #define local variable
            student_object = []

            #Convert the list of dictionaries into a list of Student objects

            student_object = [Student(first_name=student["FirstName"],
                                      last_name=student["LastName"],
                                      course_name=student["CourseName"])
                                  for student in json_students]

            file.close()

        except FileNotFoundError as e:
            IO.output_error_messages( message: "Text file must exist before\
             running this script!", e)

        except Exception as e:
            IO.output_error_messages( message: "There was a non-specific error!", e)

        finally:
            if not file.closed:
                file.close()

        return student_object
```

Figure 5: PyCharm – Assignment07.py Python read_data_from_file function

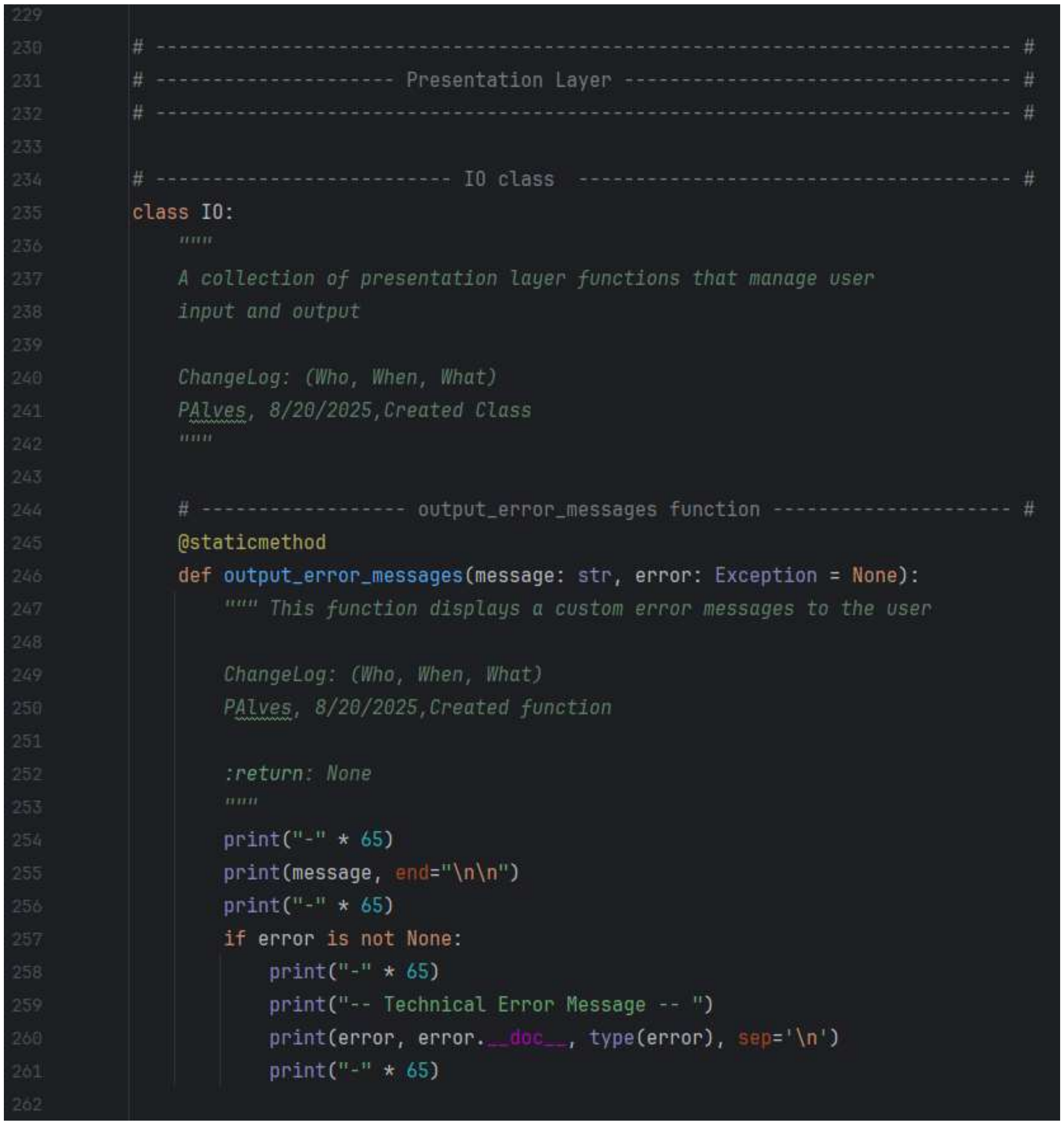
On Figure 6 below, we can see the Data Layer – FileProcessor class – write_data_to_file function:



```python


    # --------------------- write_data_to_file function --------------------- #
    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
        """ This function writes data onto the JSON file

        ChangeLog: (Who, When, What)
        PAlves, 8/20/2025,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:
            file = open(file_name, "w")

            json_students_dict = []

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

            json.dump(json_students_dict, file,indent=2)
            file.close()
            IO.output_current_student_data(student_data=student_data)

        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:
            if not file.closed:
                file.close()

```

Figure 6: PyCharm – Assignment07.py Python write_data_to_file function

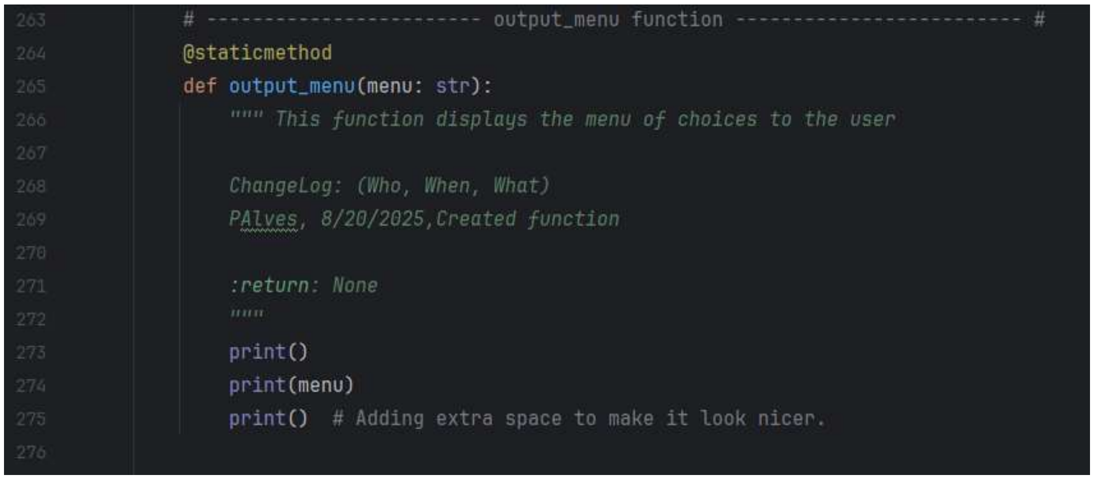
On Figure 7 below, we can see the Presentation Layer – IO class – output_error_messages function:



```python

# ----------------------------------------------------------------------------- #
# --------------------- Presentation Layer --------------------------------- #
# ----------------------------------------------------------------------------- #

# ------------------------- IO class  ------------------------------------- #
class IO:
    """
    A collection of presentation layer functions that manage user
    input and output

    ChangeLog: (Who, When, What)
    PAlves, 8/20/2025,Created Class
    """

    # ------------------ output_error_messages function --------------------- #
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        """ This function displays a custom error messages to the user

        ChangeLog: (Who, When, What)
        PAlves, 8/20/2025,Created function

        :return: None
        """
        print("-" * 65)
        print(message, end="\n\n")
        print("-" * 65)
        if error is not None:
            print("-" * 65)
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')
            print("-" * 65)

```

Figure 7: PyCharm – Assignment07.py Python output_error_messages function

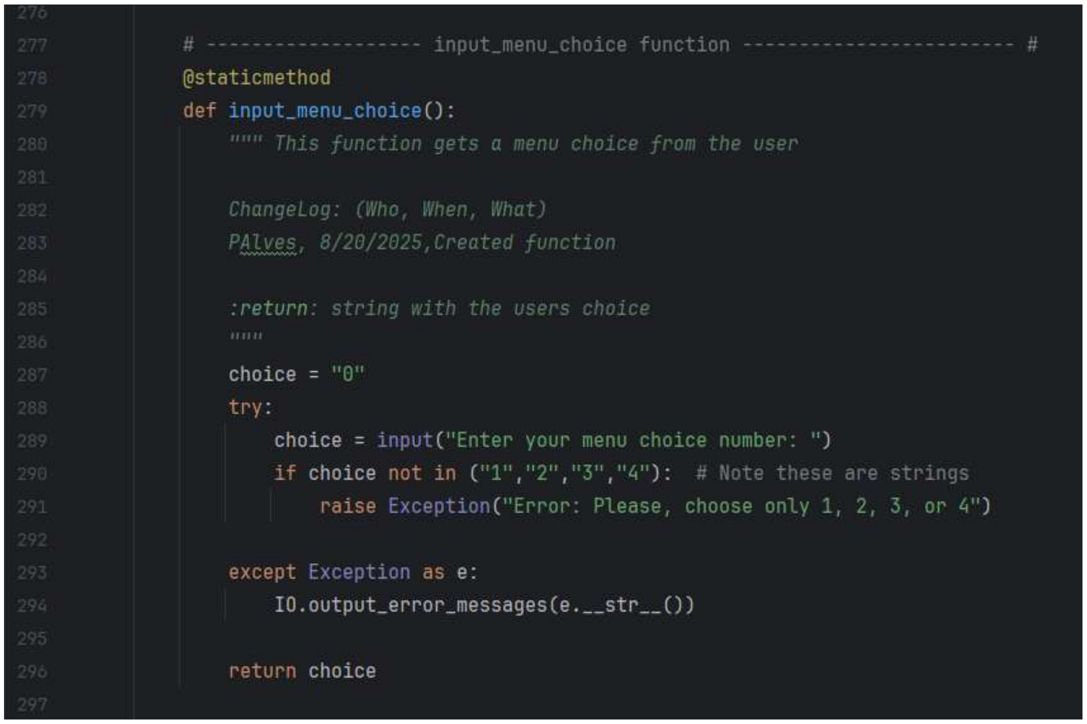
On Figure 8 below, we can see the output_menu function:



```
    # ------------------------ output_menu function ------------------------- #
    @staticmethod
    def output_menu(menu: str):
        """ This function displays the menu of choices to the user

        ChangeLog: (Who, When, What)
        PAlves, 8/20/2025,Created function

        :return: None
        """
        print()
        print(menu)
        print()  # Adding extra space to make it look nicer.

```

Figure 8: PyCharm – Assignment07.py Python output_menu function

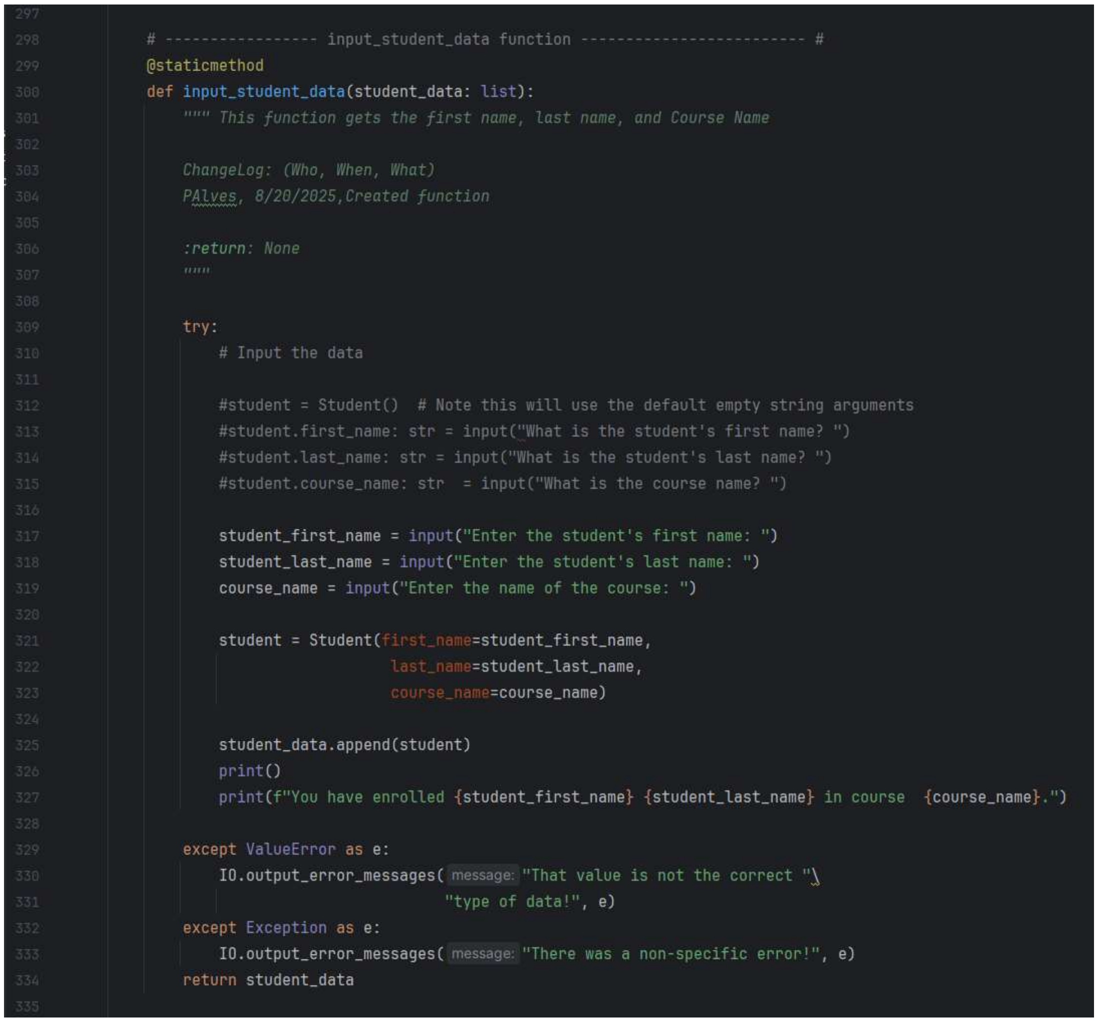
On Figure 9 below, we can see the input_menu_choice function:



```

    # ------------------- input_menu_choice function ------------------------ #
    @staticmethod
    def input_menu_choice():
        """ This function gets a menu choice from the user

        ChangeLog: (Who, When, What)
        PAlves, 8/20/2025,Created function

        :return: string with the users choice
        """
        choice = "0"
        try:
            choice = input("Enter your menu choice number: ")
            if choice not in ("1","2","3","4"):  # Note these are strings
                raise Exception("Error: Please, choose only 1, 2, 3, or 4")

        except Exception as e:
            IO.output_error_messages(e.__str__())

        return choice

```

Figure 9: PyCharm – Assignment07.py Python script input_menu_choice function

On Figure 10 below, we can see the input_student_data function:



```python
    # ----------------- input_student_data function ------------------------- #
    @staticmethod
    def input_student_data(student_data: list):
        """ This function gets the first name, last name, and Course Name

        ChangeLog: (Who, When, What)
        PAlves, 8/20/2025,Created function

        :return: None
        """

        try:
            # Input the data

            #student = Student()  # Note this will use the default empty string arguments
            #student.first_name: str = input("What is the student's first name? ")
            #student.last_name: str = input("What is the student's last name? ")
            #student.course_name: str  = input("What is the course name? ")

            student_first_name = input("Enter the student's first name: ")
            student_last_name = input("Enter the student's last name: ")
            course_name = input("Enter the name of the course: ")

            student = Student(first_name=student_first_name,
                              last_name=student_last_name,
                              course_name=course_name)

            student_data.append(student)
            print()
            print(f"You have enrolled {student_first_name} {student_last_name} in course  {course_name}.")

        except ValueError as e:
            IO.output_error_messages( message: "That value is not the correct "\
                                     "type of data!", e)
        except Exception as e:
            IO.output_error_messages( message: "There was a non-specific error!", e)
        return student_data

```

Figure 10: PyCharm – Assignment07.py Python script input_student_data function

On Figure 11 below, we can see the output_current_student_data function:

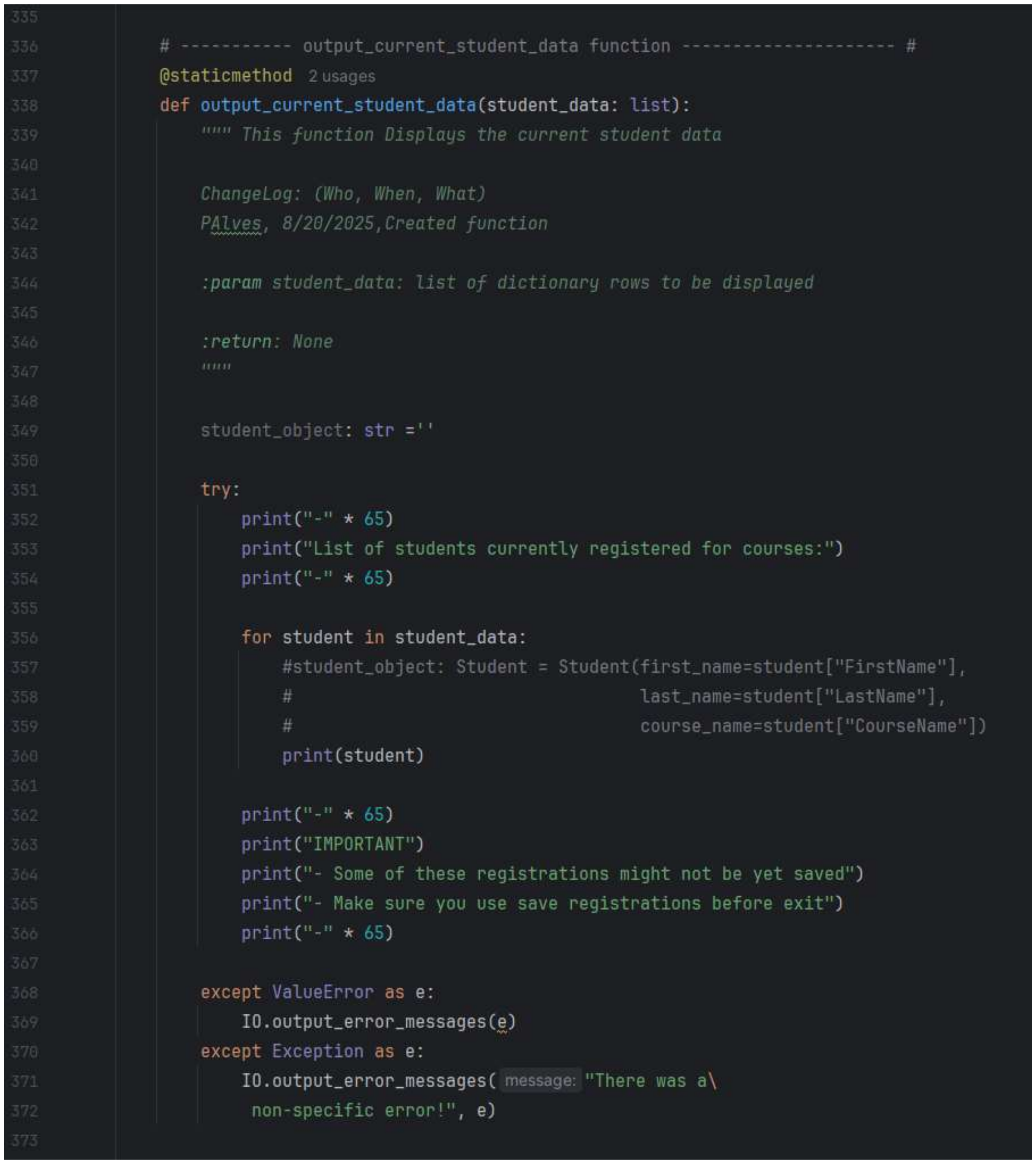
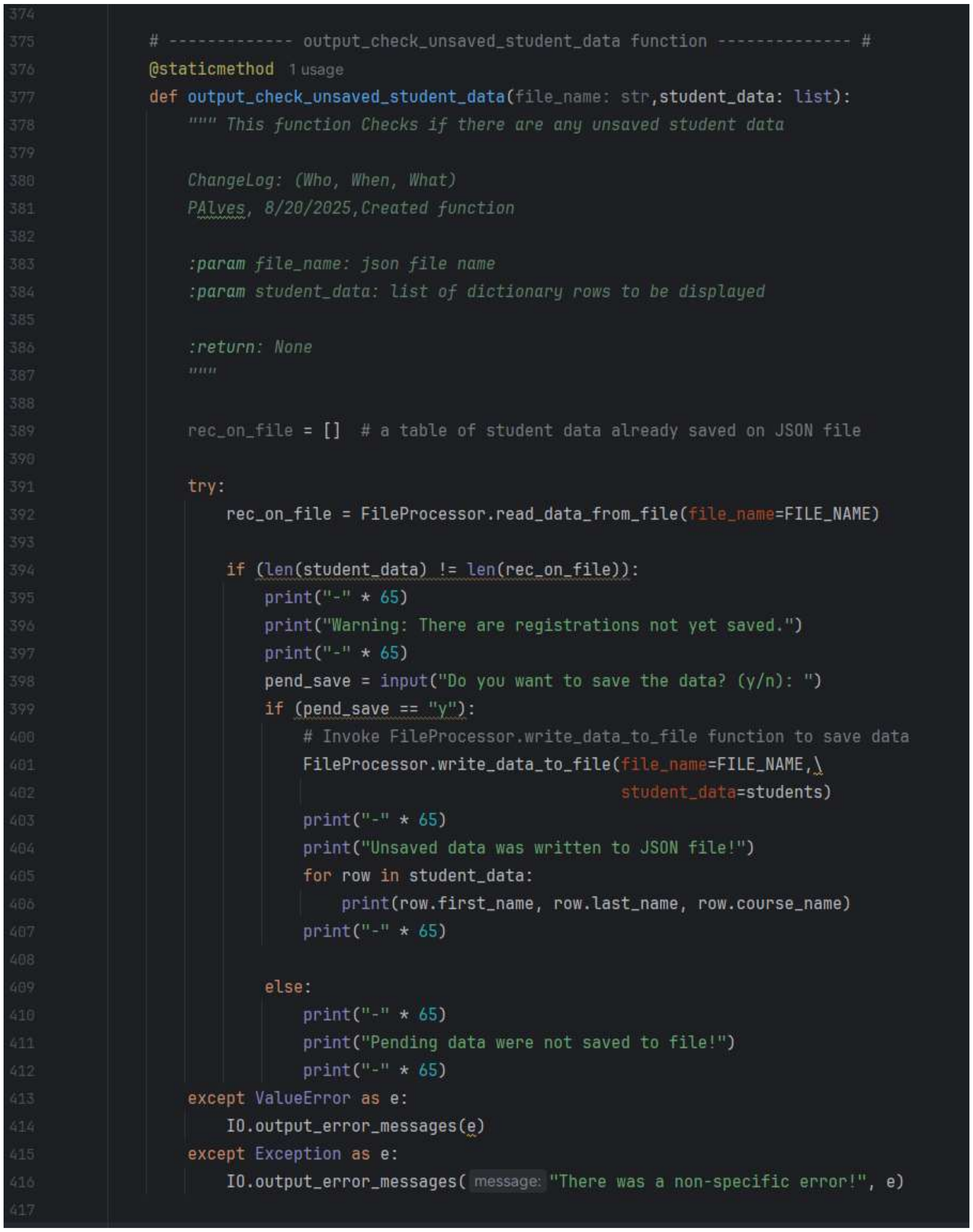

```python
    # ----------- output_current_student_data function --------------------- #
    @staticmethod  2 usages
    def output_current_student_data(student_data: list):
        """ This function Displays the current student data

        ChangeLog: (Who, When, What)
        PAlves, 8/20/2025,Created function

        :param student_data: list of dictionary rows to be displayed

        :return: None
        """

        student_object: str =''

        try:
            print("-" * 65)
            print("List of students currently registered for courses:")
            print("-" * 65)

            for student in student_data:
                #student_object: Student = Student(first_name=student["FirstName"],
                #                                   last_name=student["LastName"],
                #                                   course_name=student["CourseName"])
                print(student)

            print("-" * 65)
            print("IMPORTANT")
            print("- Some of these registrations might not be yet saved")
            print("- Make sure you use save registrations before exit")
            print("-" * 65)

        except ValueError as e:
            IO.output_error_messages(e)
        except Exception as e:
            IO.output_error_messages( message: "There was a\
             non-specific error!", e)
```

Figure 11: PyCharm – Assignment07.py Python script output_current_student_data function

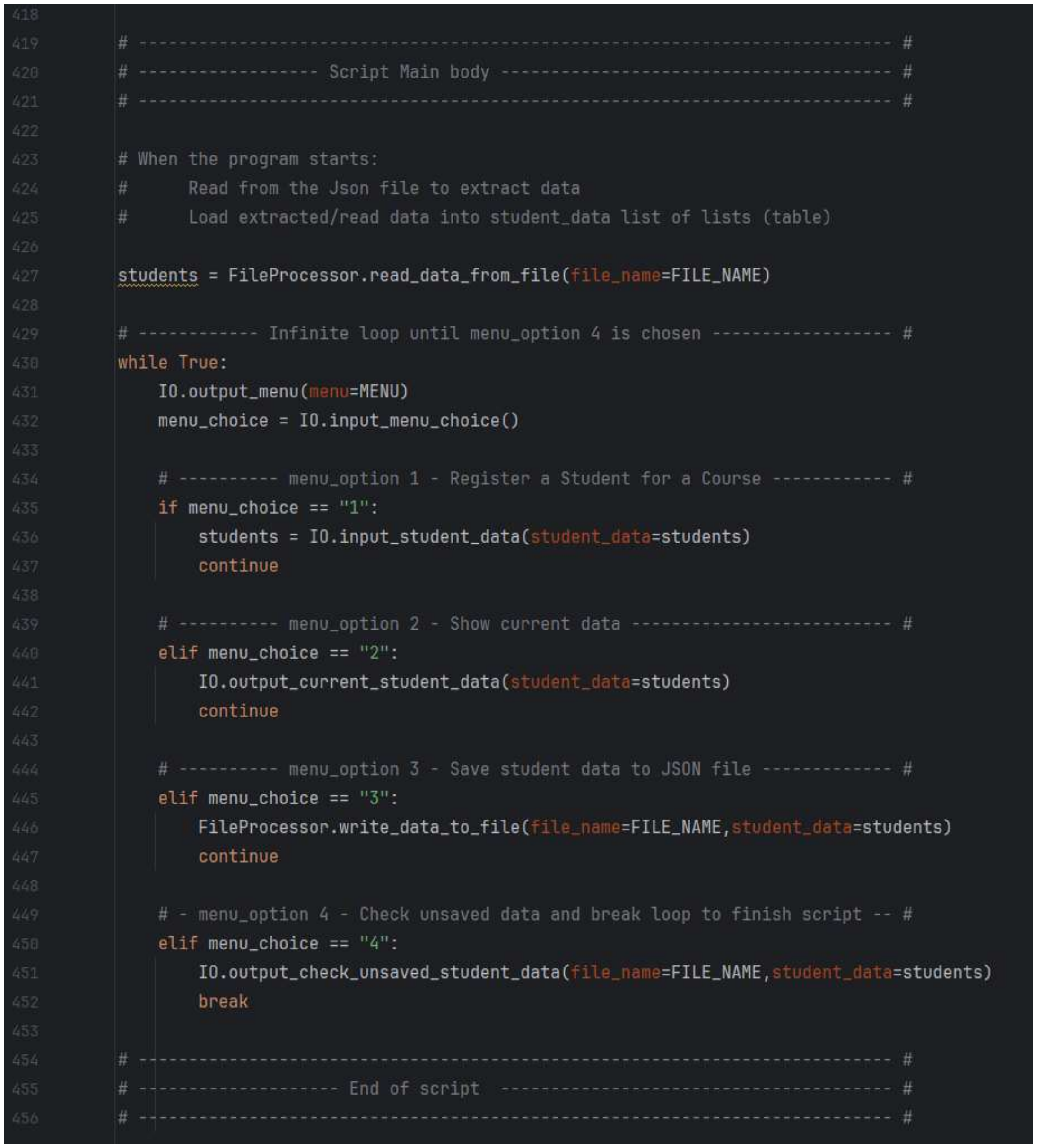
On Figure 12 below, we can see the output_check_unsaved_student_data function:



```python

    # ------------- output_check_unsaved_student_data function -------------- #
    @staticmethod  1 usage
    def output_check_unsaved_student_data(file_name: str,student_data: list):
        """ This function Checks if there are any unsaved student data

        ChangeLog: (Who, When, What)
        PAlves, 8/20/2025,Created function

        :param file_name: json file name
        :param student_data: list of dictionary rows to be displayed

        :return: None
        """

        rec_on_file = []  # a table of student data already saved on JSON file

        try:
            rec_on_file = FileProcessor.read_data_from_file(file_name=FILE_NAME)

            if (len(student_data) != len(rec_on_file)):
                print("-" * 65)
                print("Warning: There are registrations not yet saved.")
                print("-" * 65)
                pend_save = input("Do you want to save the data? (y/n): ")
                if (pend_save == "y"):
                    # Invoke FileProcessor.write_data_to_file function to save data
                    FileProcessor.write_data_to_file(file_name=FILE_NAME,\
                                                     student_data=students)
                    print("-" * 65)
                    print("Unsaved data was written to JSON file!")
                    for row in student_data:
                        print(row.first_name, row.last_name, row.course_name)
                    print("-" * 65)

                else:
                    print("-" * 65)
                    print("Pending data were not saved to file!")
                    print("-" * 65)
        except ValueError as e:
            IO.output_error_messages(e)
        except Exception as e:
            IO.output_error_messages( message: "There was a non-specific error!", e)

```

Figure 12: PyCharm – Assignment07.py Python script output_check_unsaved_student_data function

On Figure 13 below, we can see the Python script main body:



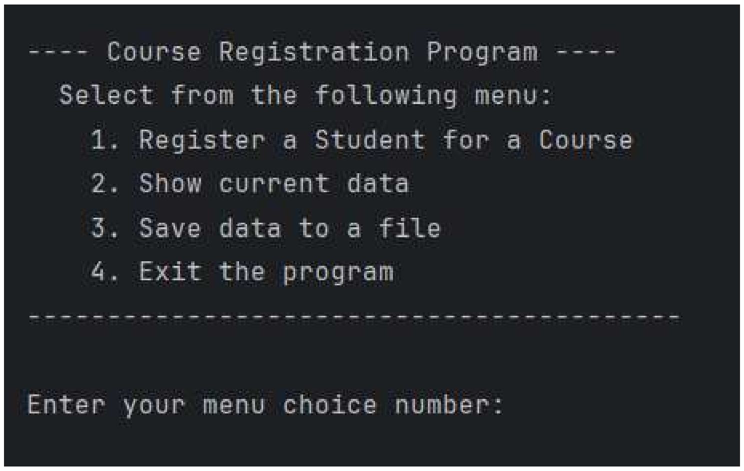
```
# ------------------------------------------------------------------------- #
# ----------------- Script Main body -------------------------------------- #
# ------------------------------------------------------------------------- #

# When the program starts:
#      Read from the Json file to extract data
#      Load extracted/read data into student_data list of lists (table)

students = FileProcessor.read_data_from_file(file_name=FILE_NAME)

# ------------ Infinite loop until menu_option 4 is chosen ------------------ #
while True:
    IO.output_menu(menu=MENU)
    menu_choice = IO.input_menu_choice()

    # ---------- menu_option 1 - Register a Student for a Course ------------ #
    if menu_choice == "1":
        students = IO.input_student_data(student_data=students)
        continue

    # ---------- menu_option 2 - Show current data -------------------------- #
    elif menu_choice == "2":
        IO.output_current_student_data(student_data=students)
        continue

    # ---------- menu_option 3 - Save student data to JSON file ------------- #
    elif menu_choice == "3":
        FileProcessor.write_data_to_file(file_name=FILE_NAME,student_data=students)
        continue

    # - menu_option 4 - Check unsaved data and break loop to finish script -- #
    elif menu_choice == "4":
        IO.output_check_unsaved_student_data(file_name=FILE_NAME,student_data=students)
        break

# ------------------------------------------------------------------------- #
# ------------------- End of script  -------------------------------------- #
# ------------------------------------------------------------------------- #
```

Figure 13: PyCharm – Assignment07.py Python script main body

Then I executed the script via PyCharm and via Windows command shell.

Executing script on PyCharm

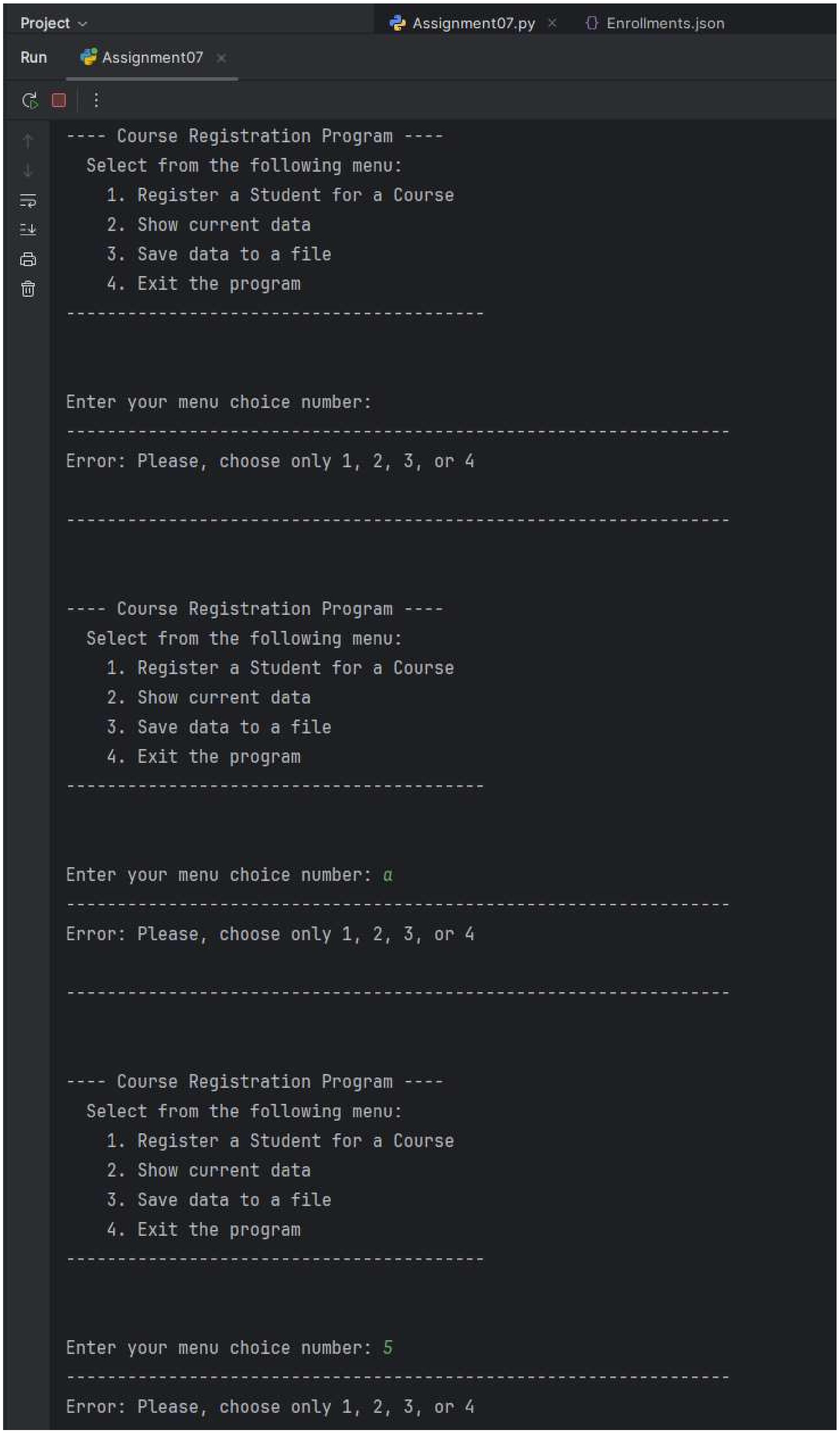
Figure 13 shown below displays the Assignment07.py Python script menu using PyCharm.



```
---- 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:
```

Figure 14: PyCharm – Assignment07.py script menu

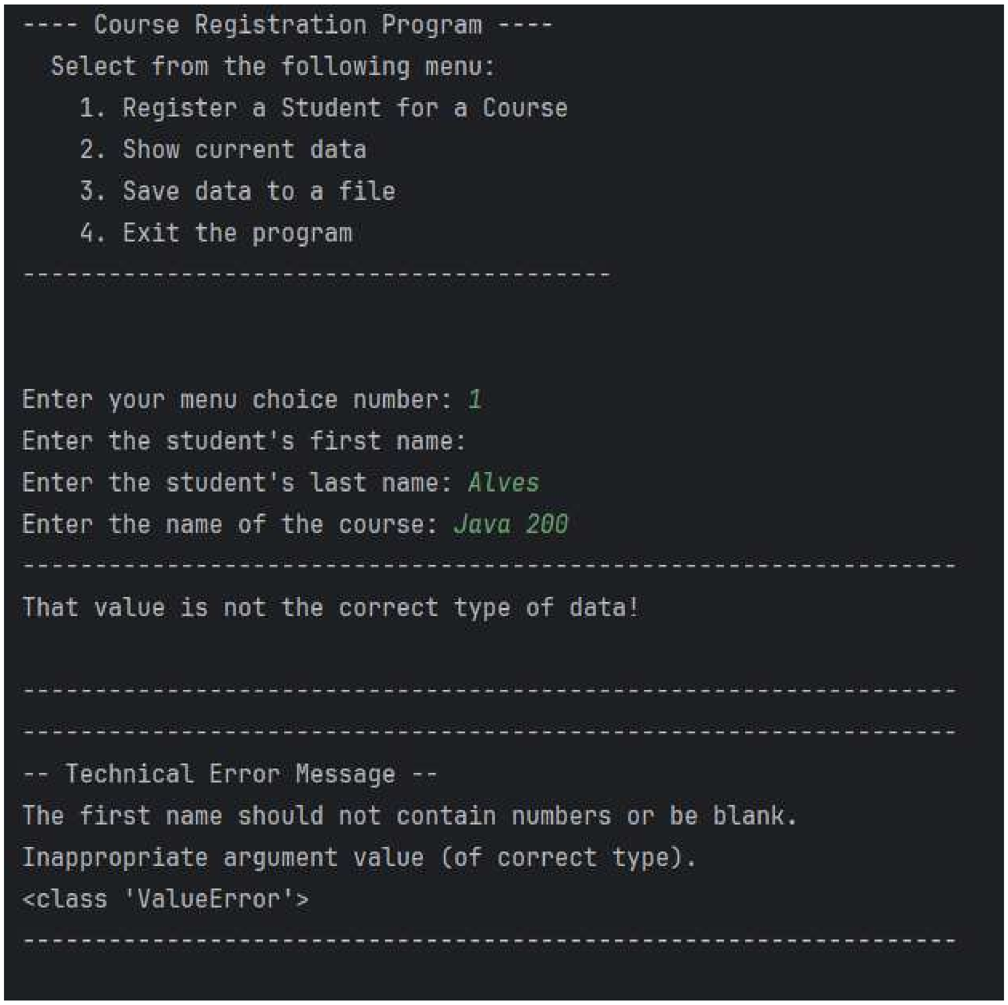
Figure 15 shown below displays the critique if an invalid menu option is chosen (either alphabetic options or integers different from 1, 2, 3 or 4):



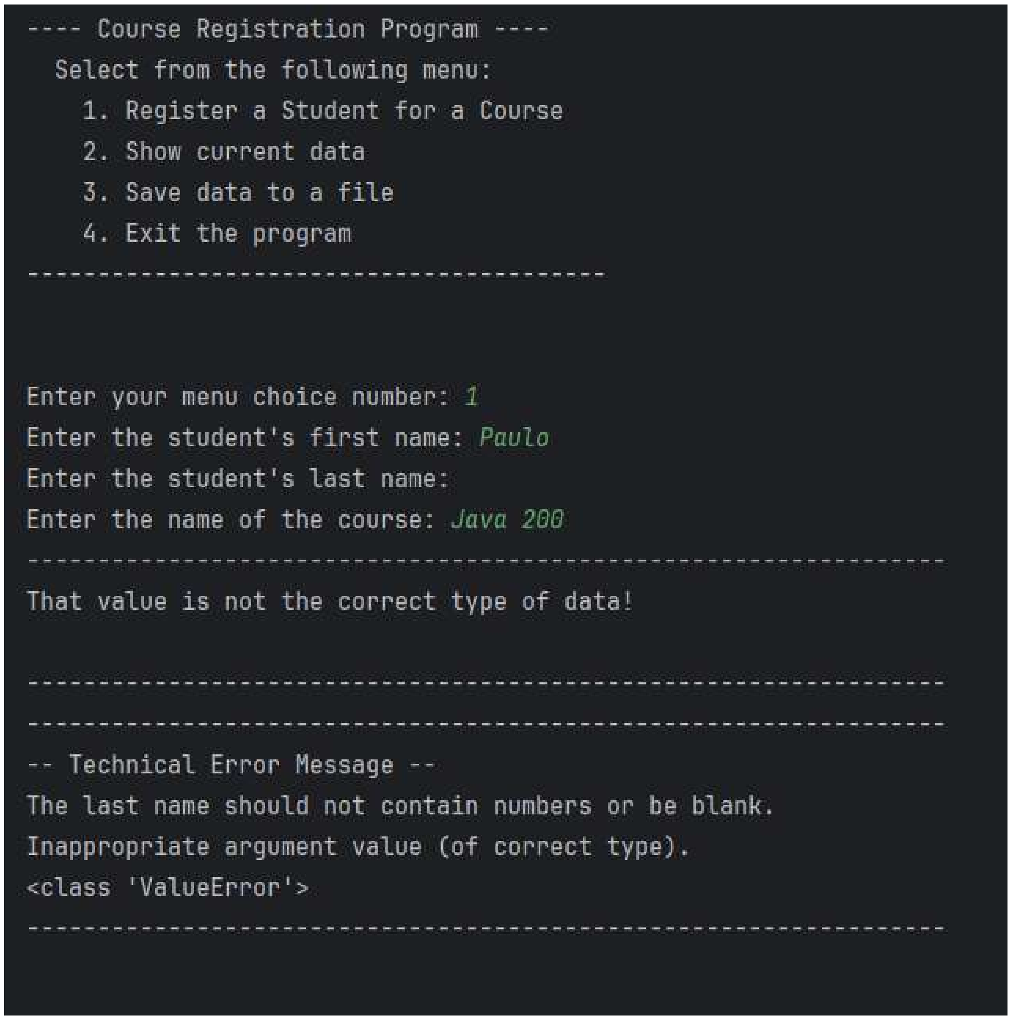
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course
    2. Show current data
    3. Save data to a file
    4. Exit the program
-----------------------------------------



Enter your menu choice number:
-----------------------------------------------------------------
Error: Please, choose only 1, 2, 3, or 4

-----------------------------------------------------------------



---- 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: a
-----------------------------------------------------------------
Error: Please, choose only 1, 2, 3, or 4

-----------------------------------------------------------------



---- 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: 5
-----------------------------------------------------------------
Error: Please, choose only 1, 2, 3, or 4
```

Figure 15: PyCharm – Assignment07.py script menu invalid option critique

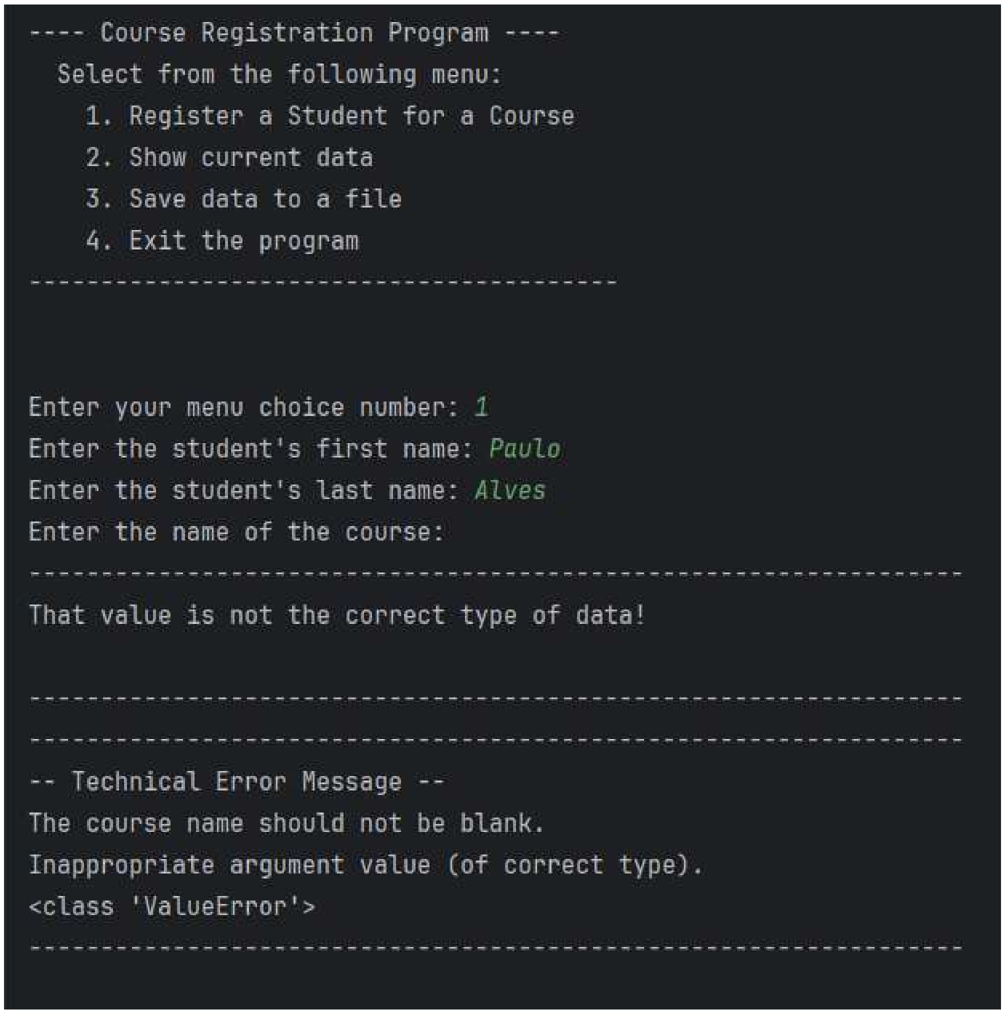
Figure 16 below shows error validation on Student's First and Last name and Course name invalid entries:



```
---- 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:
Enter the student's last name: Alves
Enter the name of the course: Java 200
-----------------------------------------------------------------
That value is not the correct type of data!

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



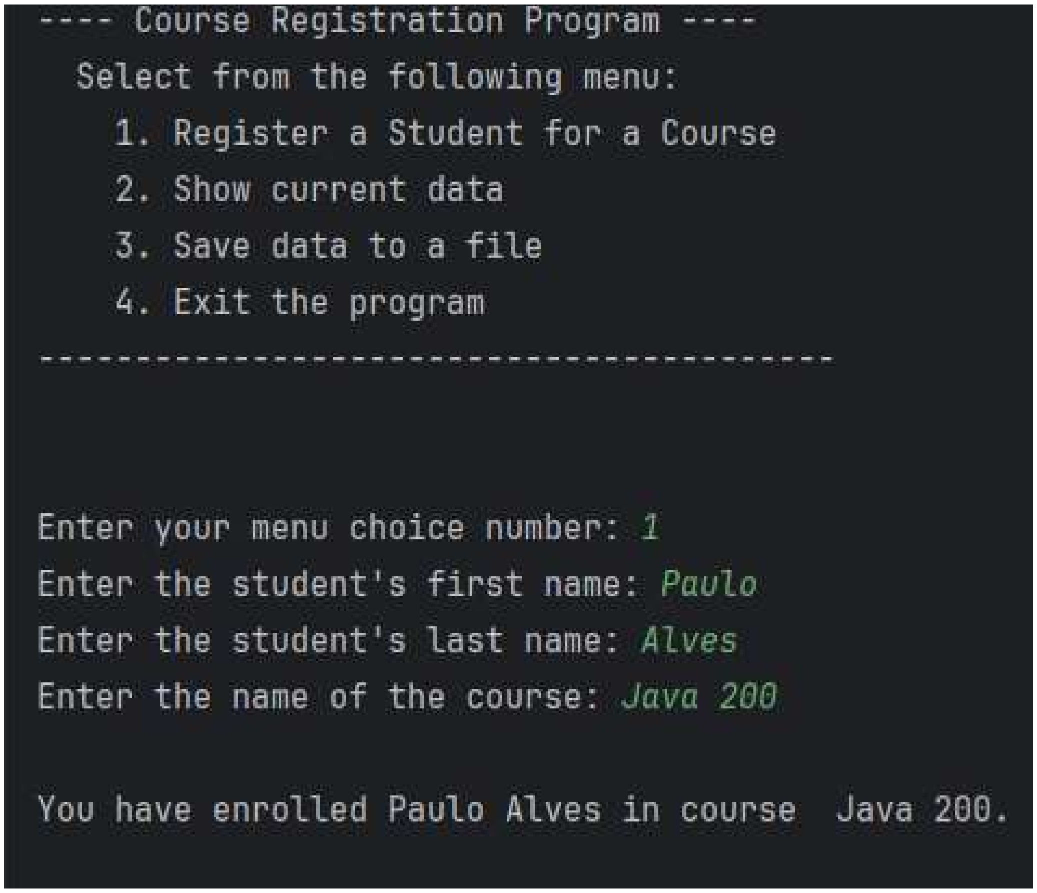
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course
    2. Show current data
    3. Save data to a file
    4. Exit the program
-----------------------------------------



Enter your menu choice number: 1
Enter the student's first name: Paulo
Enter the student's last name: 
Enter the name of the course: Java 200
-----------------------------------------------------------------
That value is not the correct type of data!

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



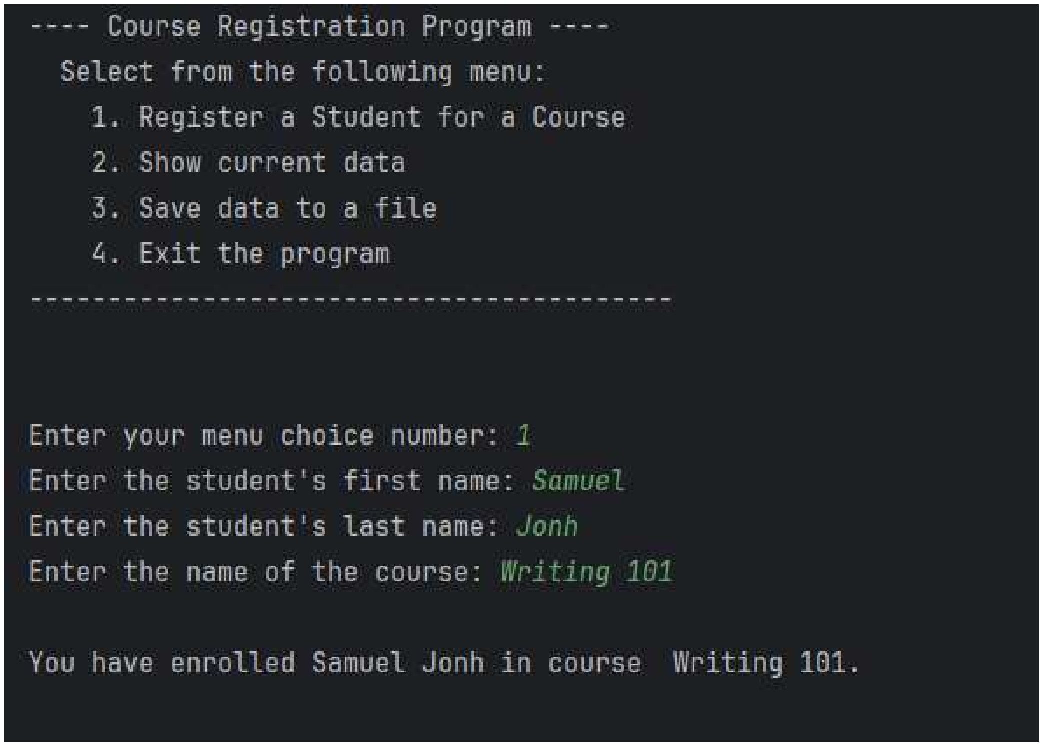
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course
    2. Show current data
    3. Save data to a file
    4. Exit the program
-----------------------------------------



Enter your menu choice number: 1
Enter the student's first name: Paulo
Enter the student's last name: Alves
Enter the name of the course: 
-----------------------------------------------------------------
That value is not the correct type of data!

-----------------------------------------------------------------
-----------------------------------------------------------------
-- Technical Error Message --
The course name should not be blank.
Inappropriate argument value (of correct type).
<class 'ValueError'>
-----------------------------------------------------------------
```

Figure 16: PyCharm – Assignment07.py script validation on Student's First and Last name and Course name invalid entries

Figure 17 below shows a couple of valid registrations made via Menu Option 1:



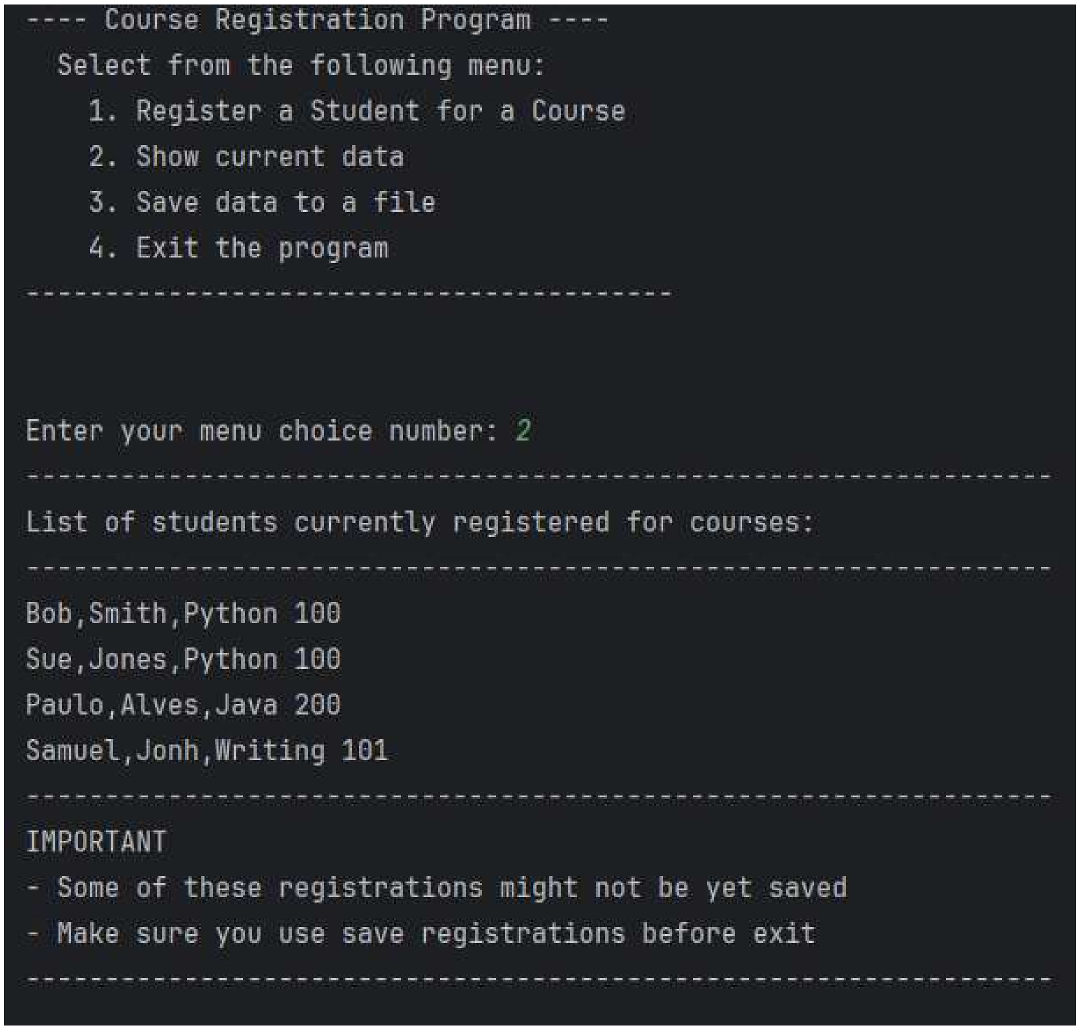
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course
    2. Show current data
    3. Save data to a file
    4. Exit the program
-----------------------------------------



Enter your menu choice number: 1
Enter the student's first name: Paulo
Enter the student's last name: Alves
Enter the name of the course: Java 200

You have enrolled Paulo Alves in course  Java 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: 1
Enter the student's first name: Samuel
Enter the student's last name: Jonh
Enter the name of the course: Writing 101

You have enrolled Samuel Jonh in course  Writing 101.
```

Figure 17: PyCharm – Assignment07.py script shows a couple of valid registrations via menu option1

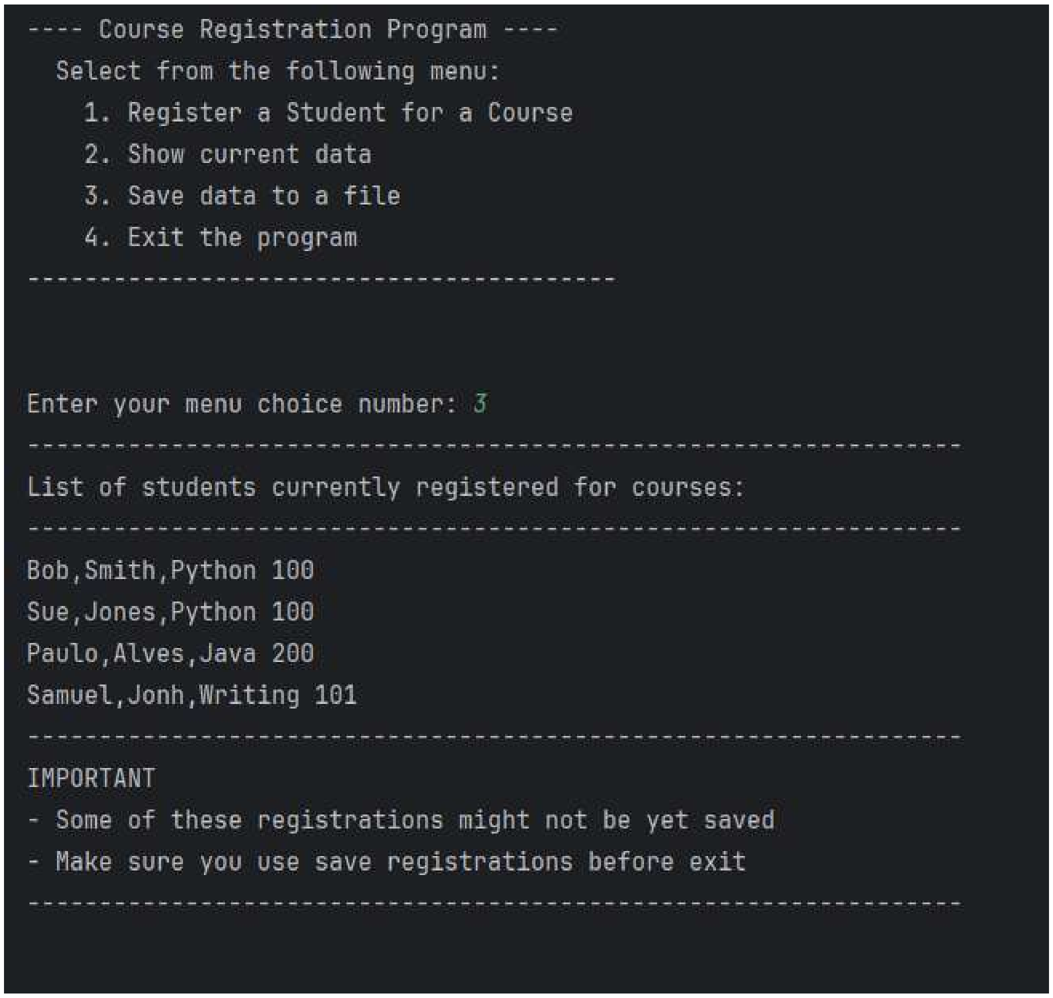
Figure 18 shown below displays the output of Menu option 2, when the current data is shown:



```
---- 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
------------------------------------------------------------------
List of students currently registered for courses:
------------------------------------------------------------------
Bob,Smith,Python 100
Sue,Jones,Python 100
Paulo,Alves,Java 200
Samuel,Jonh,Writing 101
------------------------------------------------------------------
IMPORTANT
- Some of these registrations might not be yet saved
- Make sure you use save registrations before exit
------------------------------------------------------------------
```

Figure 18: PyCharm – Assignment07.py script menu option 2 output showing current data

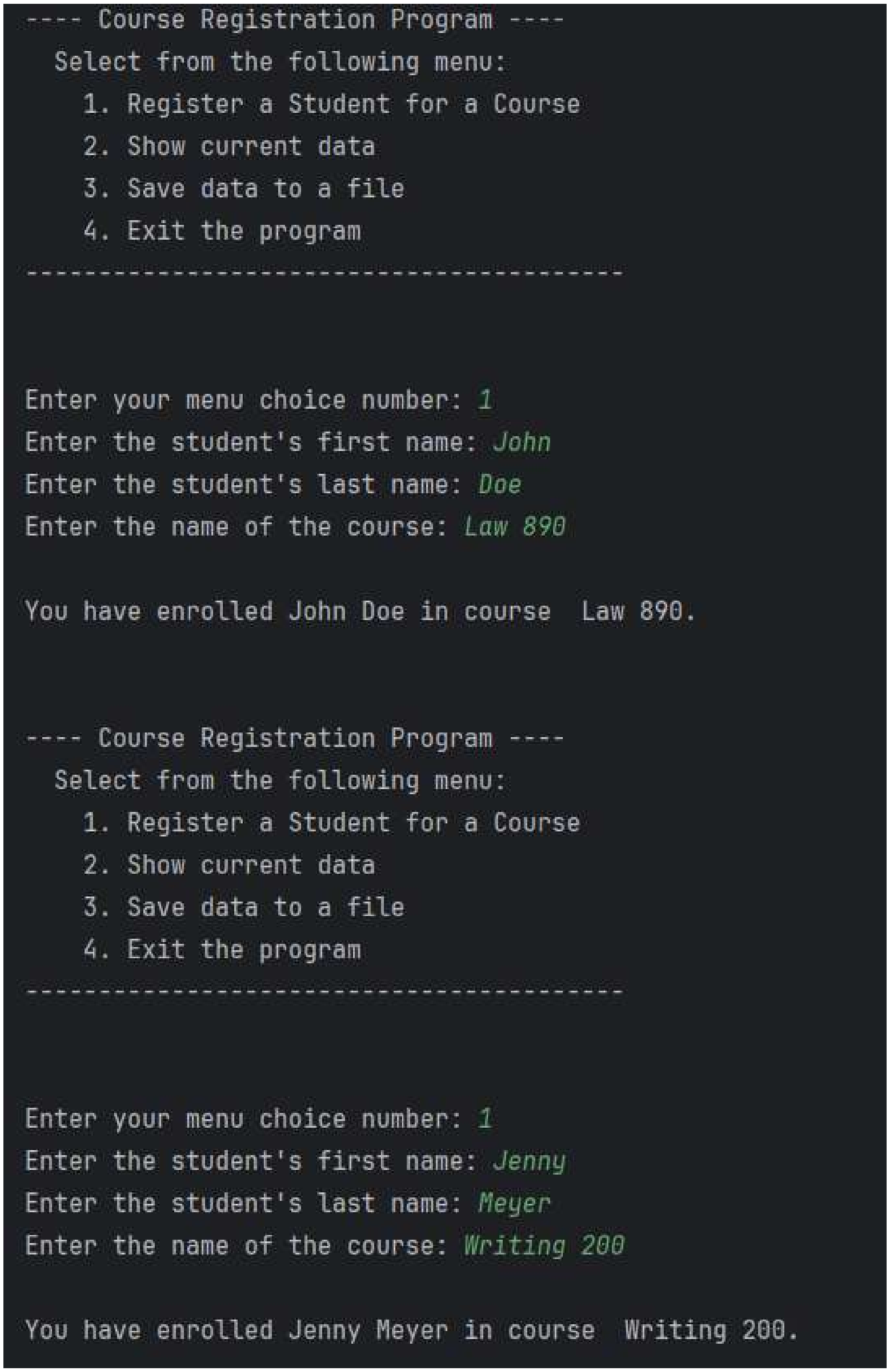
Figure 19 shown below displays the output of Menu option 3, when data is saved to JSON file and displayed:

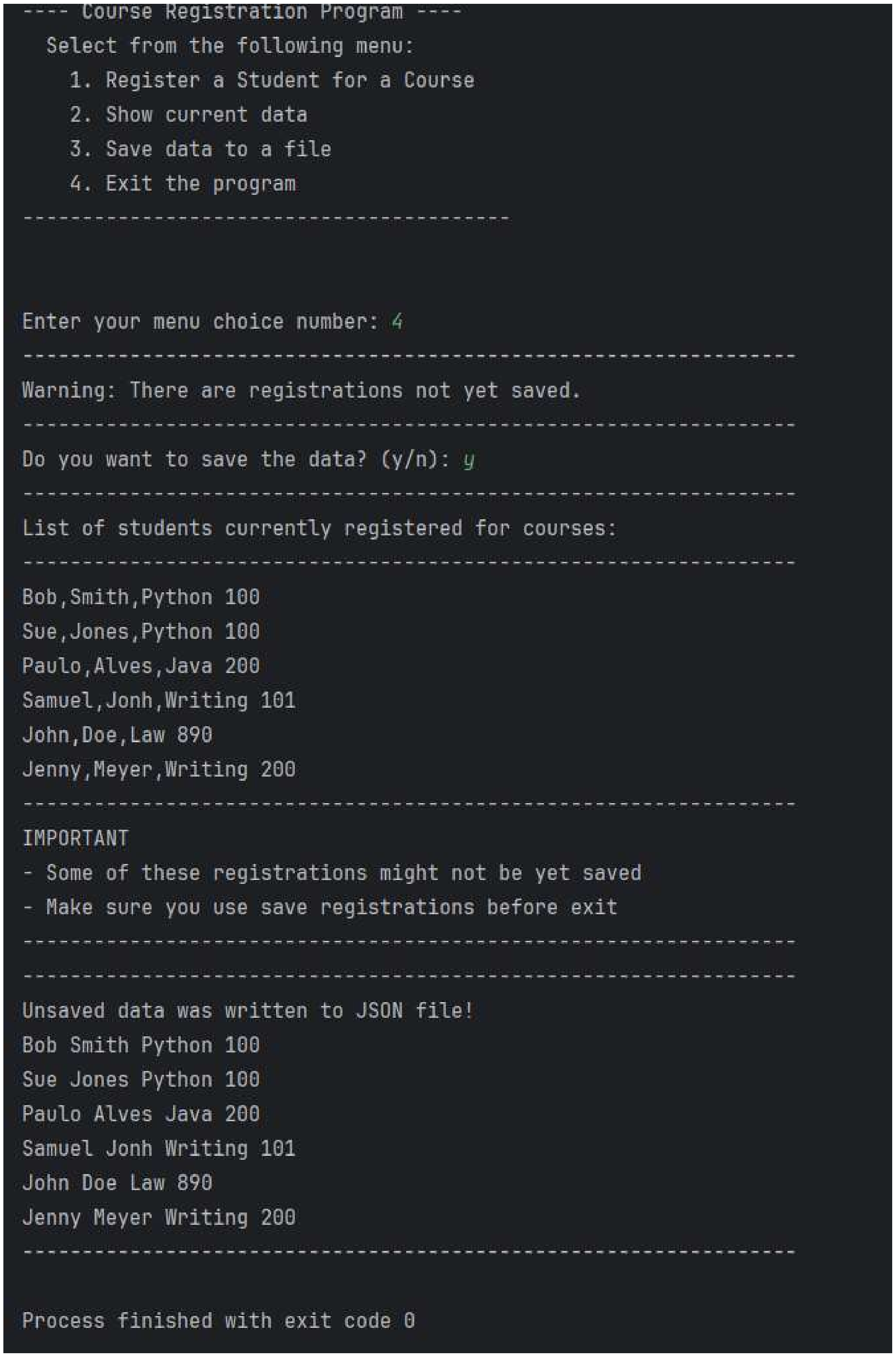


```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course
    2. Show current data
    3. Save data to a file
    4. Exit the program
-----------------------------------------



Enter your menu choice number: 3
-----------------------------------------------------------------
List of students currently registered for courses:
-----------------------------------------------------------------
Bob,Smith,Python 100
Sue,Jones,Python 100
Paulo,Alves,Java 200
Samuel,Jonh,Writing 101
-----------------------------------------------------------------
IMPORTANT
- Some of these registrations might not be yet saved
- Make sure you use save registrations before exit
-----------------------------------------------------------------
```

Figure 19: PyCharm – Assignment07.py script menu options 3 output

Figure 20 shown below displays the menu options 4 (when there is pending data to be saved to file):

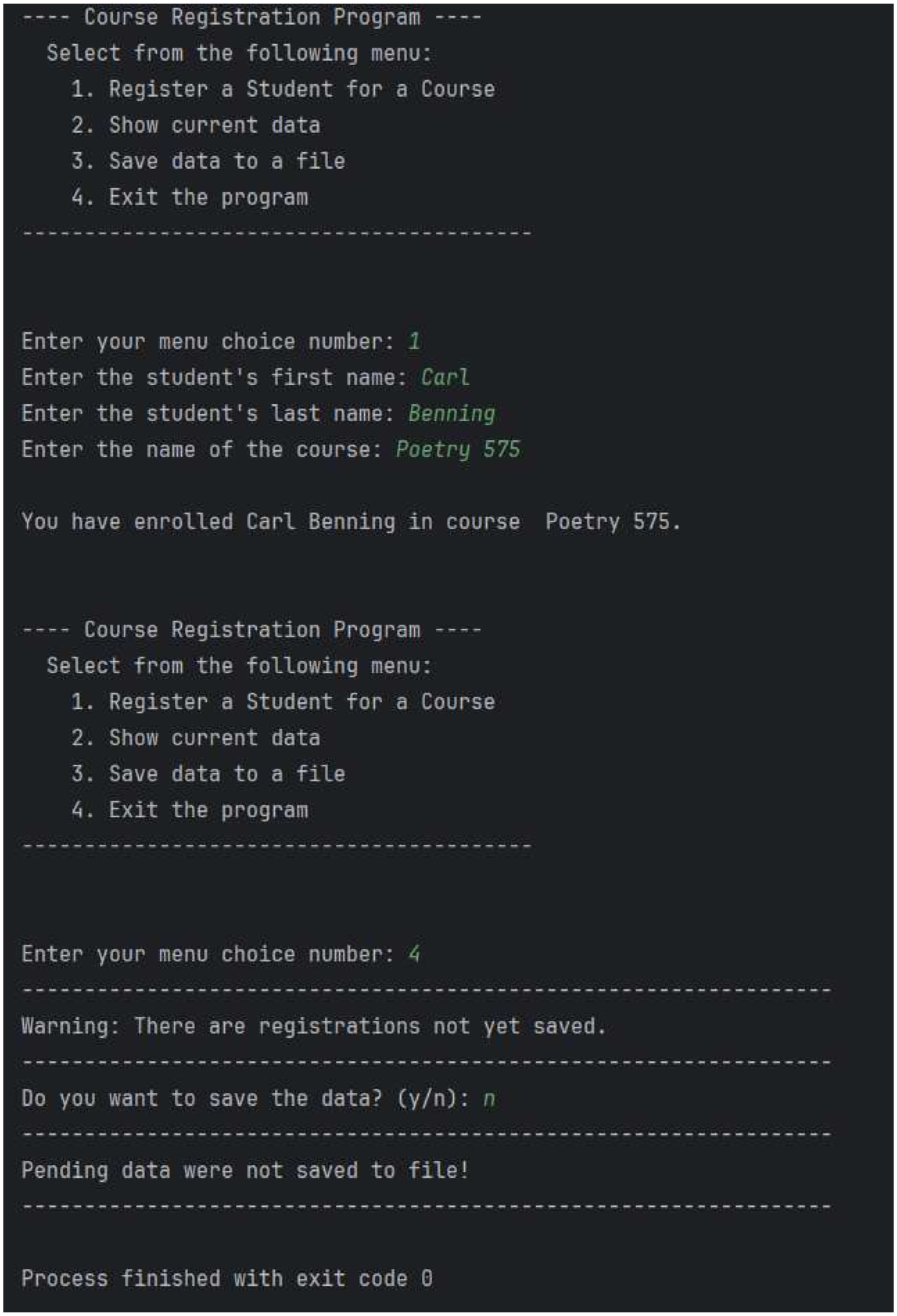

```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course
    2. Show current data
    3. Save data to a file
    4. Exit the program
-----------------------------------------



Enter your menu choice number: 1
Enter the student's first name: John
Enter the student's last name: Doe
Enter the name of the course: Law 890

You have enrolled John Doe in course  Law 890.



---- 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: Jenny
Enter the student's last name: Meyer
Enter the name of the course: Writing 200

You have enrolled Jenny Meyer in course  Writing 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
-----------------------------------------------------------------
Warning: There are registrations not yet saved.
-----------------------------------------------------------------
Do you want to save the data? (y/n): y
-----------------------------------------------------------------
List of students currently registered for courses:
-----------------------------------------------------------------
Bob,Smith,Python 100
Sue,Jones,Python 100
Paulo,Alves,Java 200
Samuel,Jonh,Writing 101
John,Doe,Law 890
Jenny,Meyer,Writing 200
-----------------------------------------------------------------
IMPORTANT
- Some of these registrations might not be yet saved
- Make sure you use save registrations before exit
-----------------------------------------------------------------
-----------------------------------------------------------------
Unsaved data was written to JSON file!
Bob Smith Python 100
Sue Jones Python 100
Paulo Alves Java 200
Samuel Jonh Writing 101
John Doe Law 890
Jenny Meyer Writing 200
-----------------------------------------------------------------


Process finished with exit code 0
```

Figure 20: PyCharm – Assignment07.py script menu option 4 output (when there is pending data)

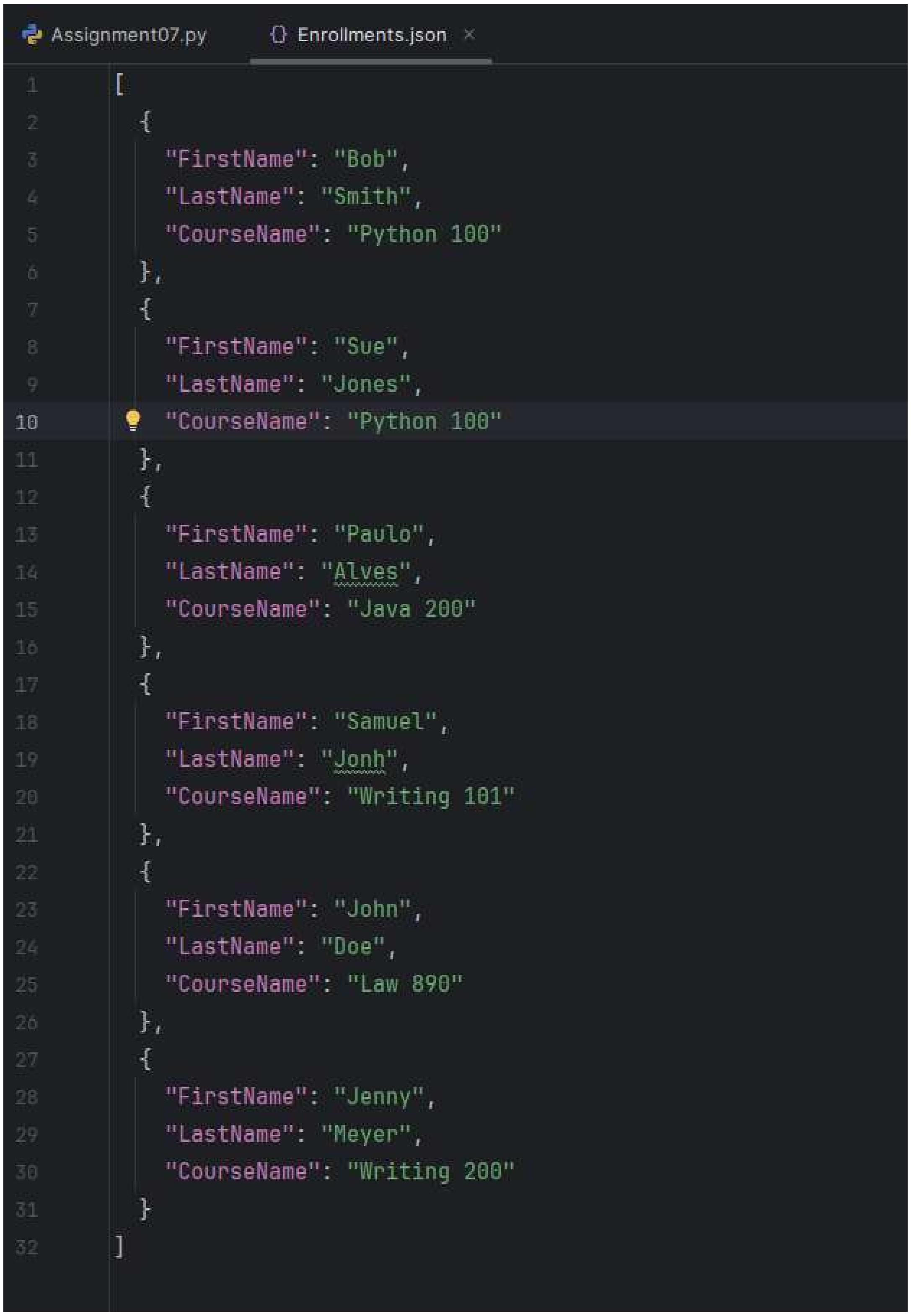
Figure 21 shown below displays the menu option 4, which exits the program (and no pending enrollment):



```
---- 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: Carl
Enter the student's last name: Benning
Enter the name of the course: Poetry 575

You have enrolled Carl Benning in course  Poetry 575.



---- 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
-----------------------------------------------------------------
Warning: There are registrations not yet saved.
-----------------------------------------------------------------
Do you want to save the data? (y/n): n
-----------------------------------------------------------------
Pending data were not saved to file!
-----------------------------------------------------------------


Process finished with exit code 0
```

Figure 21: PyCharm – Assignment07.py script menu option 4 with no pending enrollment

Figure 22 shown below displays the final content of the Enrollments.json file:

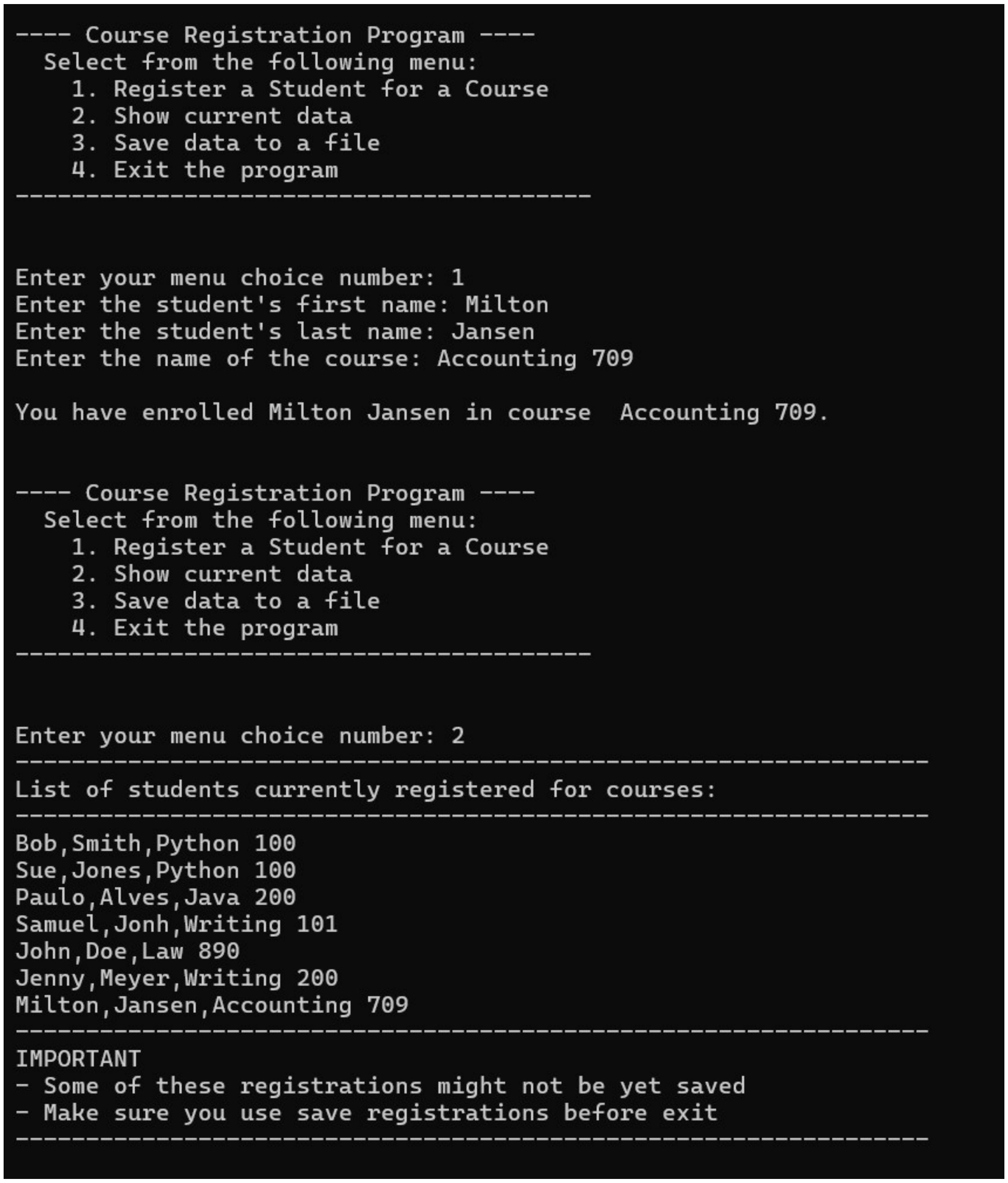


```
[
  {
    "FirstName": "Bob",
    "LastName": "Smith",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "Sue",
    "LastName": "Jones",
    "CourseName": "Python 100"
  },
  {
    "FirstName": "Paulo",
    "LastName": "Alves",
    "CourseName": "Java 200"
  },
  {
    "FirstName": "Samuel",
    "LastName": "Jonh",
    "CourseName": "Writing 101"
  },
  {
    "FirstName": "John",
    "LastName": "Doe",
    "CourseName": "Law 890"
  },
  {
    "FirstName": "Jenny",
    "LastName": "Meyer",
    "CourseName": "Writing 200"
  }
]
```

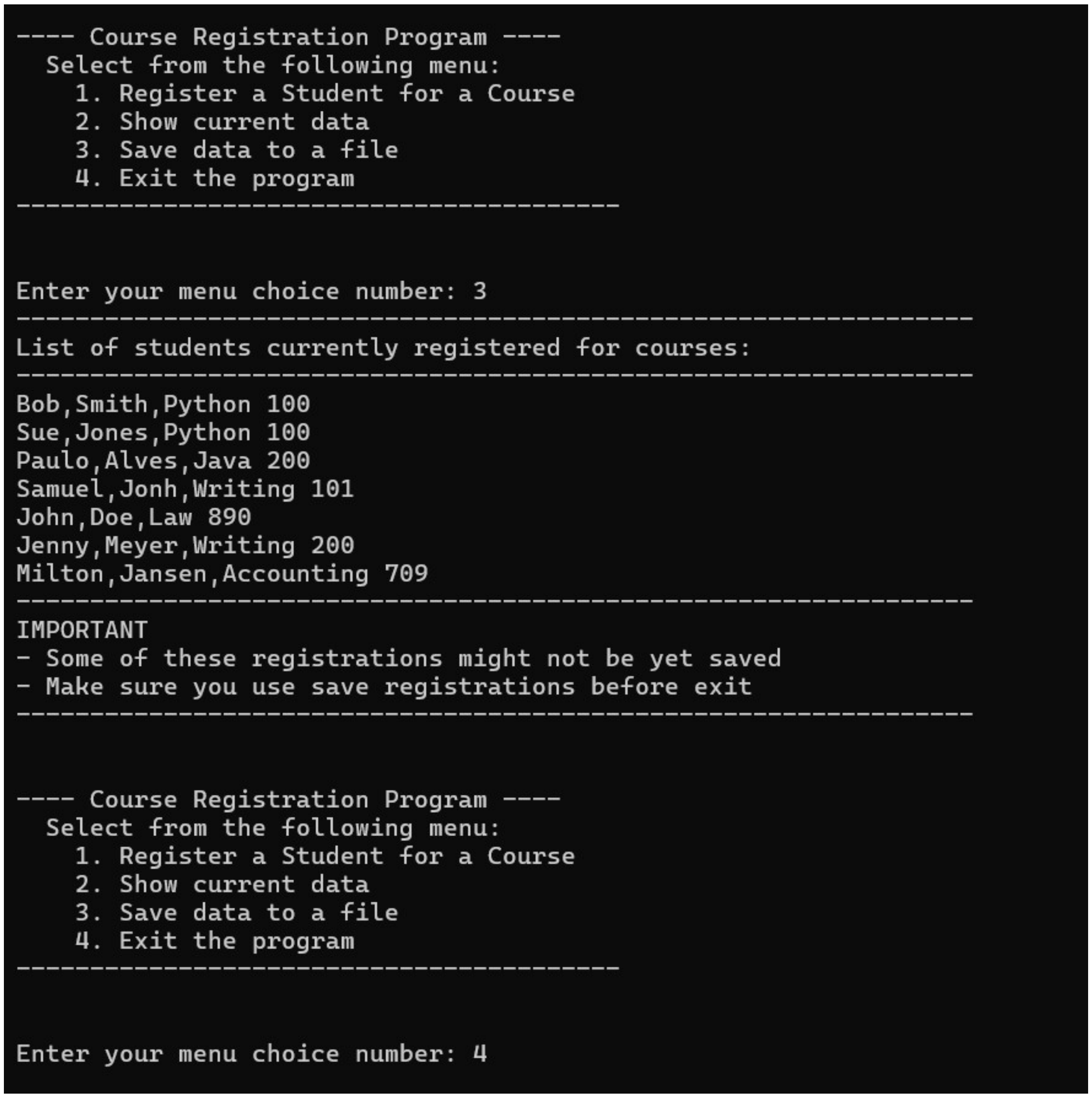
Figure 22: PyCharm – Assignment07.py script – final content on Enrollment.json file

Executing script on Windows command shell (cmd)

Figure 23 shown below, displays the successful execution of Assignment07.py Python script using Windows command shell (cmd).



```
---- 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: Milton
Enter the student's last name: Jansen
Enter the name of the course: Accounting 709

You have enrolled Milton Jansen in course  Accounting 709.


---- 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
-----------------------------------------------------------------
List of students currently registered for courses:
-----------------------------------------------------------------
Bob,Smith,Python 100
Sue,Jones,Python 100
Paulo,Alves,Java 200
Samuel,Jonh,Writing 101
John,Doe,Law 890
Jenny,Meyer,Writing 200
Milton,Jansen,Accounting 709
-----------------------------------------------------------------
IMPORTANT
- Some of these registrations might not be yet saved
- Make sure you use save registrations before exit
-----------------------------------------------------------------
```



```
---- 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
-----------------------------------------------------------------
List of students currently registered for courses:
-----------------------------------------------------------------
Bob,Smith,Python 100
Sue,Jones,Python 100
Paulo,Alves,Java 200
Samuel,Jonh,Writing 101
John,Doe,Law 890
Jenny,Meyer,Writing 200
Milton,Jansen,Accounting 709
-----------------------------------------------------------------
IMPORTANT
- Some of these registrations might not be yet saved
- Make sure you use save registrations before exit
-----------------------------------------------------------------


---- 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
```

Figure 23: Windows command shell (CMD) - Execution of Assignment07.py script

Summary

The creation and execution of this third Python script was a great way to enhance my Python programming using some of the new knowledge I learned on Module 07, including **the use of data classes.**