September 11th, 2024

IT FDN 110B: Foundations of Programming: Python

Assignment 07

GitHub Repository: https://github.com/martins-eder/IntroToProg-Python-Mod07

Professor: Luis Conejo

Student: Eder Martins

Creating a Python Script for Menu-Driven User Choices using Classes/Objects and Methods

Introduction

The goal of Assignment 07 was to create a Python program using PyCharm that presents the user with a menu from which he/she can choose to enter the student's data, present the current data, read from and save the data to a file, or exit the program. The assignment builds on the concepts learned in Assignments 05 and 06, incorporating more advanced concepts of separation of concerns pattern when using classes, objects, and methods to read from file and write to a file, as well as present options and process the inputs from the user. It also implemented error handling on reading input files and checks for unallowed characters for students' first name, last name, and course. This document outlines the steps taken to complete the assignment.

Preparation

To prepare for this assignment, I read the Module 07 notes on classes and functions. The notes extended the module 06 learnings on working with files, as well as how to feed data from files into the code and vice versa, but now utilizing functions and classes to perform the tasks. The notes samples on working with classes and functions were particularly useful in preparation for the coding assignment.

Writing and executing Module 07 Python script

After finishing reading module 07, I watched the module 07 YouTube videos from Professor Arya Ref [1] (see Figure 1), and having tried out the example Python code in the folders "Demos" and "LabAnswers," (see Figure 2), I felt prepared to start the programming assignment.

I used the provided Assignment07-Starter.py file, wich provided a good idea of how the code should be organized. The readings and videos made writing the script (Figure 3) much easier, especially the Modul07-Lab03. The steps I followed can be described as:

1. *Update the header*: with my name and current date
2. *Define constants*: set the constant for the menu options ('MENU') and the file name ('FILE_NAME') where the data will be stored and/or read from.
3. *Define variables*: Initialized the students list, which stores the data of student dictionaries, and the menu_choice variable to store the user's selected menu option.
4. *Check for existing data*: Implemented file handling by reading from the file using the FileProcessor.read_data_from_file(FILE_NAME, students) method. If the file exists, data is loaded into the students list.
5. *Display the menu:* Used a while True loop to continuously display the menu and prompt the user for input. The IO.output_menu() function displays the menu, and the IO.input_menu_choice() function captures the user's choice.
6. *Handle user choices*: Based on the user's selection, perform the following:
    - Register a student: The program prompts the user for student details (first name, last name, and course name) via IO.input_student_data(). The input is appended to the students list. Added error handling to ensure that only valid alphabetic names are accepted for first and last names, raising a ValueError for invalid input.
    - Show current data: Displays all registered students and their enrolled courses via IO.output_student_and_course_names().
    - Save data to a file: Saves the current student data to the Enrollments.json file using the FileProcessor.write_data_to_file(FILE_NAME, students) method. Also displays confirmation and the saved data.
7. *Test the Program*: run the program in PyCharm (Figure 4) and from the console (Figure 5) to ensure it worked as expected. Verified that the output was saved correctly in the Enrollments.json file (Figure 6).

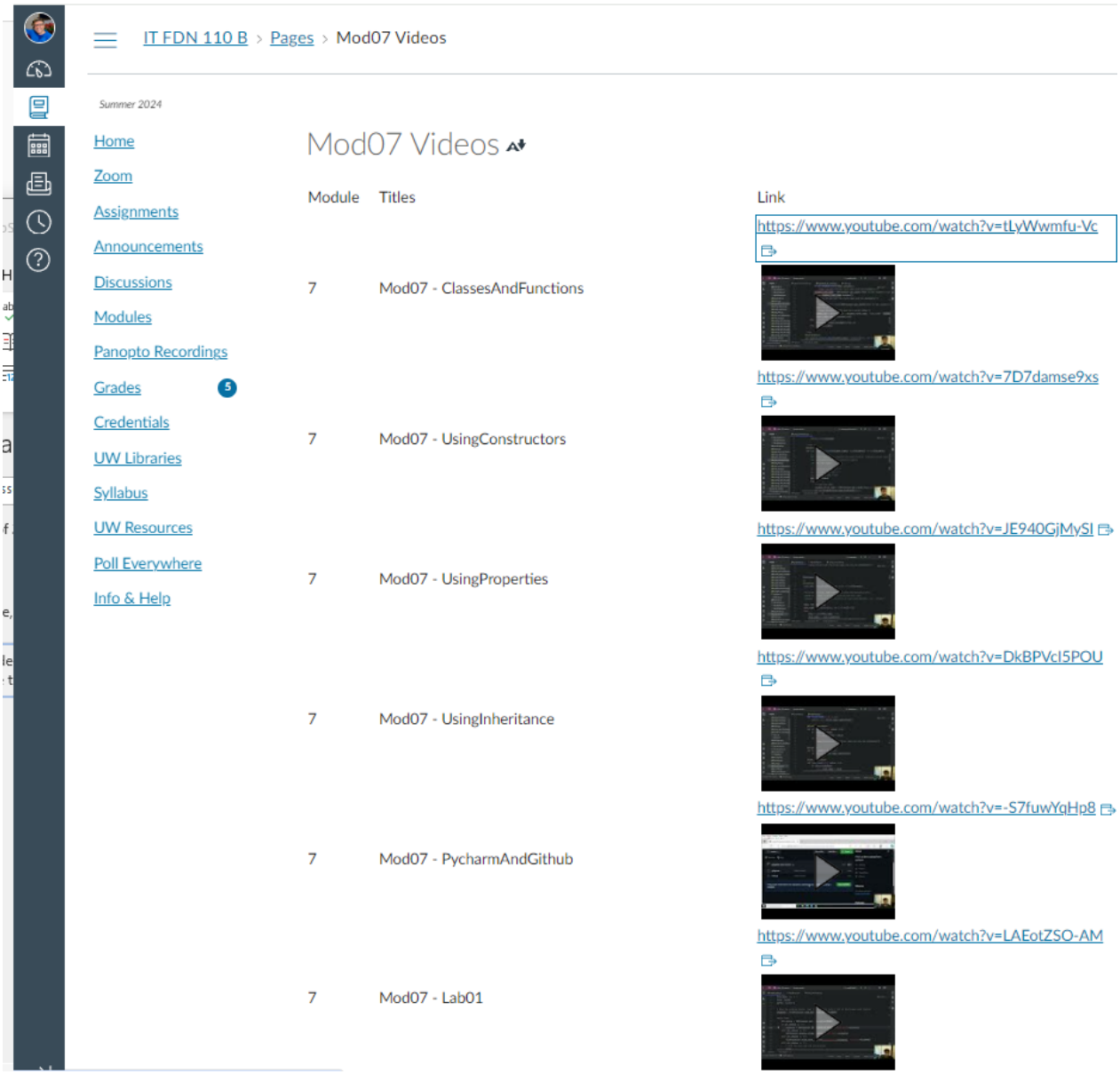

Figure 1 Module 07 Videos (screenshot from video's list in Canvas)

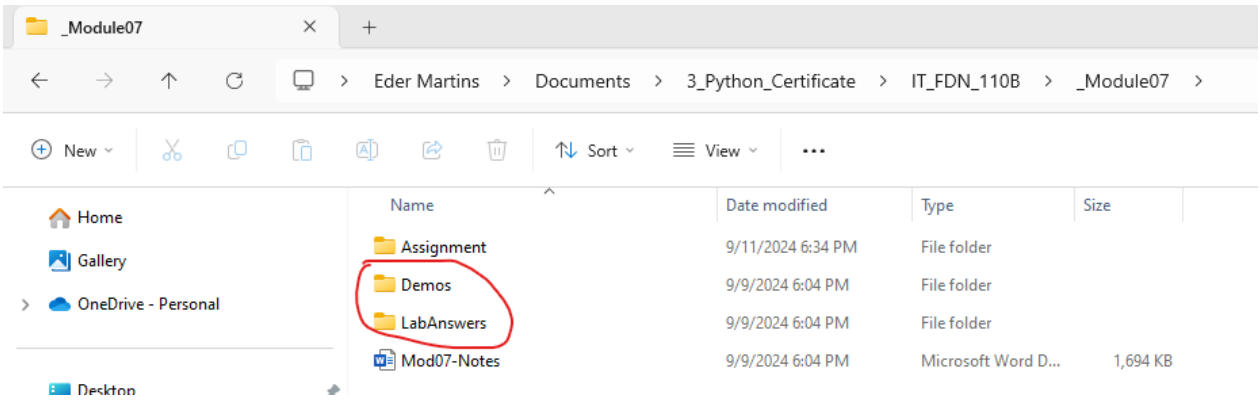

Figure 2 Module 07 subfolders "Demos" and "LabAnswers" highlighted

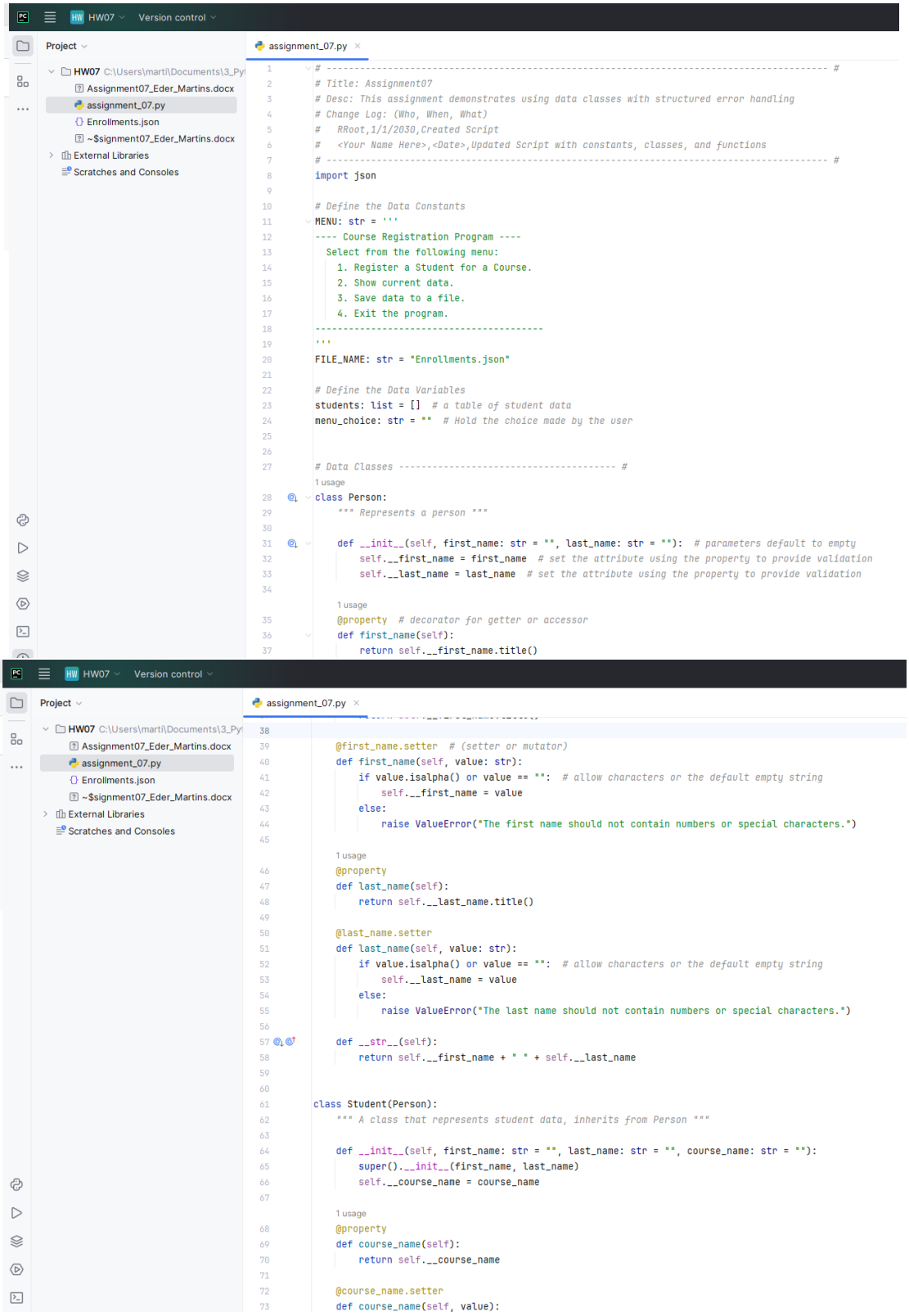

Figure 3 Python Script in PyCharm using Classes and Functions

```
        self.__course_name = value

    def __str__(self):
        return super().__str__() + " is enrolled in " + self.__course_name


# Processing ------------------------------------- #
2 usages
class FileProcessor:
    """ A collection of processing layer functions that work with Json files
        ChangeLog: (Who, When, What)
        Eder Martins,9/11/2024,Created Class
    """

    1 usage
    @staticmethod
    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)
            Eder Martins,9/11/2024,Created function

            :param file_name: string data with name of file to read from
            :param student_data: list of dictionary rows to be filled with file data

            :return: list
        """

        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            file.close()
        except Exception as e:
            IO.output_error_messages(message="Error: There was a problem with reading the file.", error=e)

        finally:
            if file.closed == False:
                file.close()
        return student_data

    1 usage
    @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)
            Eder Martins,9/11/2024,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.dump(student_data, file)
            file.close()
            # After successfully writing to the file, print confirmation and the data
            print("\nData has been successfully saved to the file. The following data was stored:")
            IO.output_student_and_course_names(student_data=student_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()


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

        ChangeLog: (Who, When, What)
        Eder Martins,9/11/2024,Created Class
```

Figure 3 Python Script in PyCharm using Classes and Functions (contnd)

```
    Eder Martins,9/11/2024,Added menu output and input functions
    Eder Martins,9/11/2024,Added a function to display the data
    Eder Martins,9/11/2024,Added a function to display custom error messages
    """

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

            ChangeLog: (Who, When, What)
            Eder Martins,9/11/2024,Created function

            :param message: string with message data to display
            :param error: Exception object with technical message to display

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

    1 usage
    @staticmethod
    def output_menu(menu: str):
        """ This function displays the menu of choices to the user

            ChangeLog: (Who, When, What)
            Eder Martins,9/11/2024,Created function


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

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

            ChangeLog: (Who, When, What)
            Eder Martins,9/11/2024,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"):
                raise Exception("Please, choose only 1, 2, 3, or 4")
        except Exception as e:
            IO.output_error_messages(e.__str__()) # Not passing e to avoid the technical message

        return choice

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

            ChangeLog: (Who, When, What)
            Eder Martins,9/11/2024,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["FirstName"]} '
                  f'{student["LastName"]} is enrolled in {student["CourseName"]}')
        print("-" * 50)
```

Figure 3 Python Script in PyCharm using Classes and Functions (contnd)

```
220
        1 usage
221         @staticmethod
222         def input_student_data(student_data: list):
223             """ This function gets the student's first name and last name, with a course name from the user
224
225             ChangeLog: (Who, When, What)
226             Eder Martins,9/11/2024,Created function
227
228             :param student_data: list of dictionary rows to be filled with input data
229
230             :return: list
231             """
232
233             try:
234                 student_first_name = input("Enter the student's first name: ")
235                 if not student_first_name.isalpha():
236                     raise ValueError("The first name should not contain numbers or spaces.")
237                 student_last_name = input("Enter the student's last name: ")
238                 if not student_last_name.isalpha():
239                     raise ValueError("The last name should not contain numbers or spaces.")
240                 course_name = input("Please enter the name of the course: ")
241                 student = {"FirstName": student_first_name, "LastName": student_last_name, "CourseName": course_name}
242                 student_data.append(student)
243                 print()
244                 print(f"\nYou have registered {student_first_name} {student_last_name} for {course_name}.")
245             except ValueError as e:
246                 IO.output_error_messages(message="One of the values was the correct type of data!", error=e)
247             except Exception as e:
248                 IO.output_error_messages(message="Error: There was a problem with your entered data.", error=e)
249             return student_data
250
251
252     # Start of main body
253
254     # When the program starts, read the file data into a list of lists (table)
255     # Extract the data from the file
256     students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)
257
258     # Present and Process the data
259     while (True):
260
261         # Present the menu of choices
262         IO.output_menu(menu=MENU)
263
264         menu_choice = IO.input_menu_choice()
265
266         # Input user data
267         if menu_choice == "1":  # This will not work if it is an integer!
268             students = IO.input_student_data(student_data=students)
269             continue
270
271         # Present the current data
272         elif menu_choice == "2":
273             IO.output_student_and_course_names(students)
274             continue
275
276         # Save the data to a file
277         elif menu_choice == "3":
278             FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
279             continue
280
281         # Stop the loop
282         elif menu_choice == "4":
283             break  # out of the loop
284         else:
285             print("Please only choose option 1, 2, or 3")
286
287     print("Program Ended")
288
```

Figure 3 Python Script in PyCharm using Classes and Functions (contnd)

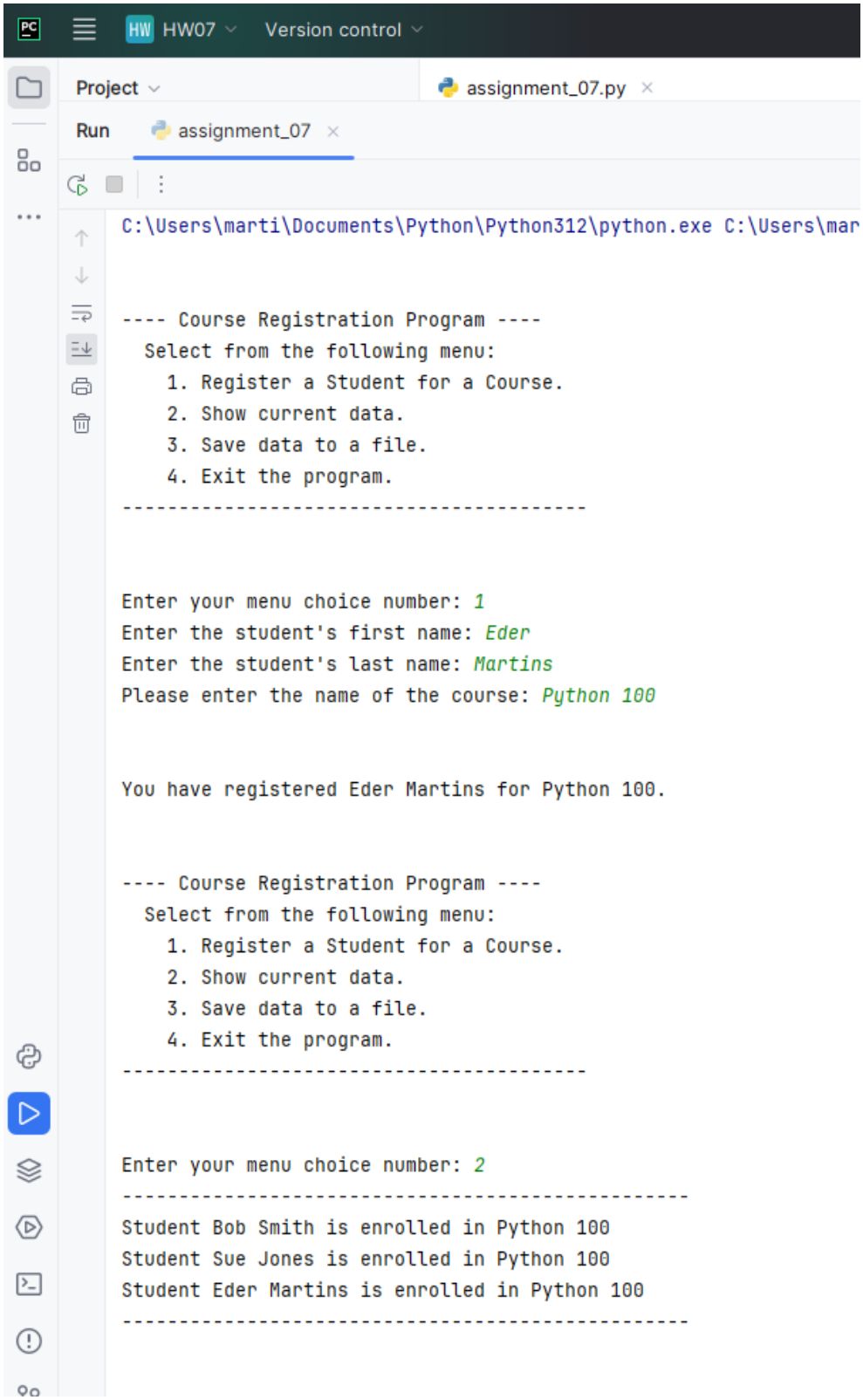

Figure 4 Executing Python Script in PyCharm

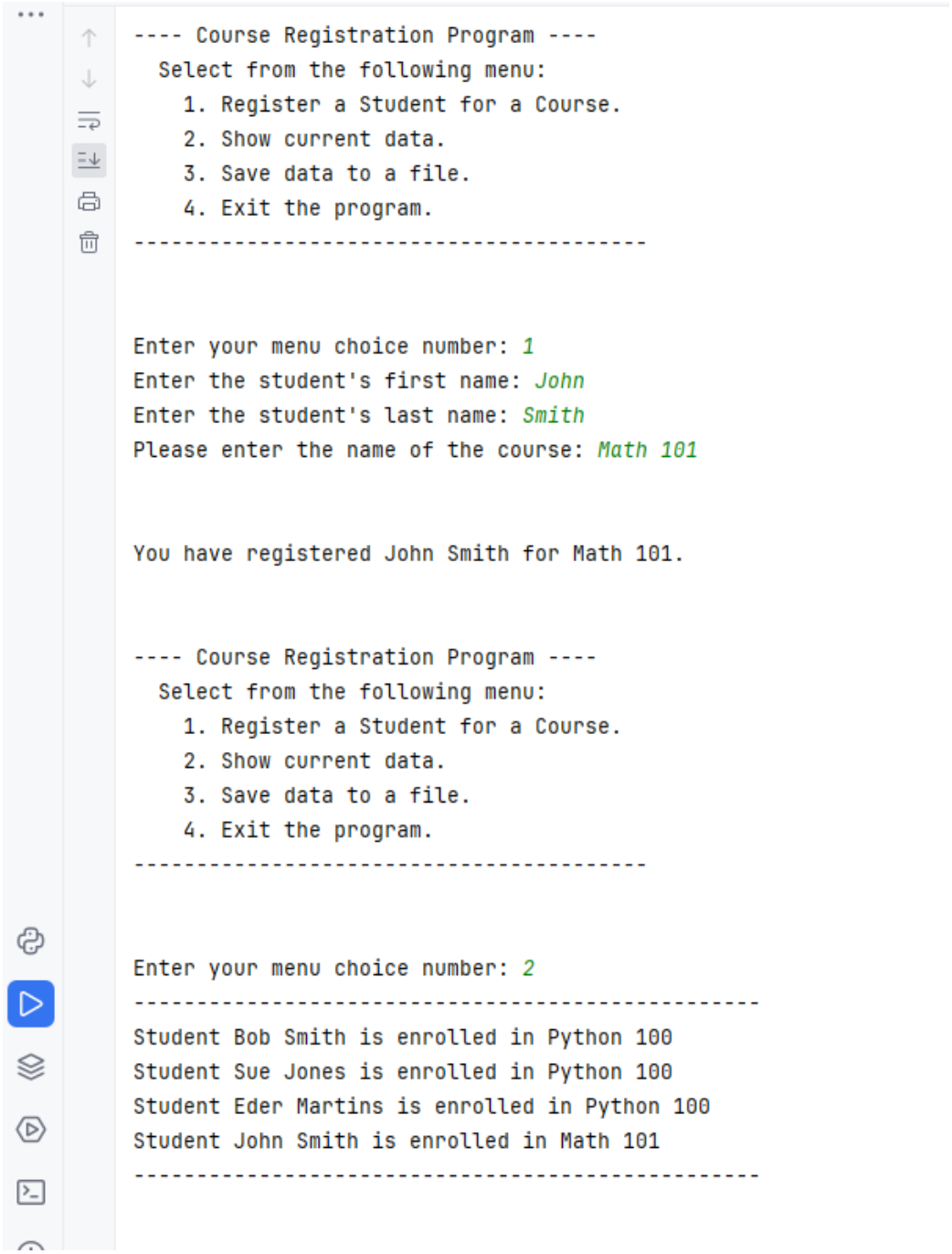

Figure 4 Executing Python Script in PyCharm (contnd)

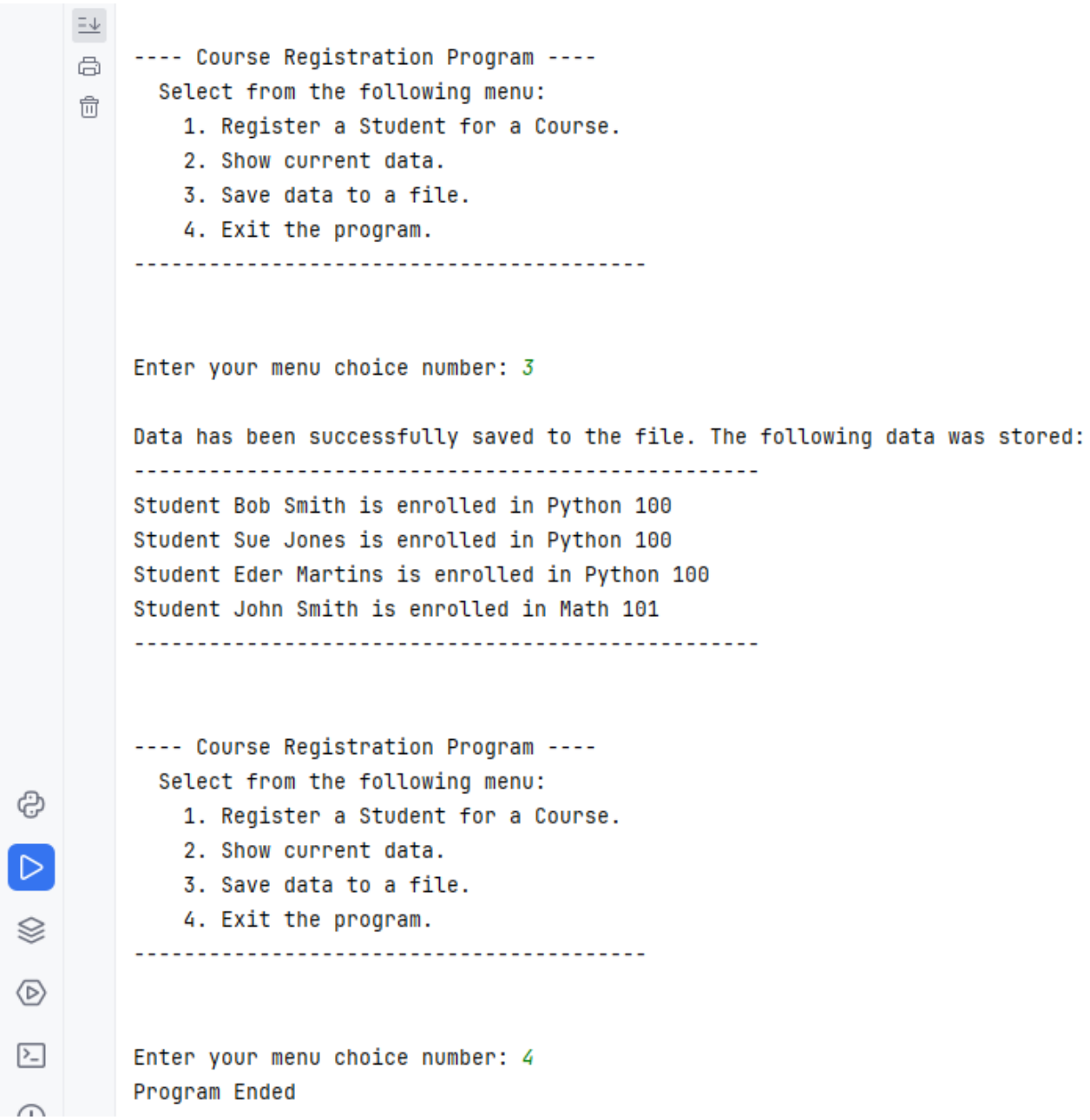

Figure 4 Executing Python Script in PyCharm (contnd)

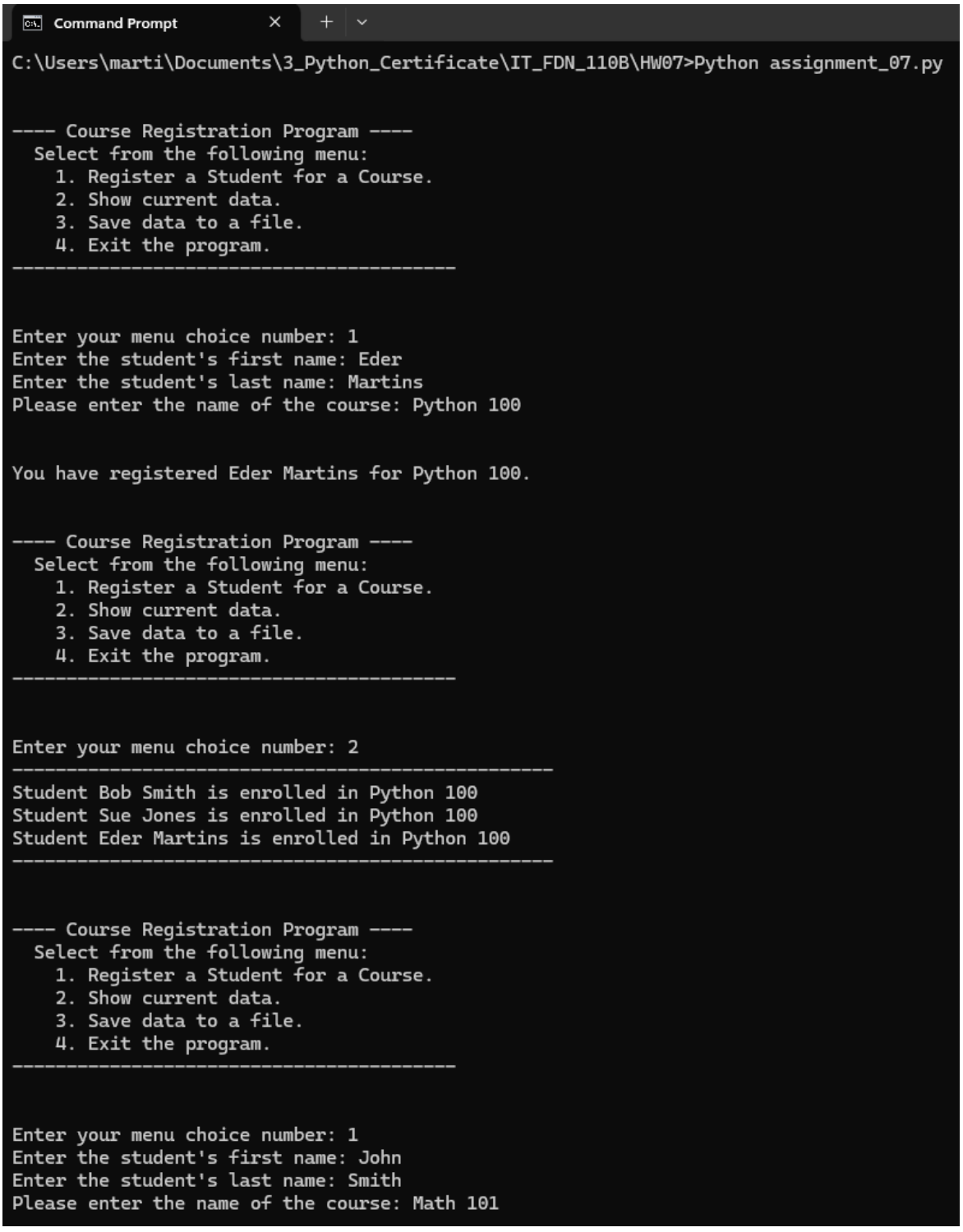

Figure 5 Executing Python Script in the Command Prompt

Figure 5 Executing Python Script in the Command Prompt (contnd)

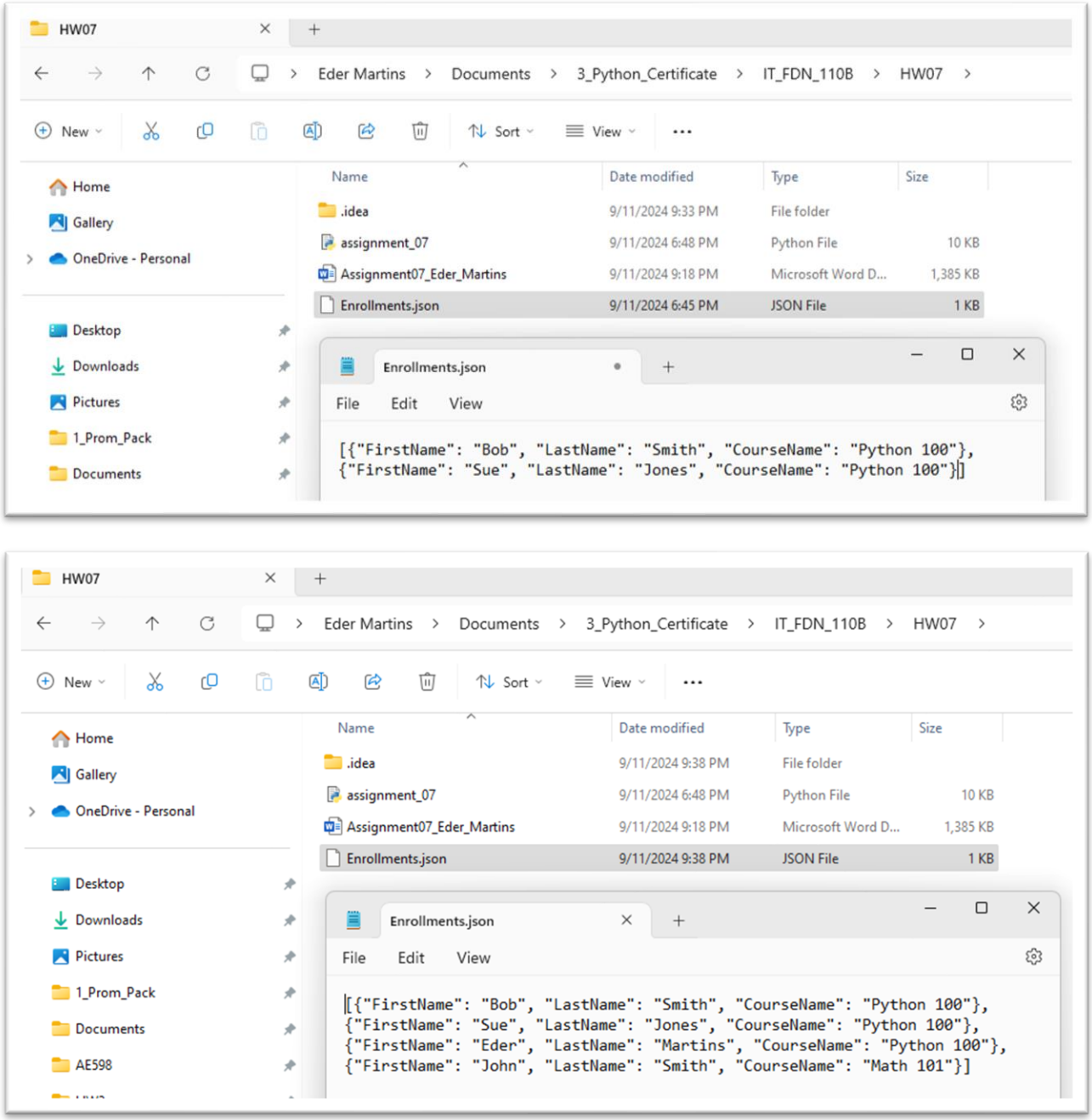

Figure 6 Enrollments.json file content before and after running the script

Summary

To complete Assignment 07, I followed a structured approach that integrated both new and previously learned concepts. The primary focus was on effectively utilizing data classes and implementing structured error handling, which enhanced the organization and reliability of the program. I learned how to define and use constants, variables, functions, and classes to manage data collection, validation, and file handling. Key new concepts included organizing the code using object-oriented principles with classes, validating user input for both names and course data, and handling potential errors (such as file access issues) in a structured way to ensure smooth program execution. Additionally, I applied earlier concepts like conditional logic, loops, and comparison operators to build a functional menu-driven program. By following a systematic approach, I created a Python program that met all assignment requirements, including displaying a menu with options to register students, show current data, save data to a file, and exit the program. The program ensured data integrity by validating user inputs for alphabetic names, utilized dictionaries to store student records, and managed data persistence by reading and writing to JSON files. Throughout the assignment, I enhanced my understanding of using functions, classes, and error handling to write cleaner, more modular code. I tested the program in both PyCharm and the command prompt to ensure it performed as expected, successfully saving and retrieving data. This project significantly improved my ability to create user-friendly, data-driven applications by focusing on principles like separation of concerns, data validation, and error handling.

Reference

[1] Arya, Anubhaw, *Module 07 Videos*. Available at YouTube: www.youtube.com/@arya0-uw