Renato Felicio
November 24, 2024
IT FDN 110 A
Assignment 07

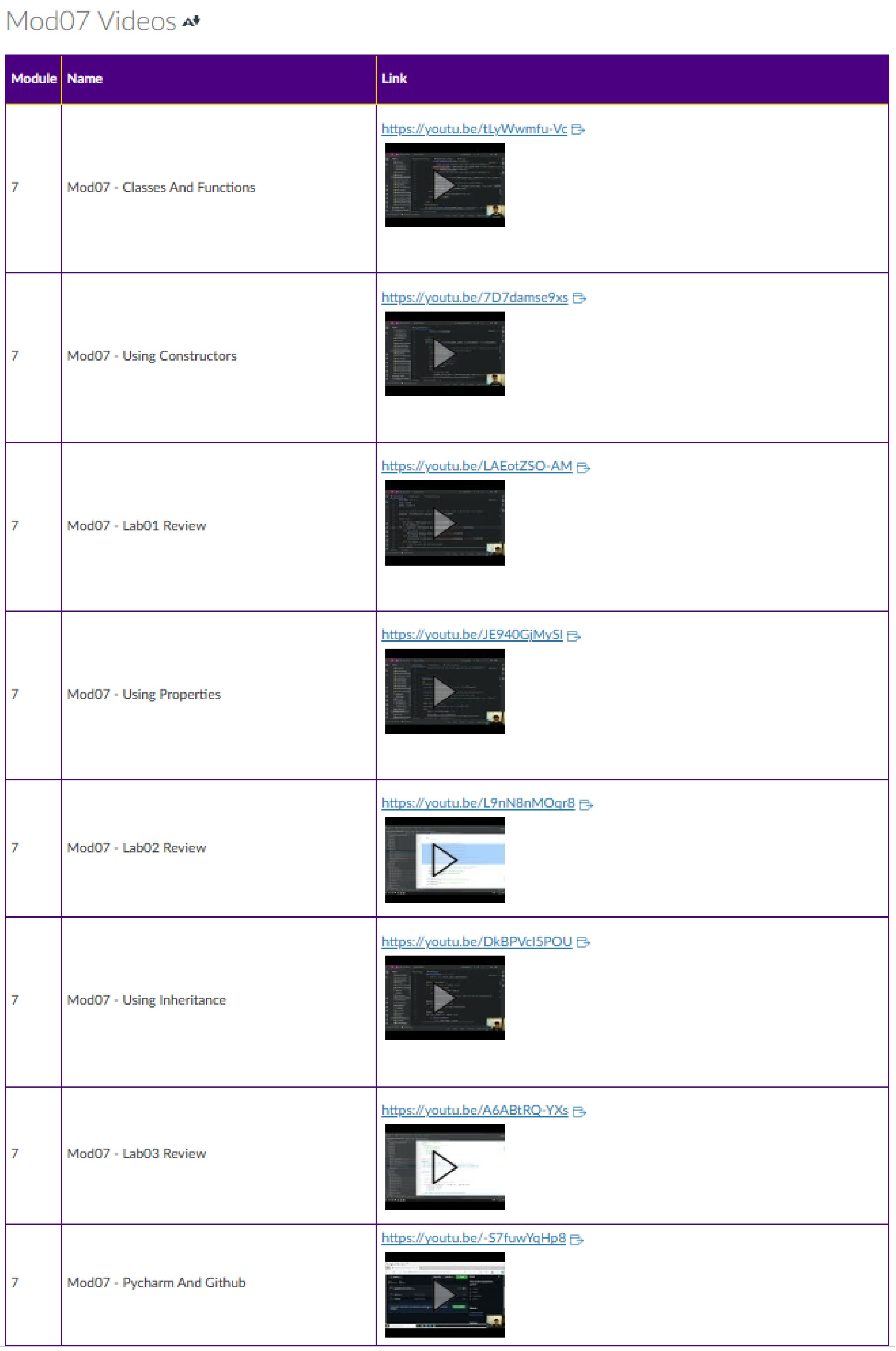
Creating Python Scripts – Classes and Objects

Introduction

This Assignment 07 involves creating a Python program that utilizes constants, variables, and print statements to display a message about a student's registration for a Python course. Building upon the knowledge gained in Assignment 06, this task introduces new concepts, including the use of data classes.

Preparation for this assignment

To prepare for this assignment, I reviewed the "Module 07 Notes" (Reference 1), completed the three lab examples, and watched both the Module 07 videos available on Washington University's Canvas platform (Figure 1) and the recommended external video in Reference 2. Through these materials, I learnt how to work with data classes, including their attributes, constructors, properties and methods. I also gained an understanding of about the Inheritance concept, where a child class inherit data value, properties and methods from its parent class, and how to override inherited methods.



Mod07 Videos

Module	Name	Link
7	Mod07 - Classes And Functions	https://youtu.be/tLyWwmfu-Vc
7	Mod07 - Using Constructors	https://youtu.be/7D7damse9xs
7	Mod07 - Lab01 Review	https://youtu.be/LAEotZSO-AM
7	Mod07 - Using Properties	https://youtu.be/JE940GjMySI
7	Mod07 - Lab02 Review	https://youtu.be/L9nN8nMOqr8
7	Mod07 - Using Inheritance	https://youtu.be/DkBPVcl5POU
7	Mod07 - Lab03 Review	https://youtu.be/A6ABtRQ-YXs
7	Mod07 - Pycharm And Github	https://youtu.be/-S7fuwYqHp8

Figure 1 – Mod07-Videos

Python Scripting

I started by using my script, *Assignment06.py*, from the previous assignment as the foundation for this project. The objective was to implement a set of data classes and objects. First, I created a Person class, which included private attributes for the first and last names, along with "getter" and "setter" properties and error handling. Next, I developed a child class called Student, which inherited the attributes of the Person class and introduced an additional attribute for the course name, complete with its own "getter" and "setter" properties.

After defining these two classes, I updated the methods in the FileProcessor and IO classes to work with objects instead of dictionaries. Finally, I modified the main body of the script to utilize objects, ensuring seamless integration of the new data class structure.

The functions (methods) are organized into four classes: Person, Student, FileProcessor, and IO. The FileProcessor class contains functions for reading data from and writing data to a .json file. The Person and Student classes define the data objects, including the student's first name, last name, and course name. The IO class manages user interactions, such as presenting menu options, gathering input data, displaying messages, and handling error notifications.

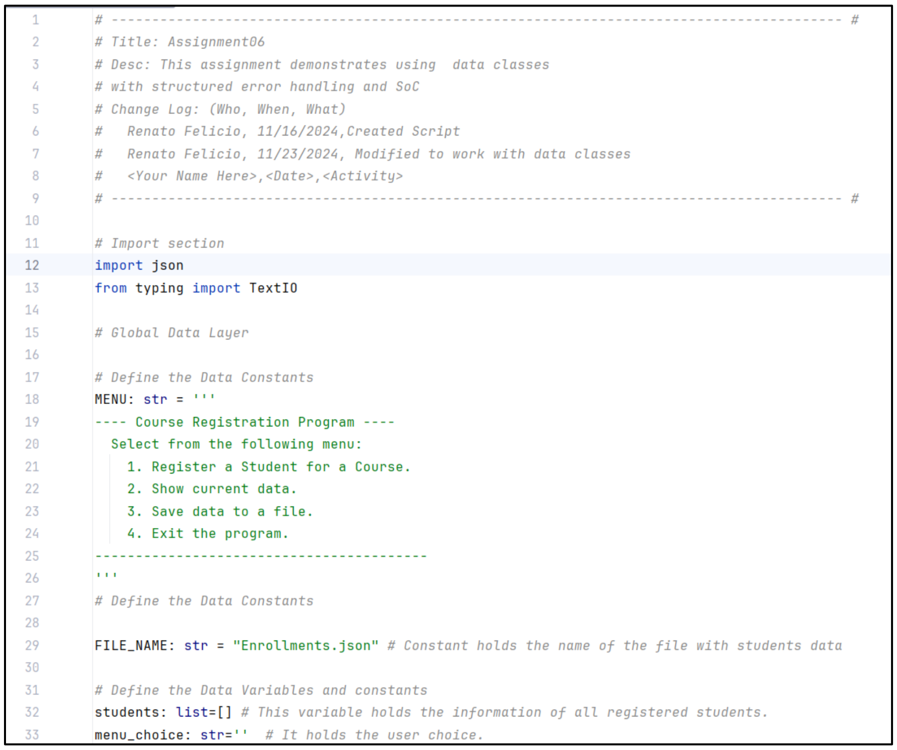
The script retains the two global variables defined in the Assignment 06 script: menu_choice, a string used to store the user's selected option, and students, a list used to hold the information of all registered students. These variables are initialized as an empty string and an empty list, respectively.

The script is structured into key sections: the header, imports, variable and constant definitions, class and function definitions, and the main body. Functions are grouped into classes based on their roles, ensuring alignment with the data processing and presentation layers for better organization and clarity.

```
# Header
# Import
# Global Data
# Data Layer
        Definition of data constants
        Definition of variable
# Class Definition
        Data Class
                Class Person
                Class Student
        Processing data layer
                Class FileProcessor
                        Function read_data_from_file created
                        Function write_data_to_file created
        Presentation data layer
                Class IO
                        Function output_error_messages created
                        Function output_menu
                        Function input_menu_choice
                        Function output_student_courses
                        Function input_student_data

# Main body of the script
```

The script is displayed in separated figures, split across Figure 2 to Figure 6 according its corresponde nt sections.

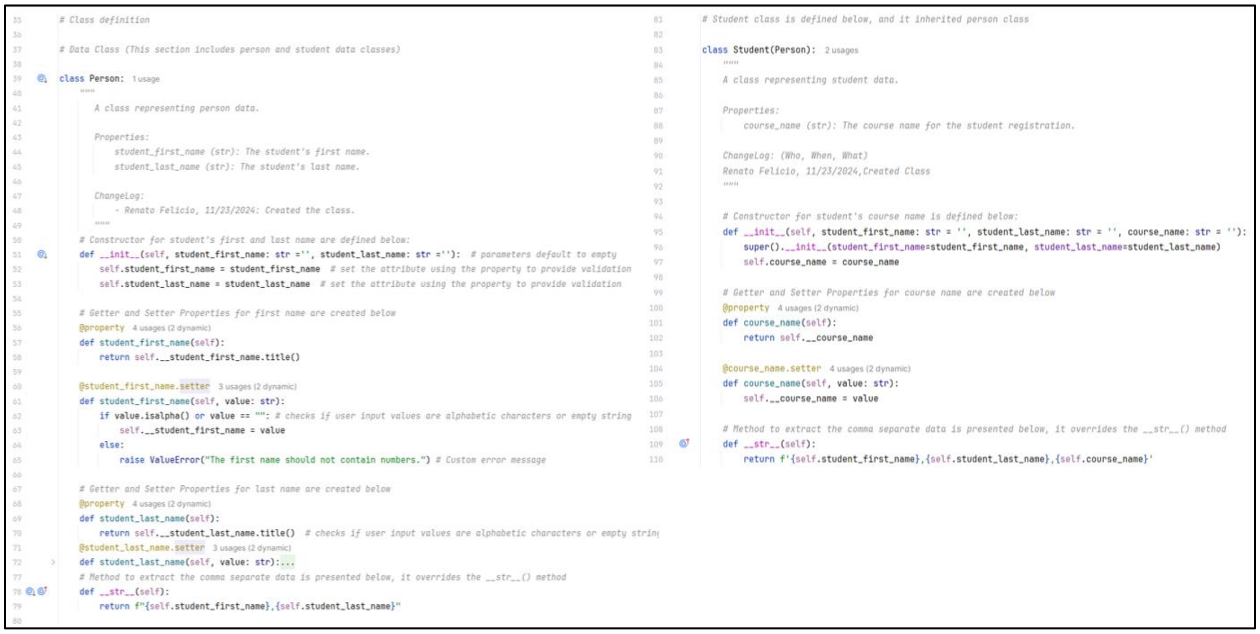
Header, import, constants and variables script parts are presented in the Figure 2.



```python
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06
# Desc: This assignment demonstrates using  data classes
# with structured error handling and SoC
# Change Log: (Who, When, What)
#   Renato Felicio, 11/16/2024,Created Script
#   Renato Felicio, 11/23/2024, Modified to work with data classes
#   <Your Name Here>,<Date>,<Activity>
# ------------------------------------------------------------------------------------------ #

# Import section
import json
from typing import TextIO

# Global Data Layer

# 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.
-----------------------------------------
'''
# Define the Data Constants

FILE_NAME: str = "Enrollments.json" # Constant holds the name of the file with students data

# Define the Data Variables and constants
students: list=[] # This variable holds the information of all registered students.
menu_choice: str=''  # It holds the user choice.
```

Figure 2 – Python Script Header Import and Variables

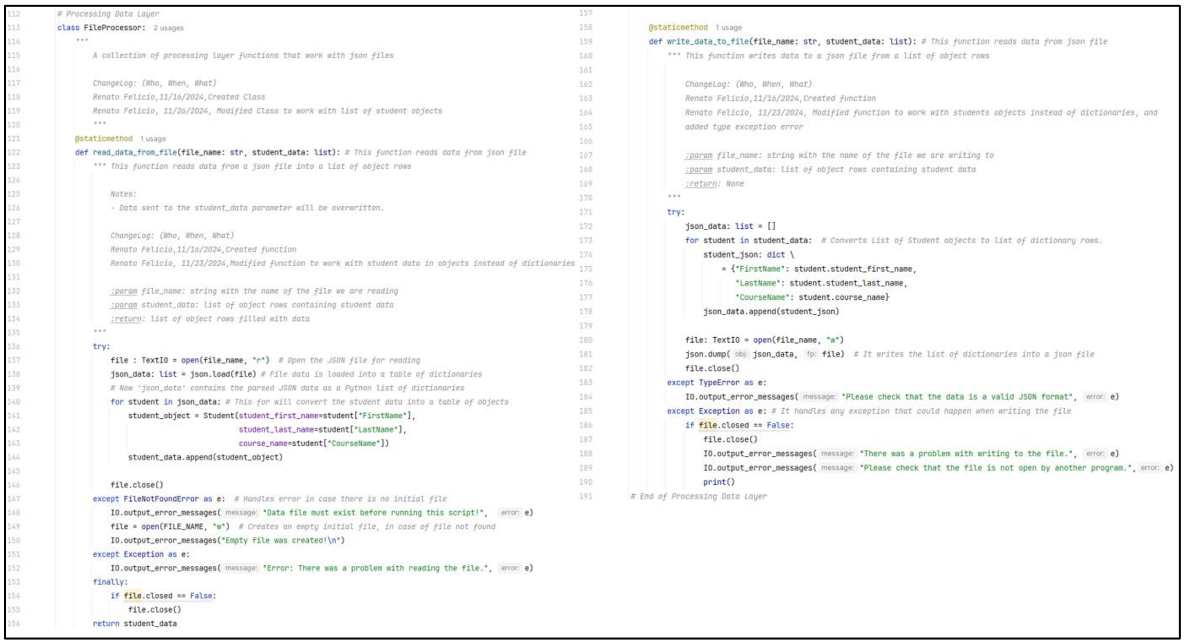
Data classes are presented in Figure 3.



```python
# Class definition

# Data Class (This section includes person and student data classes)

class Person:  1 usage
    """
        A class representing person data.

        Properties:
            student_first_name (str): The student's first name.
            student_last_name (str): The student's last name.

        ChangeLog:
            - Renato Felicio, 11/23/2024: Created the class.
        """
    # Constructor for student's first and last name are defined below:
    def __init__(self, student_first_name: str ='', student_last_name: str =''):  # parameters default to empty
        self.student_first_name = student_first_name  # set the attribute using the property to provide validation
        self.student_last_name = student_last_name  # set the attribute using the property to provide validation

    # Getter and Setter Properties for first name are created below
    @property  4 usages (2 dynamic)
    def student_first_name(self):
        return self.__student_first_name.title()

    @student_first_name.setter  3 usages (2 dynamic)
    def student_first_name(self, value: str):
        if value.isalpha() or value == "": # checks if user input values are alphabetic characters or empty string
            self.__student_first_name = value
        else:
            raise ValueError("The first name should not contain numbers.") # Custom error message

    # Getter and Setter Properties for last name are created below
    @property  4 usages (2 dynamic)
    def student_last_name(self):
        return self.__student_last_name.title()  # checks if user input values are alphabetic characters or empty strin
    @student_last_name.setter  3 usages (2 dynamic)
    def student_last_name(self, value: str):...
    # Method to extract the comma separate data is presented below, it overrides the __str__() method
    def __str__(self):
        return f"{self.student_first_name},{self.student_last_name}"

# Student class is defined below, and it inherited person class

class Student(Person):  2 usages
    """
    A class representing student data.

    Properties:
        course_name (str): The course name for the student registration.

    ChangeLog: (Who, When, What)
    Renato Felicio, 11/23/2024,Created Class
    """

    # Constructor for student's course name is defined below:
    def __init__(self, student_first_name: str = '', student_last_name: str = '', course_name: str = ''):
        super().__init__(student_first_name=student_first_name, student_last_name=student_last_name)
        self.course_name = course_name

    # Getter and Setter Properties for course name are created below
    @property  4 usages (2 dynamic)
    def course_name(self):
        return self.__course_name

    @course_name.setter  4 usages (2 dynamic)
    def course_name(self, value: str):
        self.__course_name = value

    # Method to extract the comma separate data is presented below, it overrides the __str__() method
    def __str__(self):
        return f'{self.student_first_name},{self.student_last_name},{self.course_name}'
```

Figure 3 – Python Script Data Layer

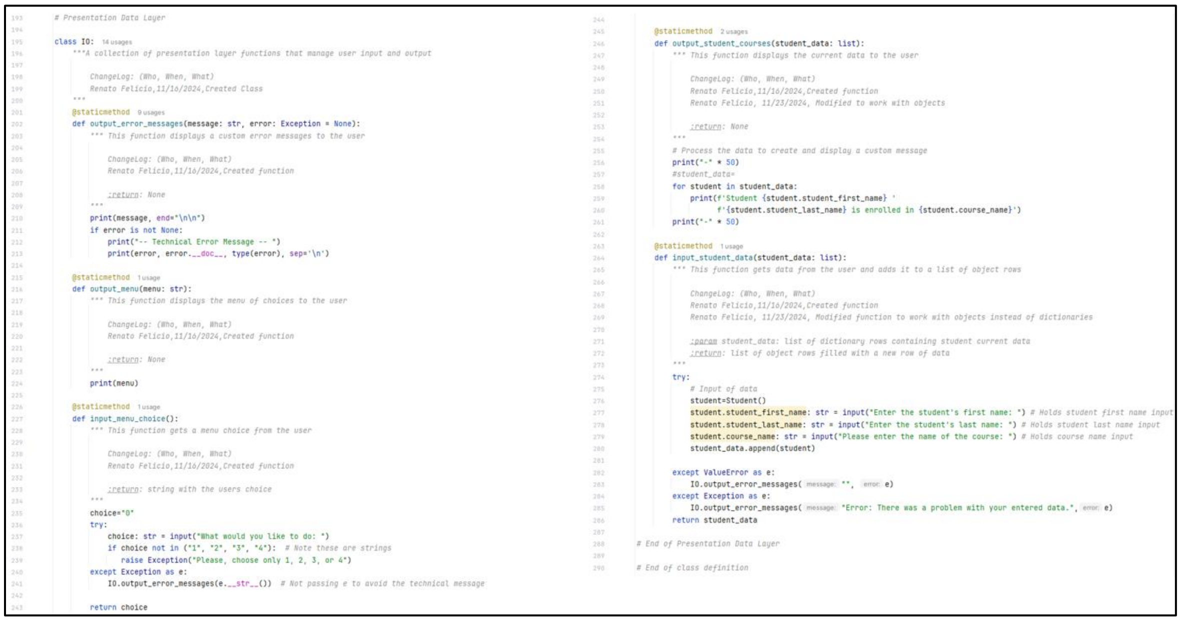
Processing data layer is presented in Figure 4.



```python
# Processing Data Layer
class FileProcessor:  2 usages
    """
        A collection of processing layer functions that work with json files

        ChangeLog: (Who, When, What)
        Renato Felicio,11/16/2024,Created Class
        Renato Felicio, 11/26/2024, Modified Class to work with list of student objects
        """
    @staticmethod  1 usage
    def read_data_from_file(file_name: str, student_data: list): # This function reads data from json file
        """ This function reads data from a json file into a list of object rows

            Notes:
            - Data sent to the student_data parameter will be overwritten.

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,Created function
            Renato Felicio, 11/23/2024,Modified function to work with student data in objects instead of dictionaries

            :param file_name: string with the name of the file we are reading
            :param student_data: list of object rows containing student data
            :return: list of object rows filled with data
        """
        try:
            file : TextIO = open(file_name, "r")  # Open the JSON file for reading
            json_data: list = json.load(file) # File data is loaded into a table of dictionaries
            # Now 'json_data' contains the parsed JSON data as a Python list of dictionaries
            for student in json_data: # This for will convert the student data into a table of objects
                student_object = Student(student_first_name=student["FirstName"],
                                         student_last_name=student["LastName"],
                                         course_name=student["CourseName"])
                student_data.append(student_object)

            file.close()
        except FileNotFoundError as e:  # Handles error in case there is no initial file
            IO.output_error_messages( message: "Data file must exist before running this script!",  error: e)
            file = open(FILE_NAME, "w")  # Creates an empty initial file, in case of file not found
            IO.output_error_messages("Empty file was created!\n")
        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

    @staticmethod  1 usage
    def write_data_to_file(file_name: str, student_data: list): # This function reads data from json file
        """ This function writes data to a json file from a list of object rows

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,Created function
            Renato Felicio, 11/23/2024, Modified function to work with students objects instead of dictionaries, and
            added type exception error

            :param file_name: string with the name of the file we are writing to
            :param student_data: list of object rows containing student data
            :return: None
        """
        try:
            json_data: list = []
            for student in student_data:  # Converts List of Student objects to list of dictionary rows.
                student_json: dict \
                    = {"FirstName": student.student_first_name,
                       "LastName": student.student_last_name,
                       "CourseName": student.course_name}
                json_data.append(student_json)

            file: TextIO = open(file_name, "w")
            json.dump( obj: json_data,  fp: file)  # It writes the list of dictionaries into a json file
            file.close()
        except TypeError as e:
            IO.output_error_messages( message: "Please check that the data is a valid JSON format",  error: e)
        except Exception as e: # It handles any exception that could happen when writing the file
            if file.closed == False:
                file.close()
                IO.output_error_messages( message: "There was a problem with writing to the file.",  error: e)
                IO.output_error_messages( message: "Please check that the file is not open by another program.", error: e)
                print()
# End of Processing Data Layer
```

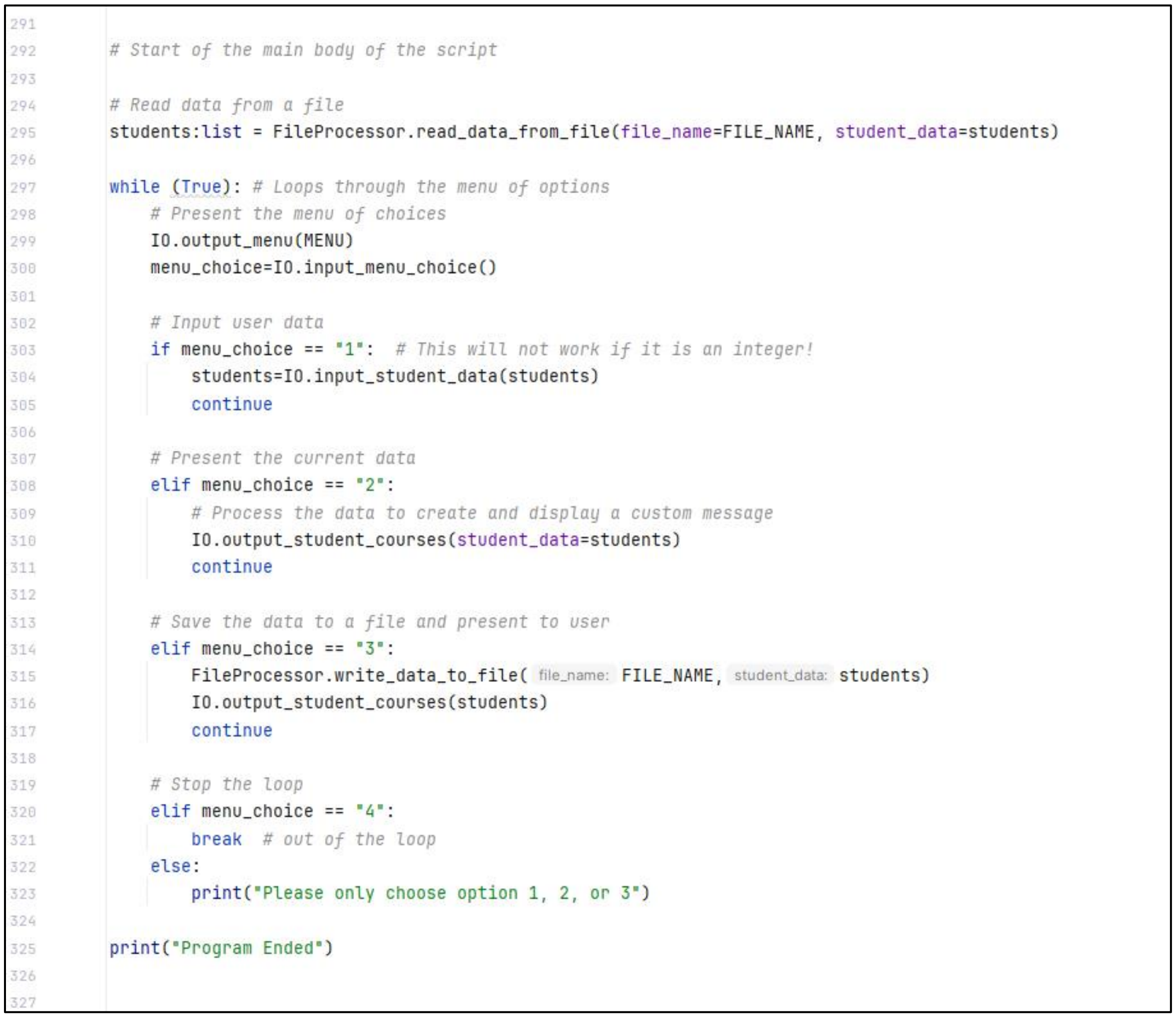
Figure 4 – Python Script Processing Data Layer

Presentation data layer and main body are presented in Figure 5 and Figure 6, respectively.



```python
# Presentation Data Layer

class IO:  14 usages
    """A collection of presentation layer functions that manage user input and output

        ChangeLog: (Who, When, What)
        Renato Felicio,11/16/2024,Created Class
    """
    @staticmethod  9 usages
    def output_error_messages(message: str, error: Exception = None):
        """ This function displays a custom error messages to the user

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,Created function

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

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

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,Created function

            :return: None
        """
        print(menu)

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

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,Created function

            :return: string with the users choice
        """
        choice="0"
        try:
            choice: str = input("What would you like to do: ")
            if choice not in ("1", "2", "3", "4"):  # Note these are strings
                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


    @staticmethod  2 usages
    def output_student_courses(student_data: list):
        """ This function displays the current data to the user

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,Created function
            Renato Felicio, 11/23/2024, Modified to work with objects

            :return: None
        """
        # Process the data to create and display a custom message
        print("-" * 50)
        #student_data=
        for student in student_data:
            print(f'Student {student.student_first_name} '
                  f'{student.student_last_name} is enrolled in {student.course_name}')
        print("-" * 50)

    @staticmethod  1 usage
    def input_student_data(student_data: list):
        """ This function gets data from the user and adds it to a list of object rows

            ChangeLog: (Who, When, What)
            Renato Felicio,11/16/2024,Created function
            Renato Felicio, 11/23/2024, Modified function to work with objects instead of dictionaries

            :param student_data: list of dictionary rows containing student current data
            :return: list of object rows filled with a new row of data
        """
        try:
            # Input of data
            student=Student()
            student.student_first_name: str = input("Enter the student's first name: ") # Holds student first name input
            student.student_last_name: str = input("Enter the student's last name: ") # Holds student last name input
            student.course_name: str = input("Please enter the name of the course: ") # Holds course name input
            student_data.append(student)

        except ValueError as e:
            IO.output_error_messages( message: "",  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

# End of Presentation Data Layer

# End of class definition
```

Figure 5 – Python Script Presentation Data Layer

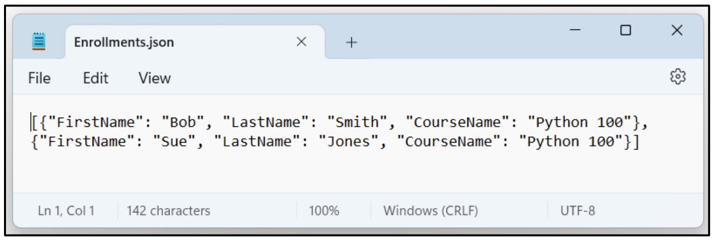


```python

# Start of the main body of the script

# Read data from a file
students:list = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

while (True): # Loops through the menu of options
    # Present the menu of choices
    IO.output_menu(MENU)
    menu_choice=IO.input_menu_choice()

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

    # Present the current data
    elif menu_choice == "2":
        # Process the data to create and display a custom message
        IO.output_student_courses(student_data=students)
        continue

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

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

print("Program Ended")

```

Figure 6 – Python Script Main Body

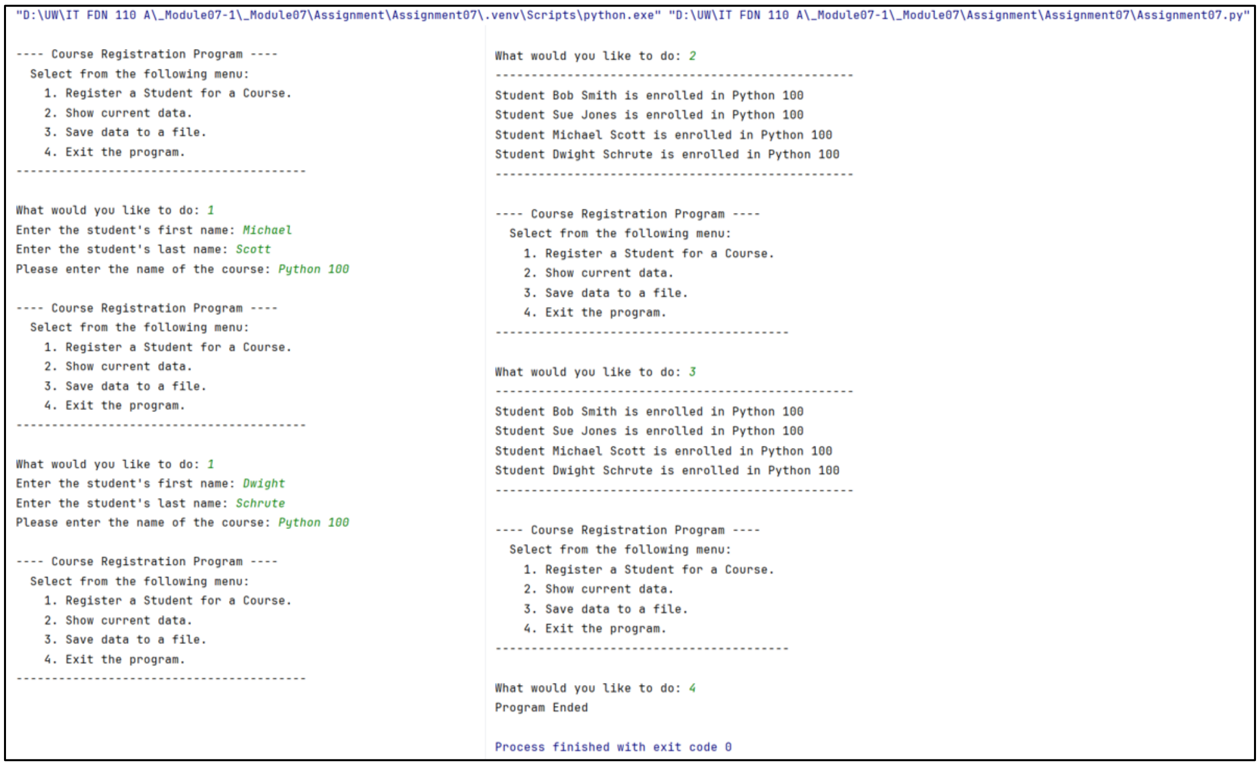
Python Script Testing

I executed the Python script using PyCharm (see Figure 8) and also tested it in the Windows Command Prompt (see Figure 9), verifying that the script performed as expected in both environments. Additionally, I confirmed that the Enrollments.json file was updated correctly and contained the expected output (see Figure 10). Following the assignment instructions, I used a initial Enrollments.json. The original content of this file is shown in Figure 7.



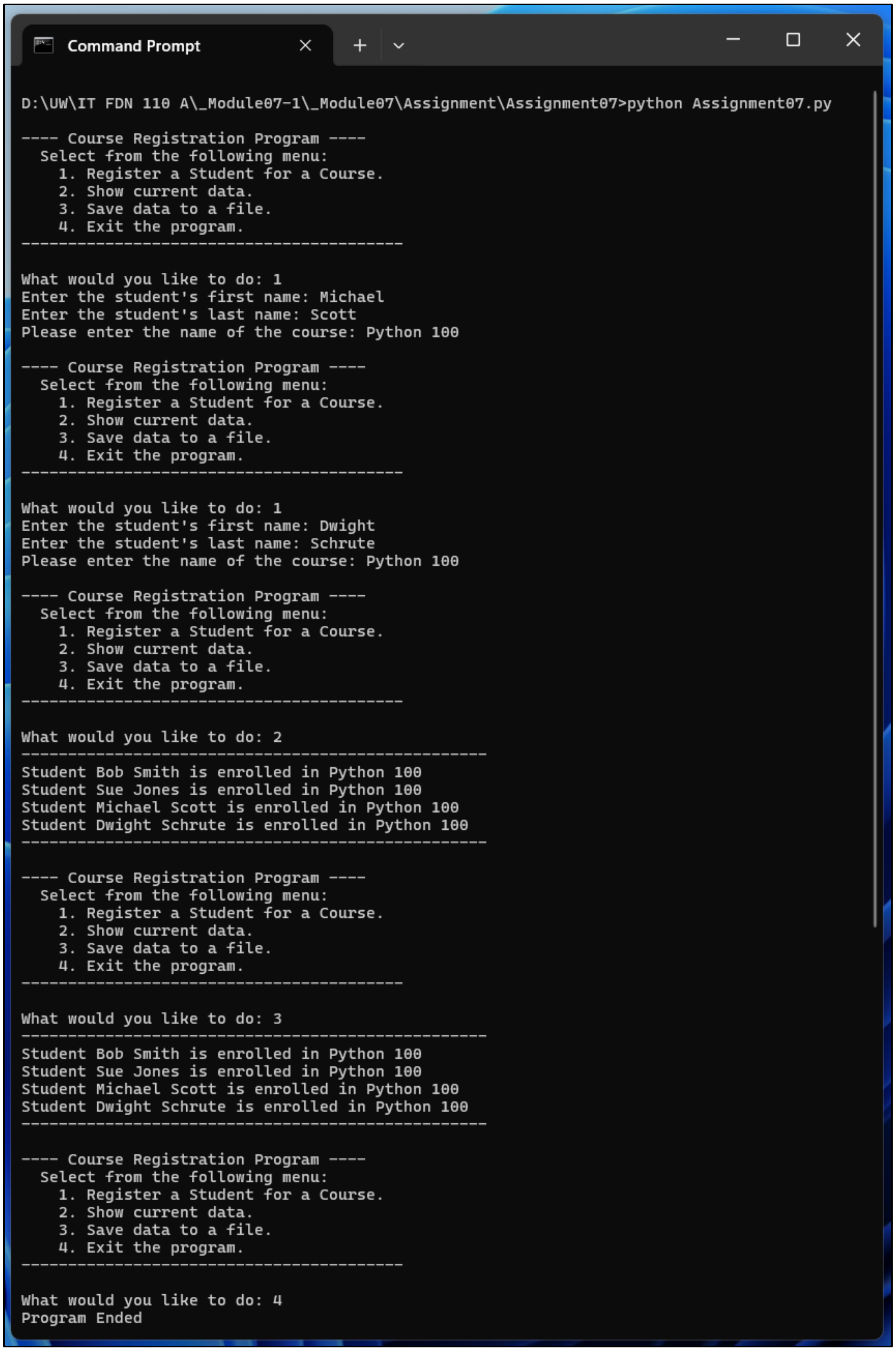
```
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"},
{"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"}]
```

Figure 7 – Initial Enrollment File Python



```
"D:\UW\IT FDN 110 A\_Module07-1\_Module07\Assignment\Assignment07\.venv\Scripts\python.exe" "D:\UW\IT FDN 110 A\_Module07-1\_Module07\Assignment\Assignment07\Assignment07.py"

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

What would you like to do: 1
Enter the student's first name: Michael
Enter the student's last name: Scott
Please enter the name of the course: Python 100

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

What would you like to do: 1
Enter the student's first name: Dwight
Enter the student's last name: Schrute
Please enter the name of the course: Python 100

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

What would you like to do: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
--------------------------------------------------

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

What would you like to do: 3
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
--------------------------------------------------

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

What would you like to do: 4
Program Ended

Process finished with exit code 0
```

Figure 8 – Python Script PyCharm Run

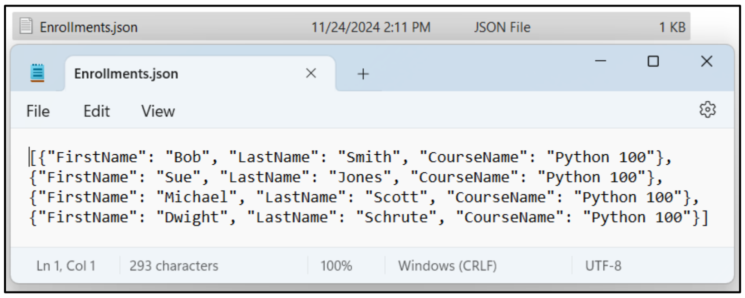


Command Prompt

```
D:\UW\IT FDN 110 A\_Module07-1\_Module07\Assignment\Assignment07>python Assignment07.py

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

What would you like to do: 1
Enter the student's first name: Michael
Enter the student's last name: Scott
Please enter the name of the course: Python 100

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

What would you like to do: 1
Enter the student's first name: Dwight
Enter the student's last name: Schrute
Please enter the name of the course: Python 100

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

What would you like to do: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
--------------------------------------------------

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

What would you like to do: 3
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
--------------------------------------------------

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

What would you like to do: 4
Program Ended
```

Figure 9 – Python Script Windows Command Prompt Run

Figure 10 shows that the Enrollments.json file was updated correctly with the two new student data.



Enrollments.json 11/24/2024 2:11 PM JSON File 1 KB

```
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"},
{"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"},
{"FirstName": "Michael", "LastName": "Scott", "CourseName": "Python 100"},
{"FirstName": "Dwight", "LastName": "Schrute", "CourseName": "Python 100"}]
```

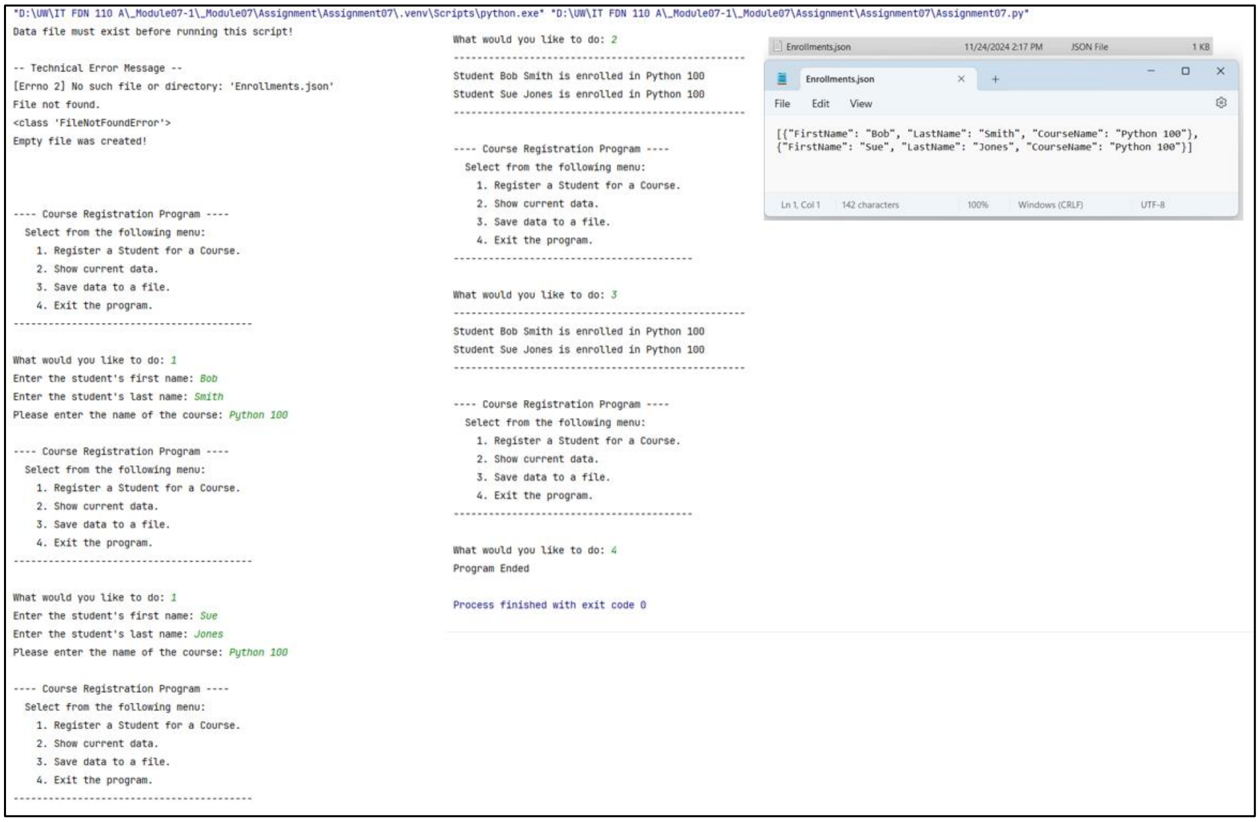
Ln 1, Col 1 | 293 characters | 100% | Windows (CRLF) | UTF-8

Figure 10 – Output JSON File Content

Python Script Error Handling Test

After testing the script with valid user inputs using a pre-existing Enrollments.json file, I evaluated its error-handling capabilities for scenarios such as a "file not found" exception and invalid first and last name inputs. All tests were conducted using PyCharm.

The Figure 11 below shows the FileNotFound handling and the Enrolments.json being updated correctly after menu choice 3 is selected.



```
"D:\UW\IT FDN 110 A\_Module07-1\_Module07\Assignment\Assignment07\.venv\Scripts\python.exe" "D:\UW\IT FDN 110 A\_Module07-1\_Module07\Assignment\Assignment07\Assignment07.py"
Data file must exist before running this script!

-- Technical Error Message --
[Errno 2] No such file or directory: 'Enrollments.json'
File not found.
<class 'FileNotFoundError'>
Empty file was created!



---- 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.
-----------------------------------------

What would you like to do: 1
Enter the student's first name: Bob
Enter the student's last name: Smith
Please enter the name of the course: Python 100

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

What would you like to do: 1
Enter the student's first name: Sue
Enter the student's last name: Jones
Please enter the name of the course: Python 100

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------
What would you like to do: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
--------------------------------------------------

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

What would you like to do: 3
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
--------------------------------------------------

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

What would you like to do: 4
Program Ended

Process finished with exit code 0
```

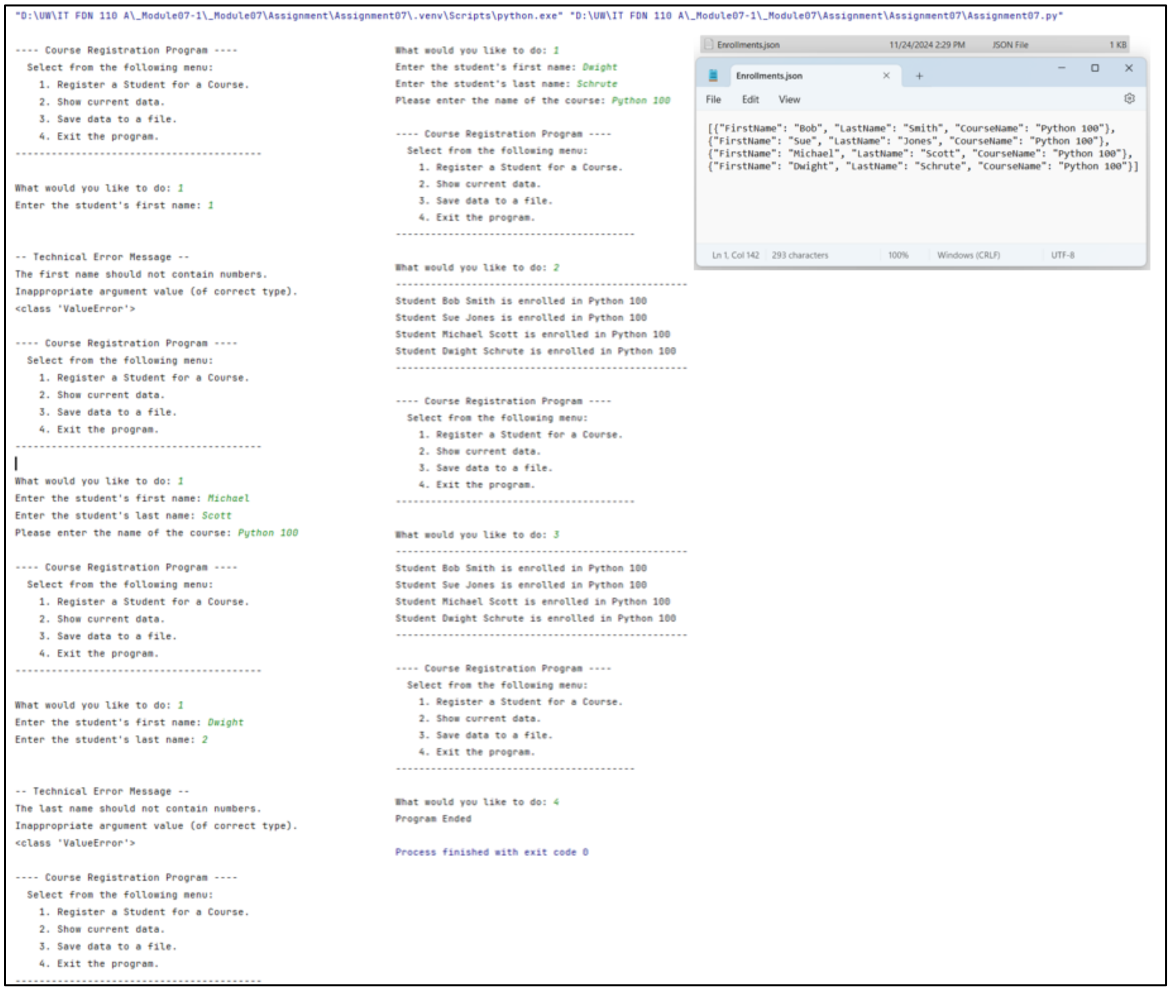
Enrollments.json 11/24/2024 2:17 PM JSON File 1 KB

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

Ln 1, Col 1 | 142 characters | 100% | Windows (CRLF) | UTF-8

Figure 11 – FileNotFoundError Handling

The second error handling test involved checking the condition where the user enters non-alphabetic characters for the first or last names (see Figure 12). The Enrollments.json file was also correctly updated



```
"D:\UW\IT FDN 110 A\_Module07-1\_Module07\Assignment\Assignment07\.venv\Scripts\python.exe" "D:\UW\IT FDN 110 A\_Module07-1\_Module07\Assignment\Assignment07\Assignment07.py"

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

What would you like to do: 1
Enter the student's first name: 1

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

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

What would you like to do: 1
Enter the student's first name: Michael
Enter the student's last name: Scott
Please enter the name of the course: Python 100

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

What would you like to do: 1
Enter the student's first name: Dwight
Enter the student's last name: 2

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

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------
What would you like to do: 1
Enter the student's first name: Dwight
Enter the student's last name: Schrute
Please enter the name of the course: Python 100

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

What would you like to do: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
--------------------------------------------------

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

What would you like to do: 3
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Michael Scott is enrolled in Python 100
Student Dwight Schrute is enrolled in Python 100
--------------------------------------------------

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

What would you like to do: 4
Program Ended

Process finished with exit code 0
```

Enrollments.json 11/24/2024 2:29 PM JSON File 1 KB
Enrollments.json
File Edit View
Ln 1, Col 142 293 characters 100% Windows (CRLF) UTF-8

```
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"},
{"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"},
{"FirstName": "Michael", "LastName": "Scott", "CourseName": "Python 100"},
{"FirstName": "Dwight", "LastName": "Schrute", "CourseName": "Python 100"}]
```

Figure 12 – Non-Alphabetic Characters Error Handling

GitHub

Script and documentation for this assignment is available in my GitHub site:

https://github.com/rfnaval/IntroToProg-Python-Mod07.git

Summary

This assignment built upon the previous one, providing an opportunity to learn and practice key Python concepts such as data classes, their private attributes, constructors, “self” keyword, “getter” and “setter” properties, the inheritance concept, and overriding methods. It was very interesting to work with tables (list of lists) containing objects instead of dictionaries, and to modify the FileProcessor and IO classes, along with the script main body to work with objects as well.

References

1. Module 07 - Classes and Objects, Randal Root, January 02, 2024.
2. External site: Python OOP Tutorial 1: Classes and Instances, Corey Schafer.