Jeff Howard

11/19/2024

Foundations Of Programming: Python

Assignment 06

GitHub

Functions

Introduction

In this paper, I will review the steps I took to create a program in Python that presents a menu of options for a user to select, and depending on the user's selection, will either gather their data, present the data back, save the data to a file, or exit the program. This program builds on our 5th assignment by using parameters, functions, classes and separation of concerns.

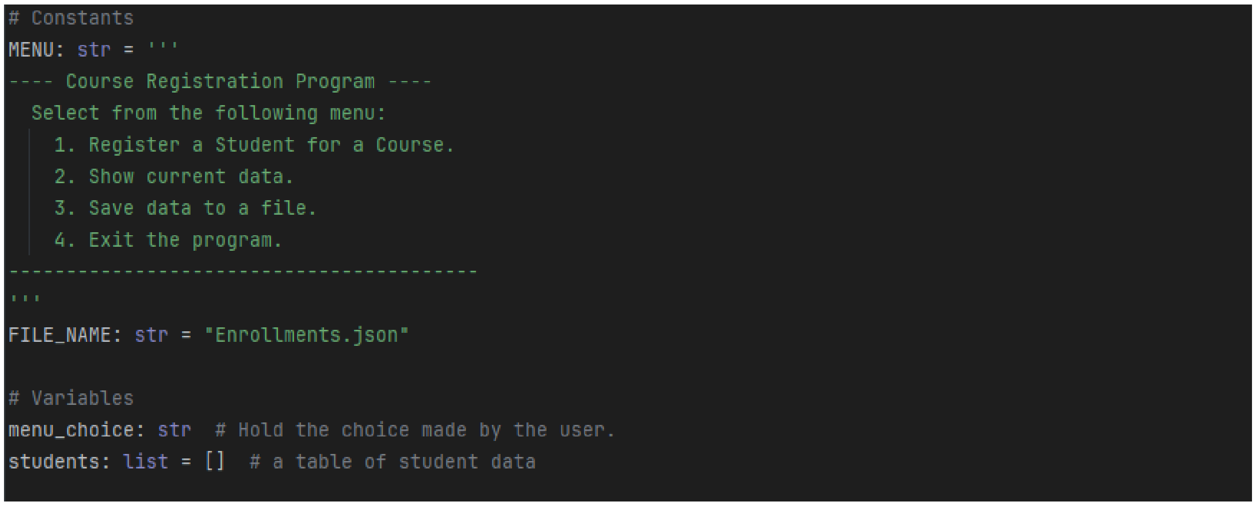
To improve readability, maintainability, and scalability, this code is formatted following the separation of concerns approach. The first category in the overall structure of the code is the Data Layer.

Topic 1: Data Layer

Constants and Variables

In this assignment, we wanted to present a menu of options for the user to choose from - to be used in our conditional logic later in the code. The first constant is the menu of options and the second constant is the JSON file name where we will be saving the data.

The variables for this project are user inputted values - which are the first and last name of the student, and the list of students. (Fig 1)



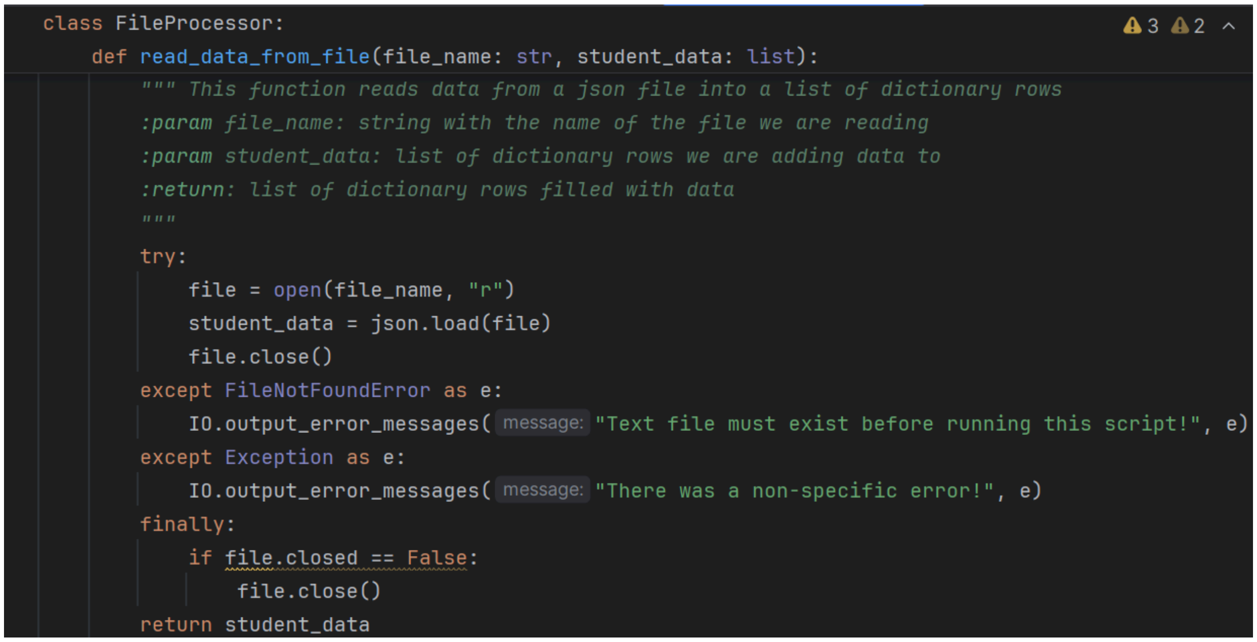
```python
# Constants
MENU: str = '''
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------
'''
FILE_NAME: str = "Enrollments.json"

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

Figure 1: Defined Constants & Variables

Topic 2: Class Definitions

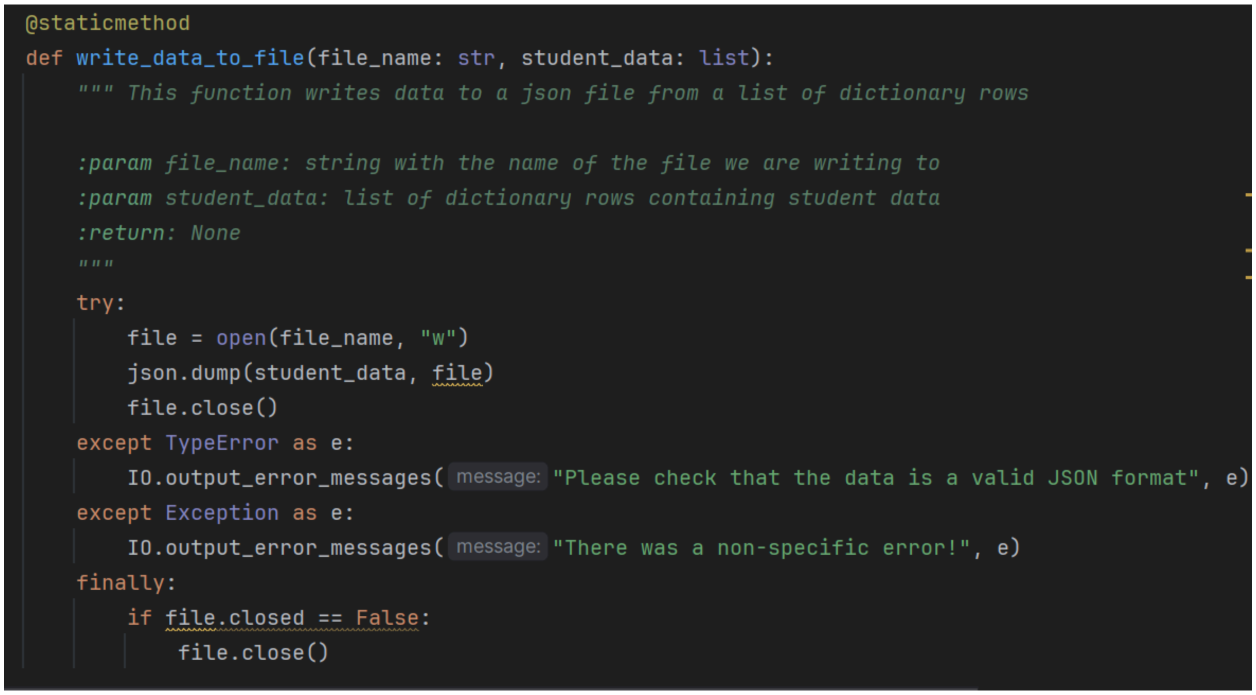
SubTopic 2a: File Processor Class

The file processing class (FileProcessor) has the code needed to read the json file and write our student data back to the json file. In these and all the other functions in the program, I first started by adding descriptive document strings. Then I used the @staticmethod to define functions read_data_from_file (fig 2a) and write_data_to_file (fig 2b) with the file name and student list as parameters.



```
class FileProcessor:
    def read_data_from_file(file_name: str, student_data: list):
        """ This function reads data from a json file into a list of dictionary rows
        :param file_name: string with the name of the file we are reading
        :param student_data: list of dictionary rows we are adding data to
        :return: list of dictionary rows filled with data
        """
        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            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 file.closed == False:
                file.close()
        return student_data
```

Figure 2a: Data processing: reading data function

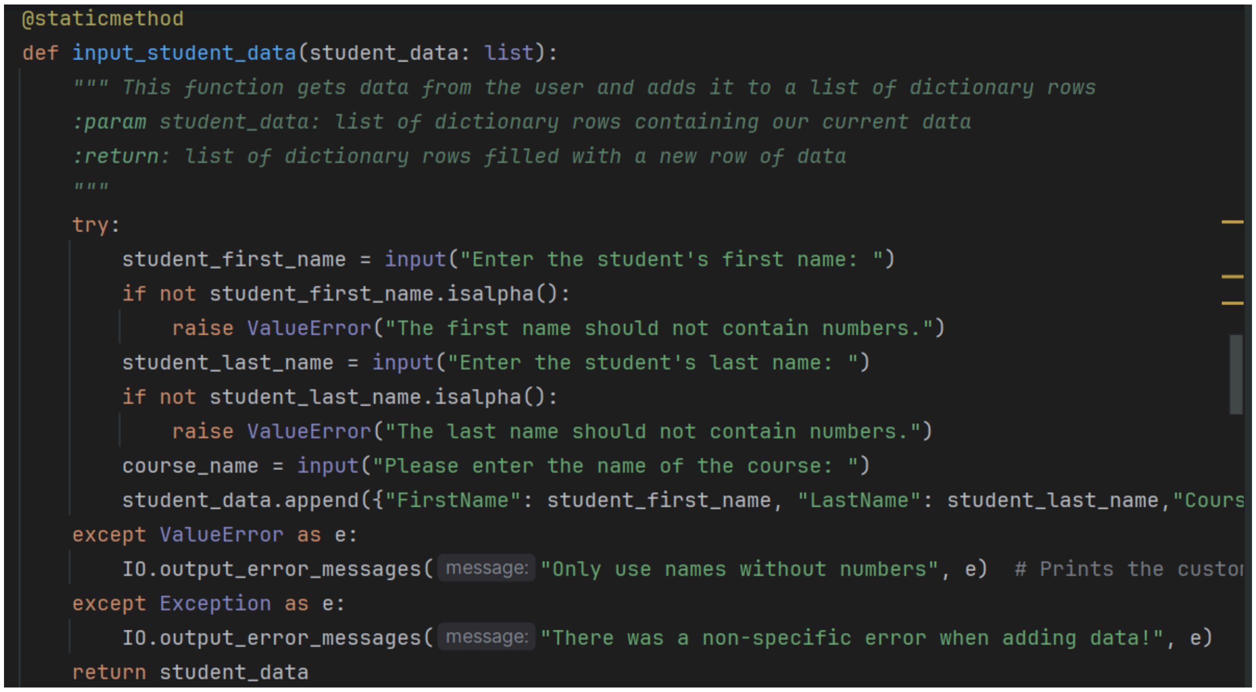


```python
@staticmethod
def write_data_to_file(file_name: str, student_data: list):
    """ This function writes data to a json file from a list of dictionary rows

    :param file_name: string with the name of the file we are writing to
    :param student_data: list of dictionary rows containing student data
    :return: None
    """
    try:
        file = open(file_name, "w")
        json.dump(student_data, file)
        file.close()
    except TypeError as e:
        IO.output_error_messages( message: "Please check that the data is a valid JSON format", e)
    except Exception as e:
        IO.output_error_messages( message: "There was a non-specific error!", e)
    finally:
        if file.closed == False:
            file.close()
```

Figure 2b: Data processing: writing data function

SubTopic 2b - Presentation (Input/Output) Class

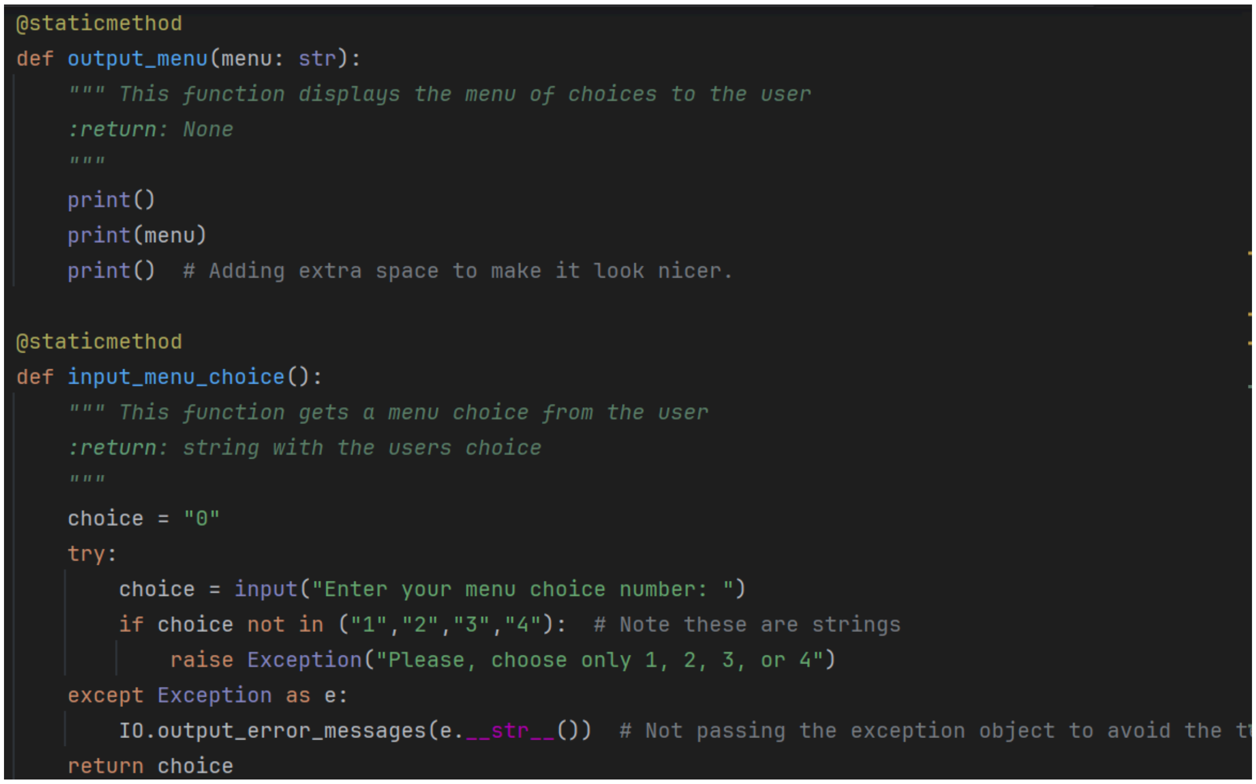
The first function defined in the IO class was input_student_data. If the user wants to register a student for a course by selecting option 1 in our menu, the program calls input_student_data to gather the user's first, last and course name and prints custom error messages if they occur. (Fig 3a)



```python
@staticmethod
def input_student_data(student_data: list):
    """ This function gets data from the user and adds it to a list of dictionary rows
    :param student_data: list of dictionary rows containing our current data
    :return: list of dictionary rows filled with a new row of data
    """
    try:
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("The first name should not contain numbers.")
        student_last_name = input("Enter the student's last name: ")
        if not student_last_name.isalpha():
            raise ValueError("The last name should not contain numbers.")
        course_name = input("Please enter the name of the course: ")
        student_data.append({"FirstName": student_first_name, "LastName": student_last_name,"Cours
    except ValueError as e:
        IO.output_error_messages( message: "Only use names without numbers", e)  # Prints the custo
    except Exception as e:
        IO.output_error_messages( message: "There was a non-specific error when adding data!", e)
    return student_data
```

Figure 3a: Input/Output: gathering user data

The next two functions present the menu of options to the user and gather their inputted selection. The input_menu_choice function also has custom error handling based on the user input. (Fig 2b)



```
@staticmethod
def output_menu(menu: str):
    """ This function displays the menu of choices to the user
    :return: None
    """
    print()
    print(menu)
    print()  # Adding extra space to make it look nicer.

@staticmethod
def input_menu_choice():
    """ This function gets a menu choice from the user
    :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("Please, choose only 1, 2, 3, or 4")
    except Exception as e:
        IO.output_error_messages(e.__str__())  # Not passing the exception object to avoid the t
    return choice
```

Figure 3b : Menu and User Selection

The next and final two functions in the class definitions section are outputs presented back to the user; output_error_messages & output_student_courses. Output_error_messages is used to print a custom error message to the user if they occur. Output_student_courses prints the current student data in the program. (Fig 3c)

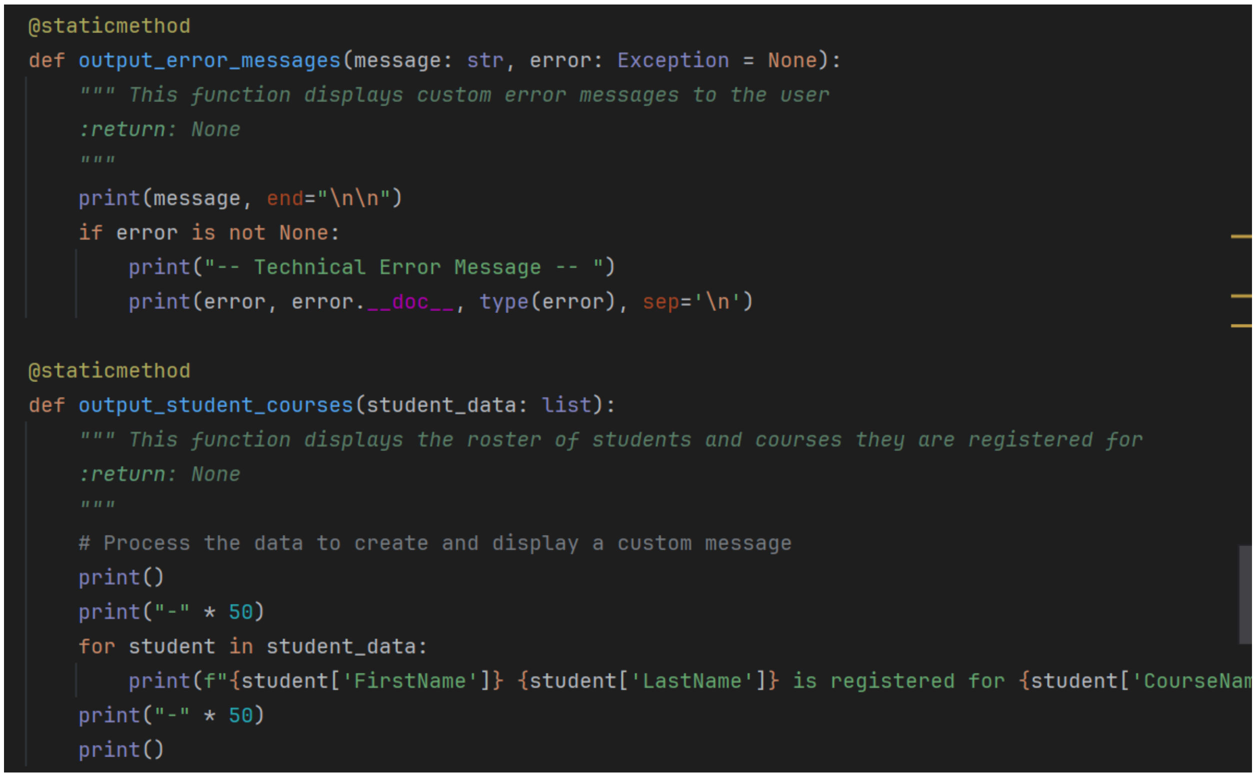
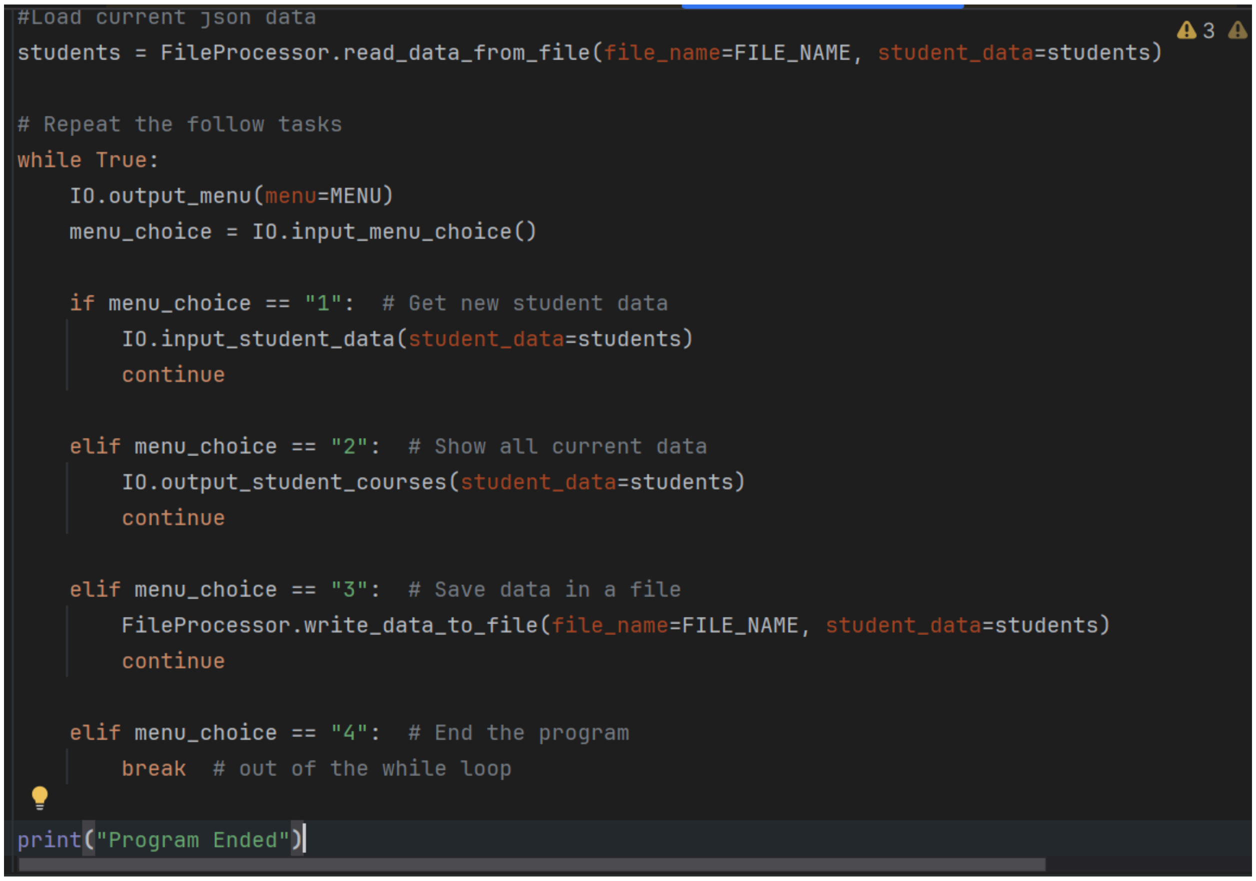

```
@staticmethod
def output_error_messages(message: str, error: Exception = None):
    """ This function displays custom error messages to the user
    :return: None
    """
    print(message, end="\n\n")
    if error is not None:
        print("-- Technical Error Message -- ")
        print(error, error.__doc__, type(error), sep='\n')

@staticmethod
def output_student_courses(student_data: list):
    """ This function displays the roster of students and courses they are registered for
    :return: None
    """
    # Process the data to create and display a custom message
    print()
    print("-" * 50)
    for student in student_data:
        print(f"{student['FirstName']} {student['LastName']} is registered for {student['CourseNam
    print("-" * 50)
    print()
```

Figure 3c : Outputs

Topic 3: Conditional Logic

After defining the class functions, the main body of our program uses a while loop to cycle through each condition of the menu. First, I used FileProcessor.read_data_from_file to open the json and read the current file data into the program. Next I entered the appropriate function for each condition of the if statement and based on the user's entry. (Fig 4)



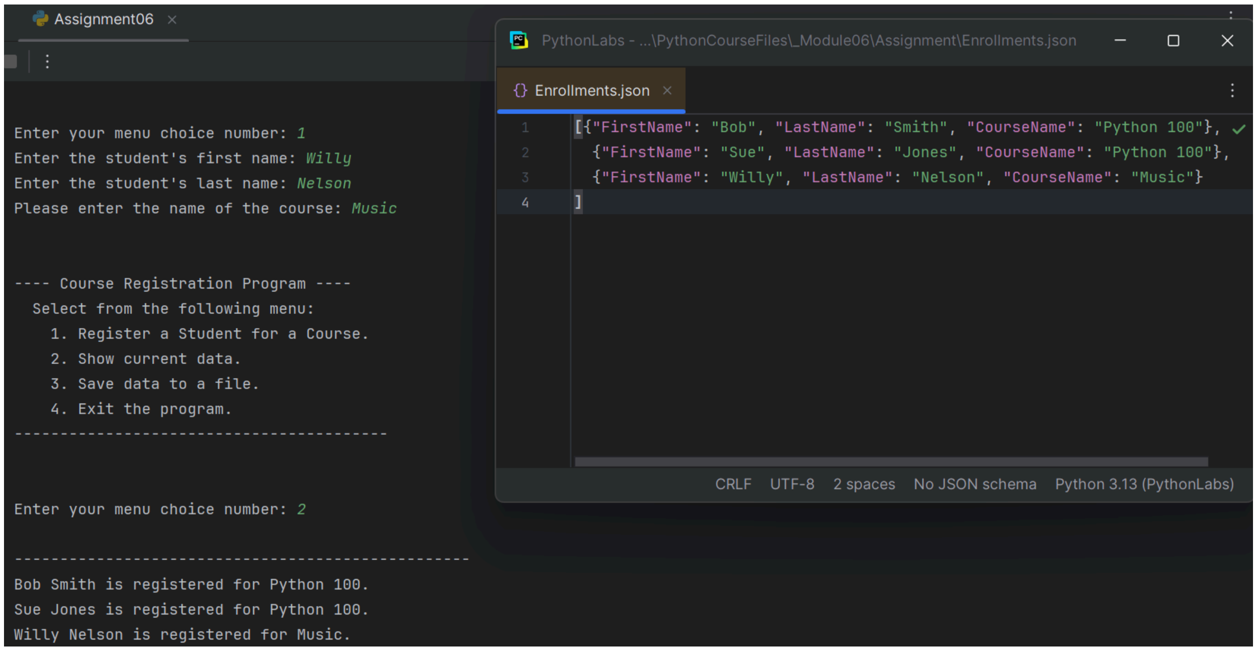
```
#Load current json data
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

# Repeat the follow tasks
while True:
    IO.output_menu(menu=MENU)
    menu_choice = IO.input_menu_choice()

    if menu_choice == "1":  # Get new student data
        IO.input_student_data(student_data=students)
        continue

    elif menu_choice == "2":  # Show all current data
        IO.output_student_courses(student_data=students)
        continue

    elif menu_choice == "3":  # Save data in a file
        FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
        continue

    elif menu_choice == "4":  # End the program
        break  # out of the while loop

print("Program Ended")
```

Figure 4 : Main Body

Topic 4: Testing code

SubTopic 4a - Testing in PyCharm

To test that the script is working properly in PyCharm and the console, I ran the script and followed each of the prompts to ensure the program was saving the user-inputted data to the Enrollemnts.csv file. (Fig 8 and Fig 9)



```
Enter your menu choice number: 1
Enter the student's first name: Willy
Enter the student's last name: Nelson
Please enter the name of the course: Music


---- 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

--------------------------------------------------
Bob Smith is registered for Python 100.
Sue Jones is registered for Python 100.
Willy Nelson is registered for Music.
```

PythonLabs - ...\PythonCourseFiles_Module06\Assignment\Enrollments.json

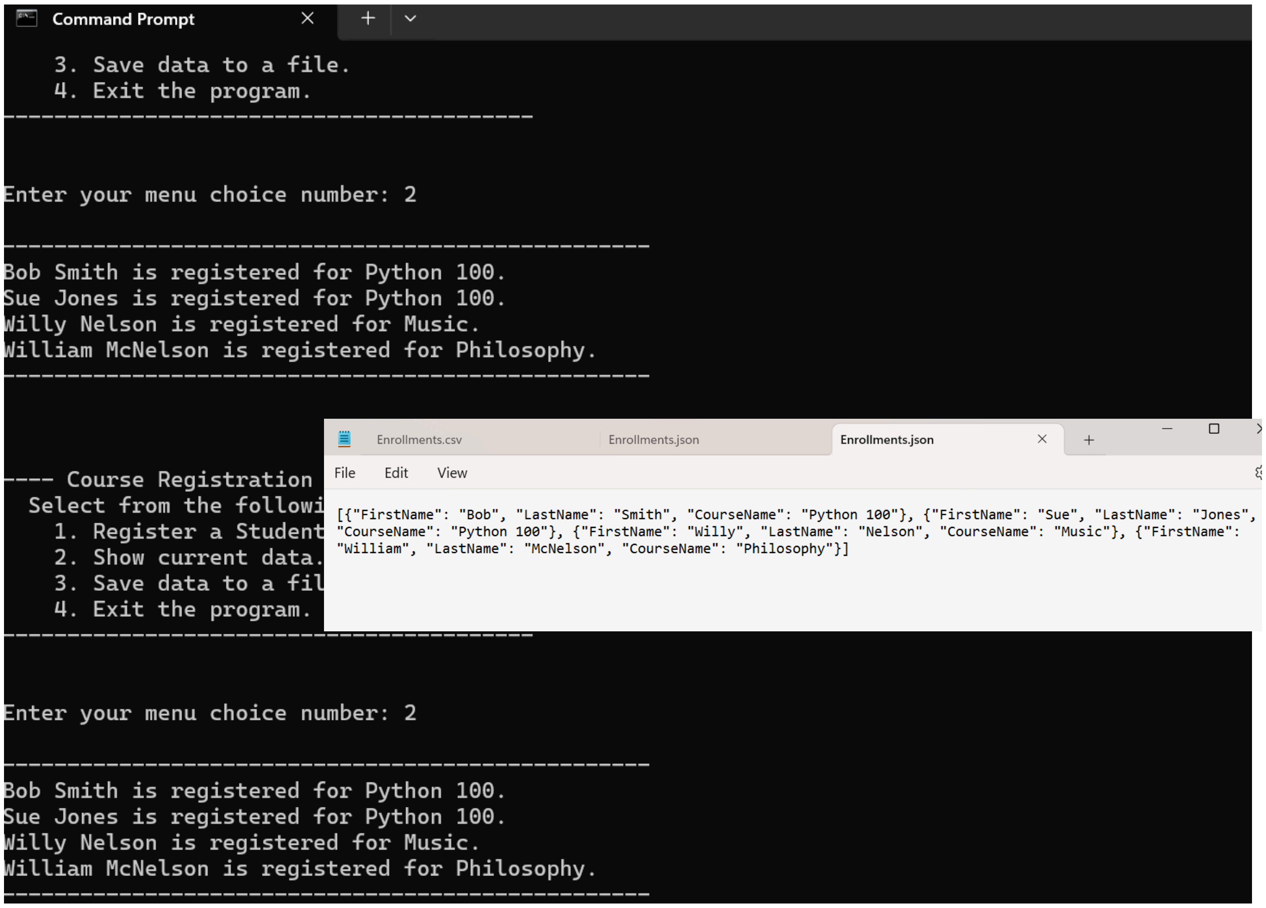
Enrollments.json

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

CRLF UTF-8 2 spaces No JSON schema Python 3.13 (PythonLabs)

Figure 5: Testing in PyCharm

SubTopic 5b - Testing in Console



Command Prompt

```
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


Enter your menu choice number: 2

--------------------------------------------------
Bob Smith is registered for Python 100.
Sue Jones is registered for Python 100.
Willy Nelson is registered for Music.
William McNelson is registered for Philosophy.
--------------------------------------------------


---- Course Registration
  Select from the followi
    1. Register a Student
    2. Show current data.
    3. Save data to a fil
    4. Exit the program.
-----------------------------------------


Enter your menu choice number: 2

--------------------------------------------------
Bob Smith is registered for Python 100.
Sue Jones is registered for Python 100.
Willy Nelson is registered for Music.
William McNelson is registered for Philosophy.
--------------------------------------------------
```

Enrollments.csv | Enrollments.json | Enrollments.json

File Edit View

```
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"}, {"FirstName": "Sue", "LastName": "Jones",
"CourseName": "Python 100"}, {"FirstName": "Willy", "LastName": "Nelson", "CourseName": "Music"}, {"FirstName":
"William", "LastName": "McNelson", "CourseName": "Philosophy"}]
```

Figure 5b: Testing in Console

Summary

In summary, I reviewed the steps I took to create a program in Python that opens and reads a json file's data into the program, presents a menu of options for a user to select, and depending on the user's selection, will either gather their data, present the data back, save the data to a file, or exit the program.