Giovanna Torres

2/26/2025

IT FDN 110 A Wi 25: Foundations Of Programming: Python

Assignment 5

GitHub Link: https://github.com/giozoar/IntroToProg-Python-Mod05

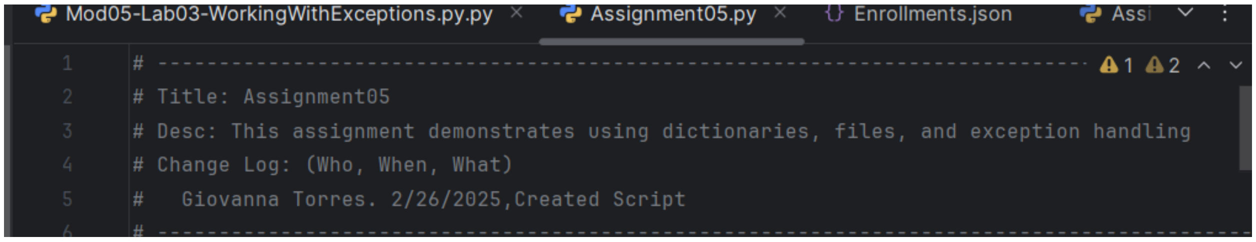
Advanced Collections and Error Handling

Introduction

This week as a part of my Foundations of Programming course, I learned how to work with dictionaries, which is a new data collection. I also learned how to read, process, and write these data collections to a JSON file, and handle associated errors. Below is how I went about the assignment.

Creating the Script

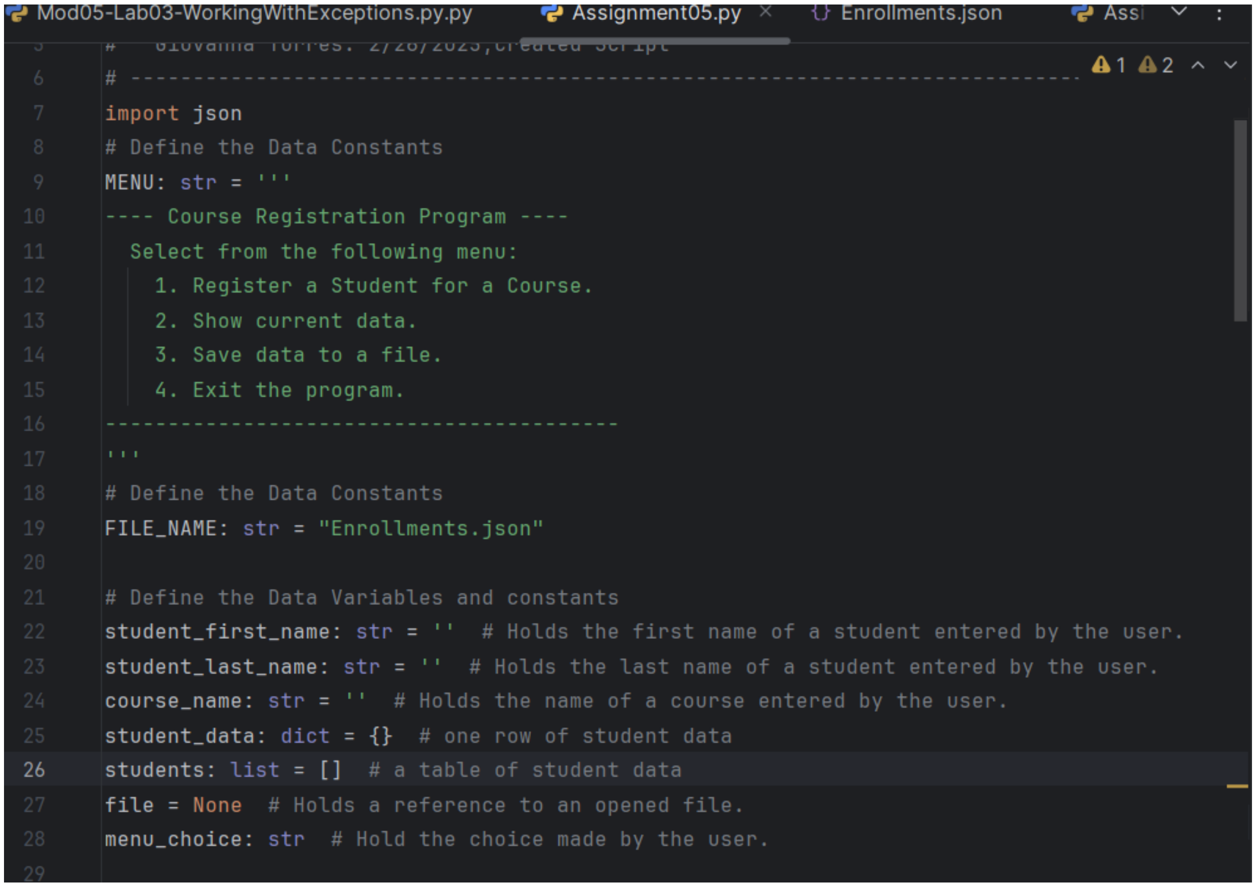
After reading the acceptance criteria described in the Mod05-Assignment file, I began to work on my script using the PyCharm Community Edition IDE. I reused the header from the Assignment05-Starter.py file included in the module materials to display the necessary information.



```
# ---------------------------------------------------------------------------
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   Giovanna Torres. 2/26/2025,Created Script
# ---------------------------------------------------------------------------
```

Figure 1 - Script Header

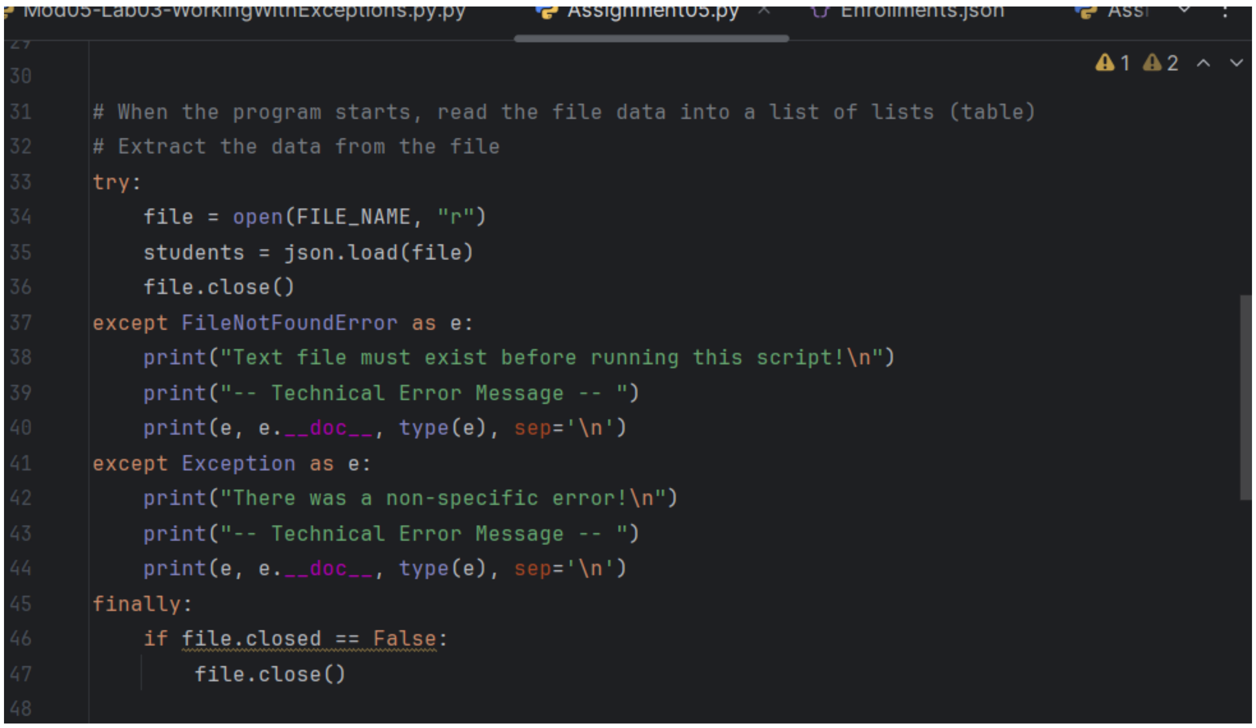
Afterwards, I added the ‘import json’ statement to my code, and copied the variables from the starter file, and verified the variables were consistent with components of the script body required to complete the assignment acceptance criteria. There were some edits needed such as recategorizing variables as dictionaries, and deleting unnecessary variables left over from Module 4.



```
# ------------------------------------------------------------------------
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.
-----------------------------------------
'''
# Define the Data Constants
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables and constants
student_first_name: str = ''  # Holds the first name of a student entered by the user.
student_last_name: str = ''  # Holds the last name of a student entered by the user.
course_name: str = ''  # Holds the name of a course entered by the user.
student_data: dict = {}  # one row of student data
students: list = []  # a table of student data
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
```

Figure 2 - Declaring constants and variables

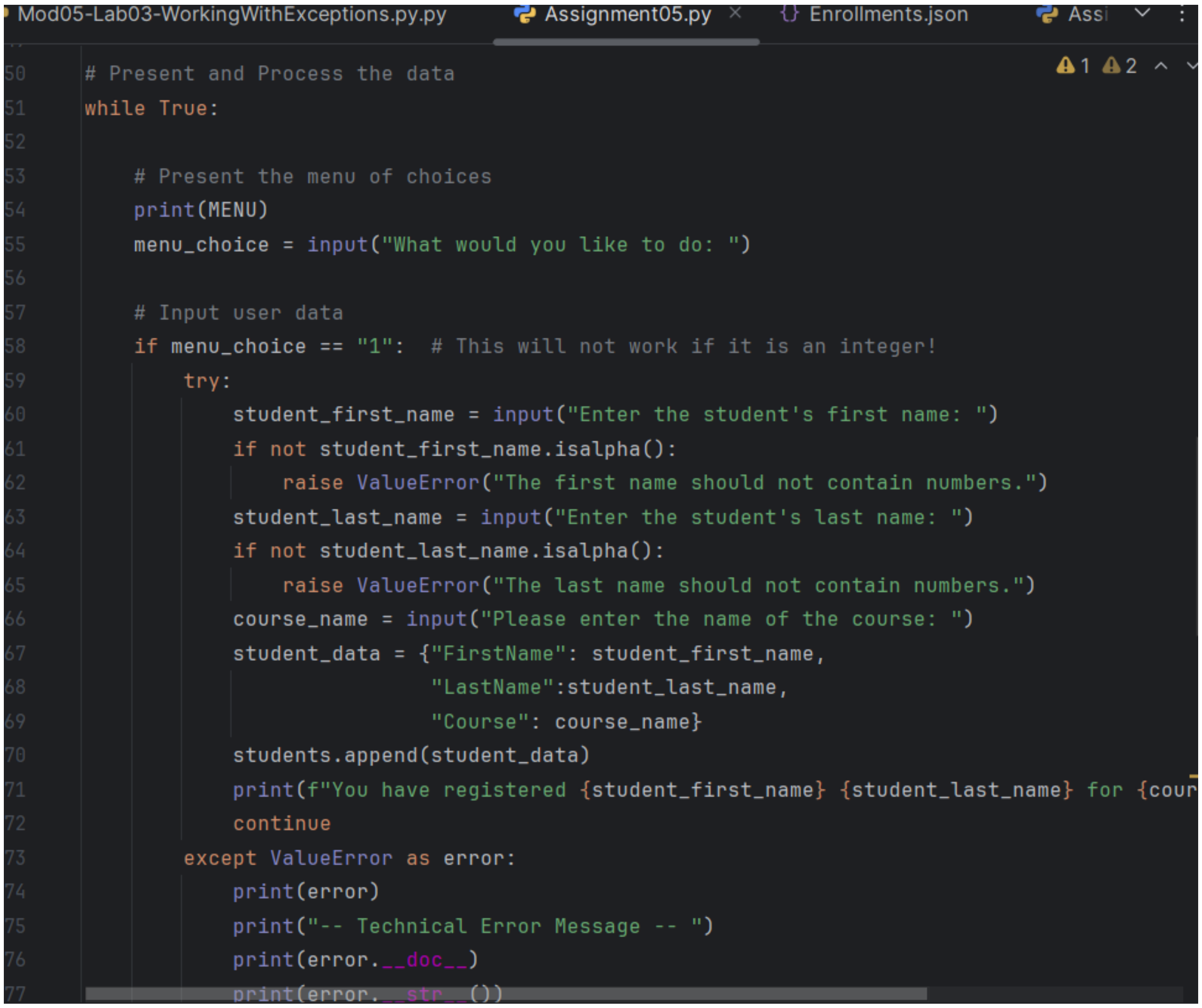
I then got to work on adding specific body components called out in the acceptance criteria, such as reading the ‘Enrollments.json’ file (renaming from the .csv in the previous module) using json commands and saving the data to a list called 'students'. This section also included error handling of the file read for having an existing file in place before the script can proceed.



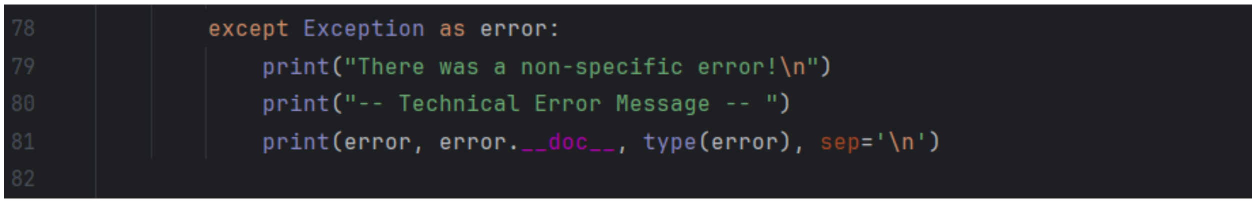
```python
# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)
    file.close()
except FileNotFoundError as e:
    print("Text file must exist before running this script!\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
except Exception as e:
    print("There was a non-specific error!\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
finally:
    if file.closed == False:
        file.close()
```

Figure 3 – JSON data ingestion and error handling

I then created a while loop to run my program infinitely until the user decides to break, and ask the user to enter student first name, last name, and registration course to register a student. The user input is then added to a dictionary and appended to the students list defined at the beginning. I included error handling for non-alphanumeric characters in the student_first_name and student_last_name fields, which displays specific error messages if an error is caught. If no error, the program displays the registered student information to the user.



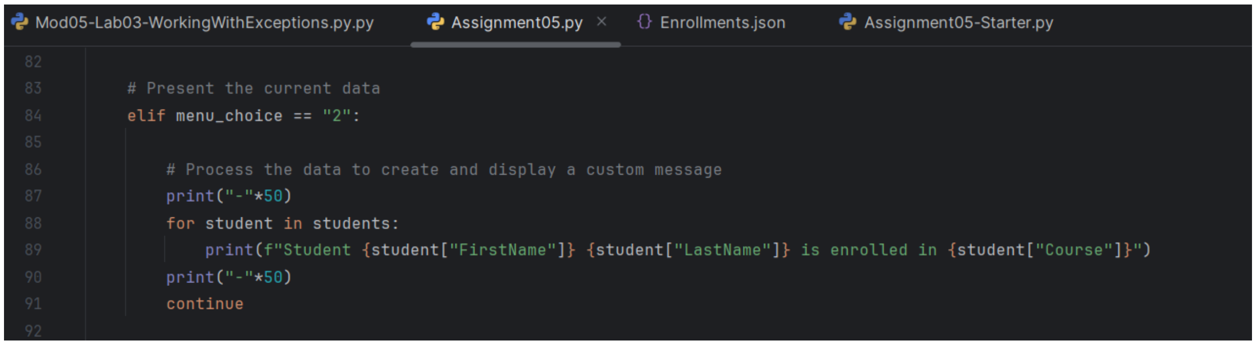
```
# Present and Process the data
while True:

    # Present the menu of choices
    print(MENU)
    menu_choice = input("What would you like to do: ")

    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        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 = {"FirstName": student_first_name,
                            "LastName":student_last_name,
                            "Course": course_name}
            students.append(student_data)
            print(f"You have registered {student_first_name} {student_last_name} for {cour
            continue
        except ValueError as error:
            print(error)
            print("-- Technical Error Message -- ")
            print(error.__doc__)
            print(error.__str__())
```



```
        except Exception as error:
            print("There was a non-specific error!\n")
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')

```

Figure 4 – Option 1 in loop for user input of enrollment data.

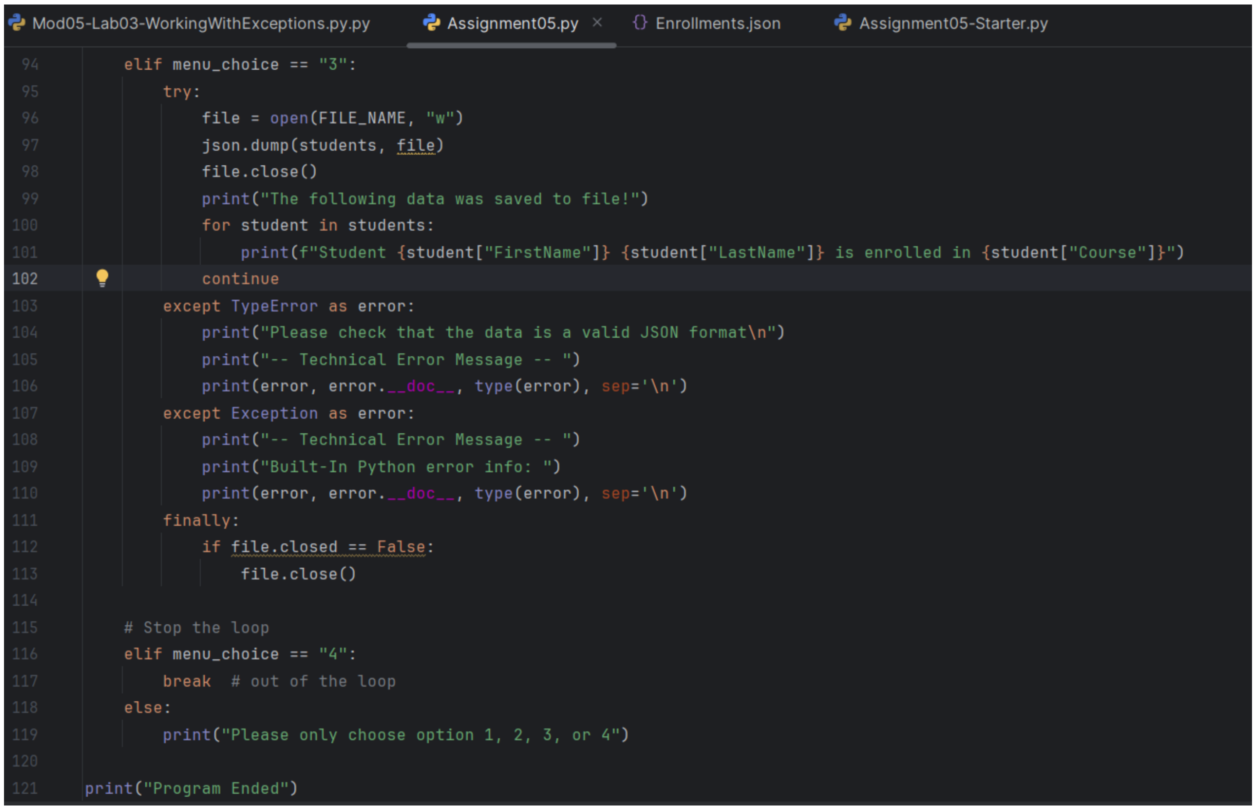
For option 2 of the menu options, I display the dictionaries in the list as previously registered student information.



```python

    # Present the current data
    elif menu_choice == "2":

        # Process the data to create and display a custom message
        print("-"*50)
        for student in students:
            print(f"Student {student["FirstName"]} {student["LastName"]} is enrolled in {student["Course"]}")
        print("-"*50)
        continue

```

Figure 5 - Option 2 display of registered student data.

In option 3 in my loop, I write the data saved in my list to my 'Enrollments.json' file using the built-in json language commands. I included error handling here as well for ensuring the data is in a valid JSON format. I finish my loop with option 4 for closing the file and a catch-all 'else' statement for invalid menu options.



```python
    elif menu_choice == "3":
        try:
            file = open(FILE_NAME, "w")
            json.dump(students, file)
            file.close()
            print("The following data was saved to file!")
            for student in students:
                print(f"Student {student["FirstName"]} {student["LastName"]} is enrolled in {student["Course"]}")
            continue
        except TypeError as error:
            print("Please check that the data is a valid JSON format\n")
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')
        except Exception as error:
            print("-- Technical Error Message -- ")
            print("Built-In Python error info: ")
            print(error, error.__doc__, type(error), sep='\n')
        finally:
            if file.closed == False:
                file.close()

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

print("Program Ended")
```

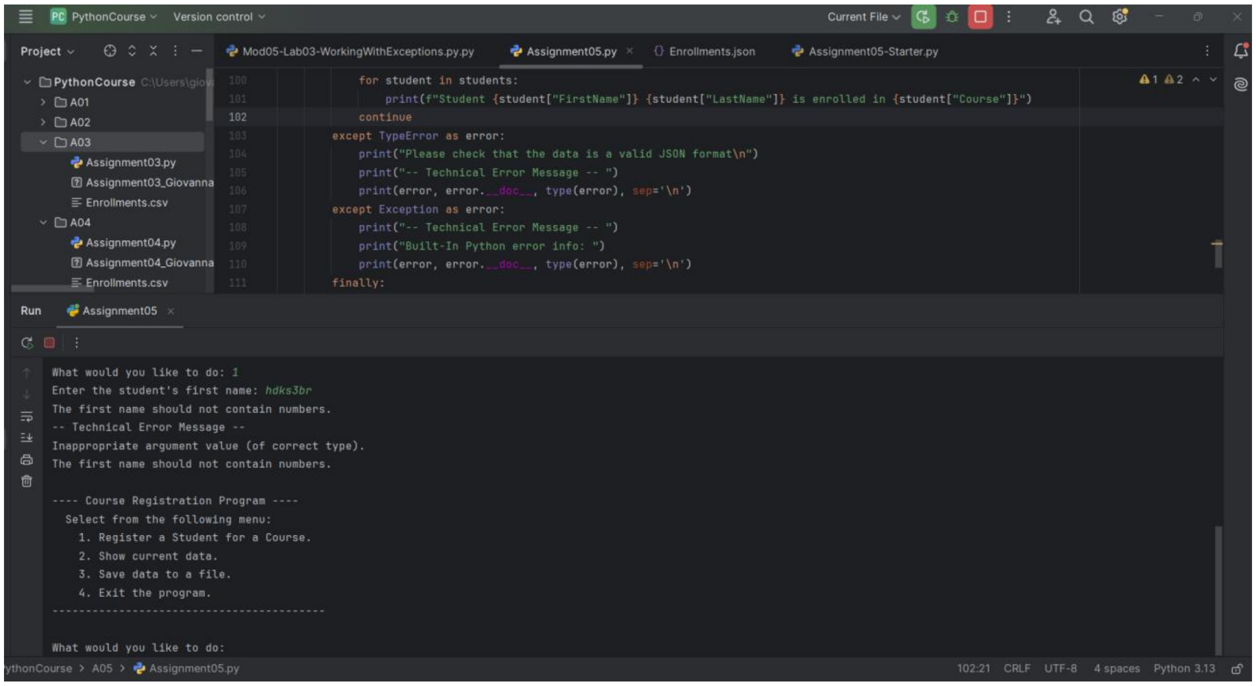
Figure 6 - Options 3 and 4 writing to JSON file and exiting the program with error handling.

I then proceeded to test my code, discussed in the next section.

Testing the Script

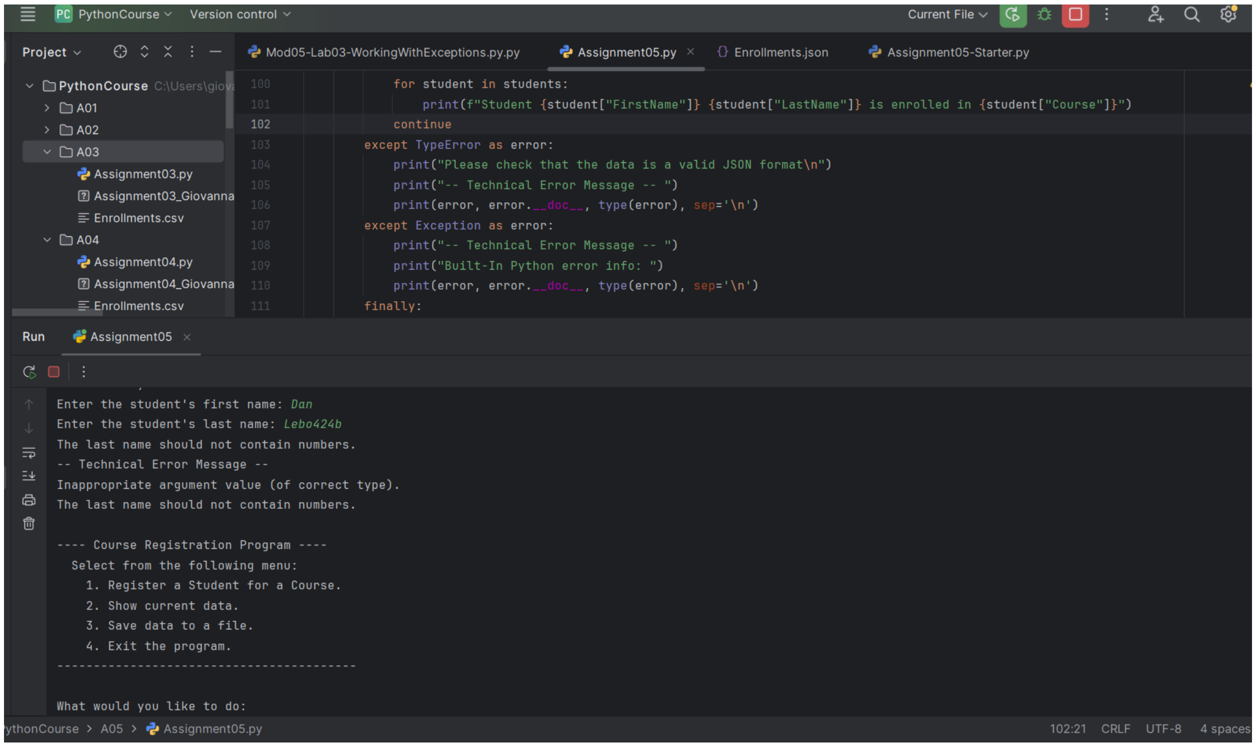
After setting up my initial ‘Enrollments.json’ data file with data from module 4 and running the script in the IDE to ensure that the correct outputs were being displayed and created in the directory, I proceeded to run the script in the Command Prompt terminal window. I changed the directory over to my ‘A05’ file within my PythonCourse file and ran the script.

I tested multiple cases, including adding multiple names, and adding characters that weren’t the numerical in the student name fields, and 1-4 menu option values. As I expected, the script would prompt the user to select another option if the input was invalid. It would also display multiple entries of student registrations and write them to the file as expected. The outputs were the same in either window, as seen below.



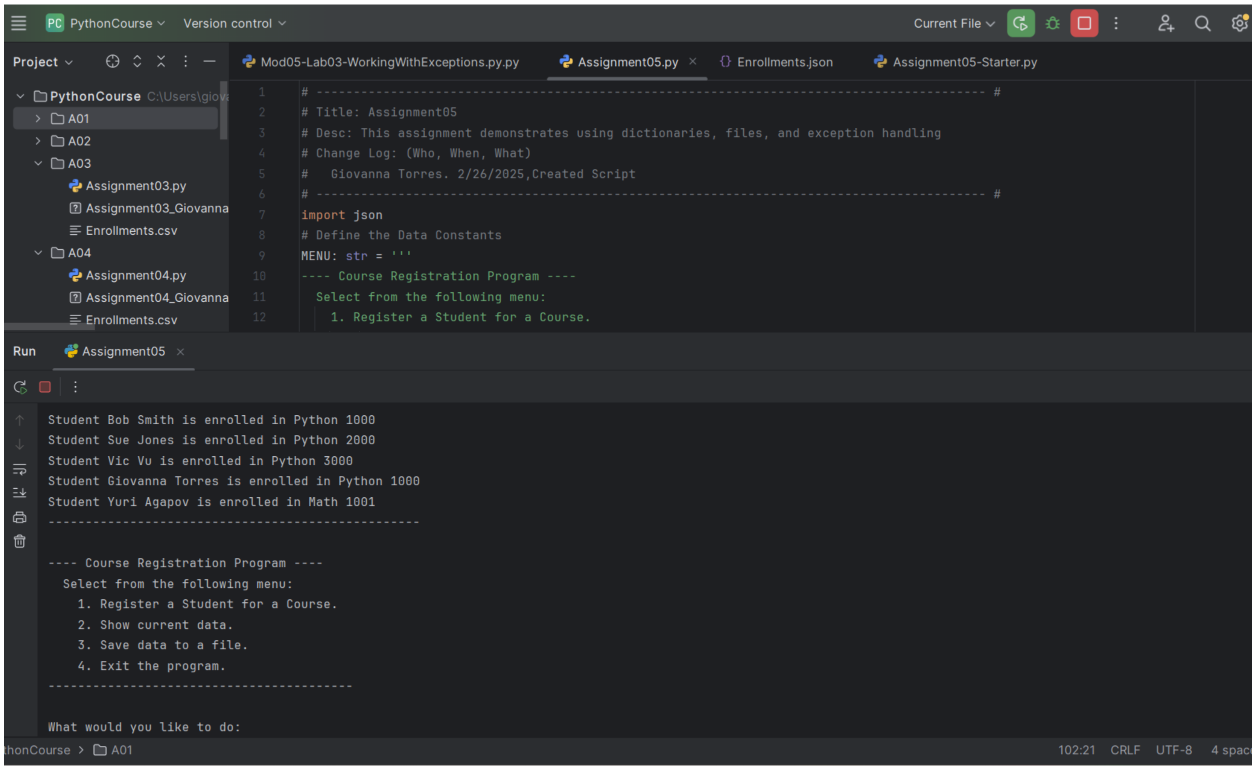
```
            for student in students:
                print(f"Student {student["FirstName"]} {student["LastName"]} is enrolled in {student["Course"]}")
            continue
        except TypeError as error:
            print("Please check that the data is a valid JSON format\n")
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')
        except Exception as error:
            print("-- Technical Error Message -- ")
            print("Built-In Python error info: ")
            print(error, error.__doc__, type(error), sep='\n')
        finally:
```

```
What would you like to do: 1
Enter the student's first name: hdks3br
The first name should not contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The first name should not contain numbers.

---- 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:
```



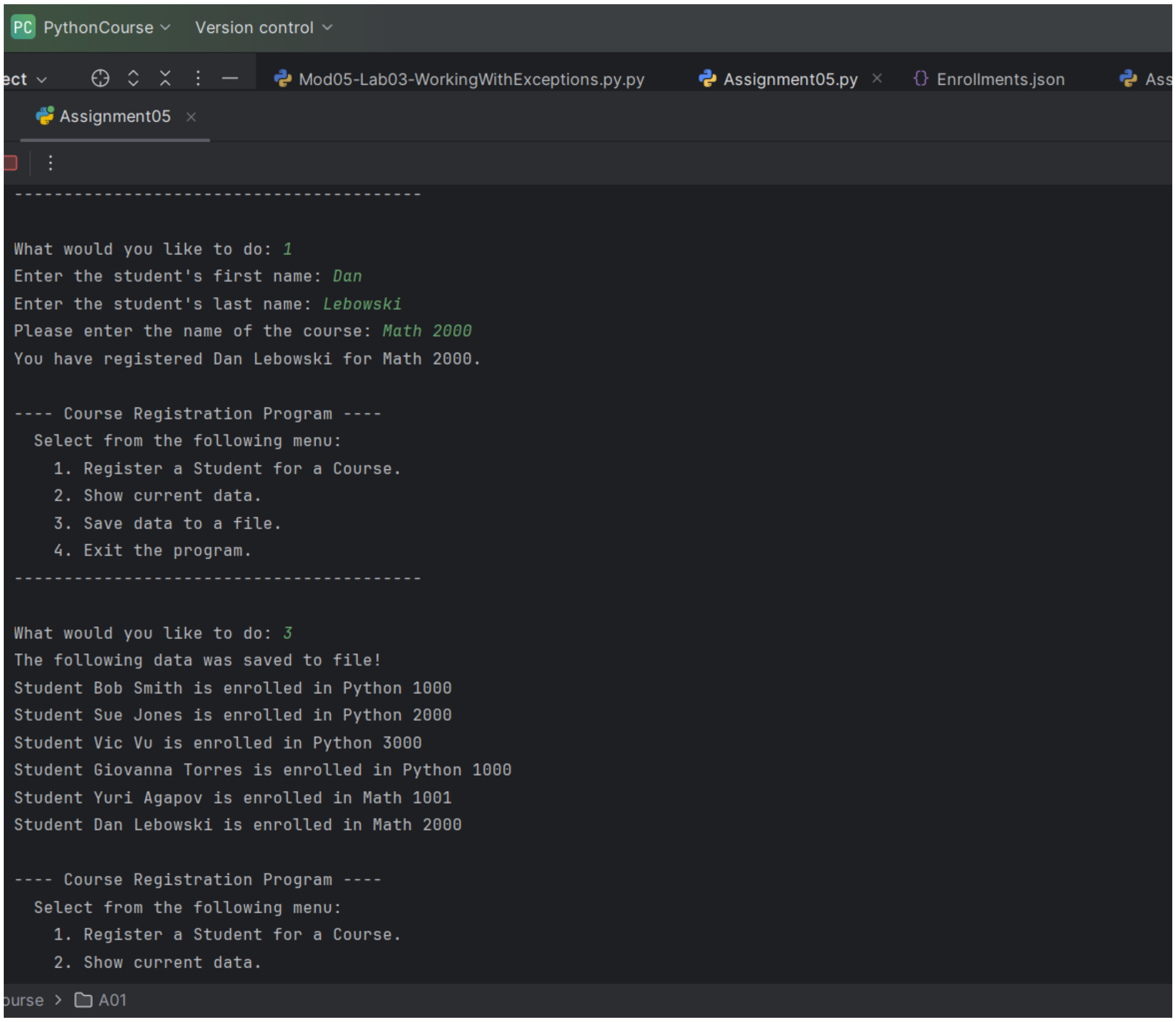
```python
            for student in students:
                print(f"Student {student["FirstName"]} {student["LastName"]} is enrolled in {student["Course"]}")
            continue
        except TypeError as error:
            print("Please check that the data is a valid JSON format\n")
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')
        except Exception as error:
            print("-- Technical Error Message -- ")
            print("Built-In Python error info: ")
            print(error, error.__doc__, type(error), sep='\n')
        finally:
```

```
Enter the student's first name: Dan
Enter the student's last name: Lebo424b
The last name should not contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The last name should not contain numbers.

---- 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:
```



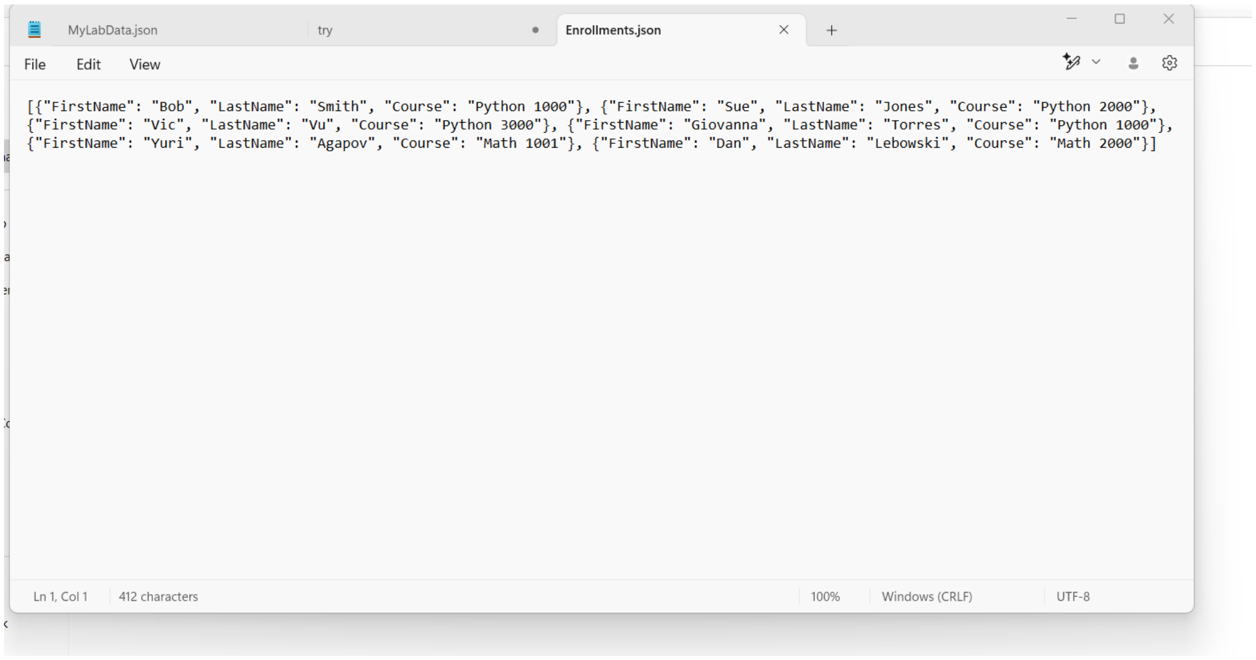
```python
# ------------------------------------------------------------------------------------------ #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   Giovanna Torres. 2/26/2025,Created Script
# ------------------------------------------------------------------------------------------ #
import json
# Define the Data Constants
MENU: str = '''
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
```

```
Student Bob Smith is enrolled in Python 1000
Student Sue Jones is enrolled in Python 2000
Student Vic Vu is enrolled in Python 3000
Student Giovanna Torres is enrolled in Python 1000
Student Yuri Agapov is enrolled in Math 1001
--------------------------------------------------

---- 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:
```



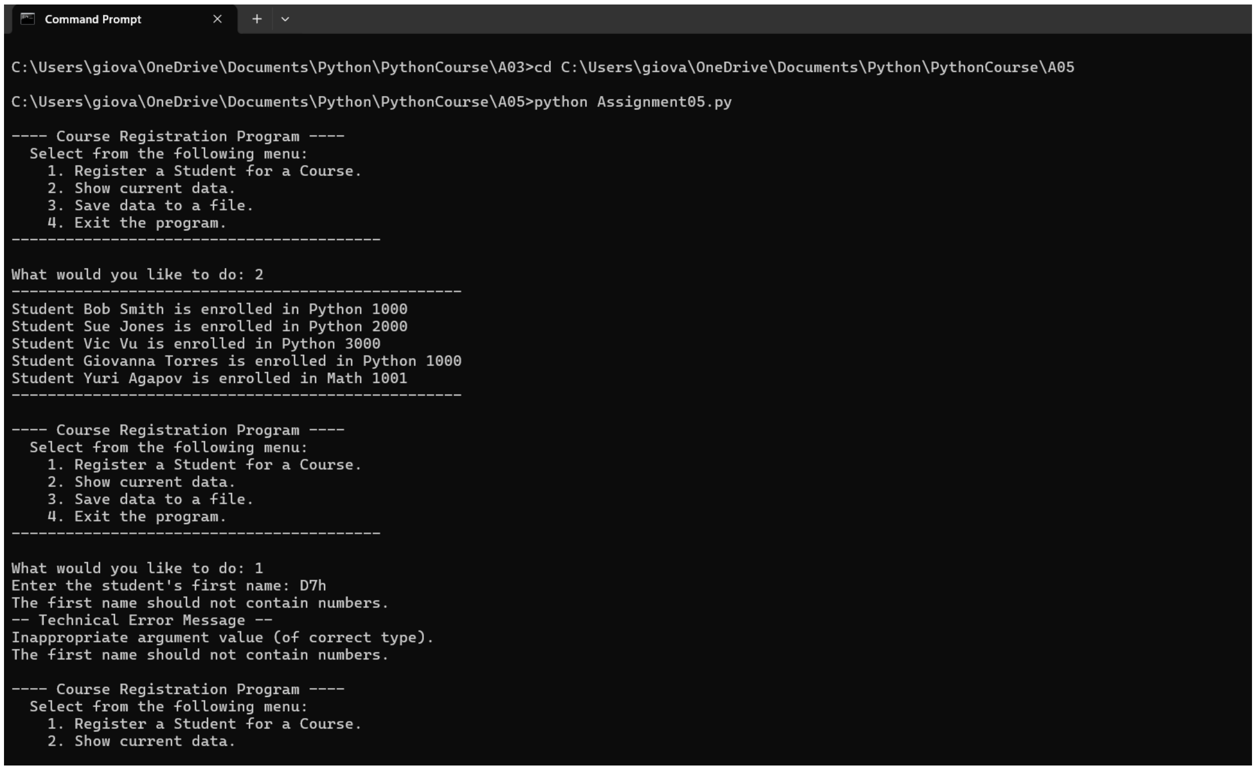
```
-----------------------------------------

What would you like to do: 1
Enter the student's first name: Dan
Enter the student's last name: Lebowski
Please enter the name of the course: Math 2000
You have registered Dan Lebowski for Math 2000.

---- 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
The following data was saved to file!
Student Bob Smith is enrolled in Python 1000
Student Sue Jones is enrolled in Python 2000
Student Vic Vu is enrolled in Python 3000
Student Giovanna Torres is enrolled in Python 1000
Student Yuri Agapov is enrolled in Math 1001
Student Dan Lebowski is enrolled in Math 2000

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
```

Figure 7 - Testing inputs in PyCharm IDE.

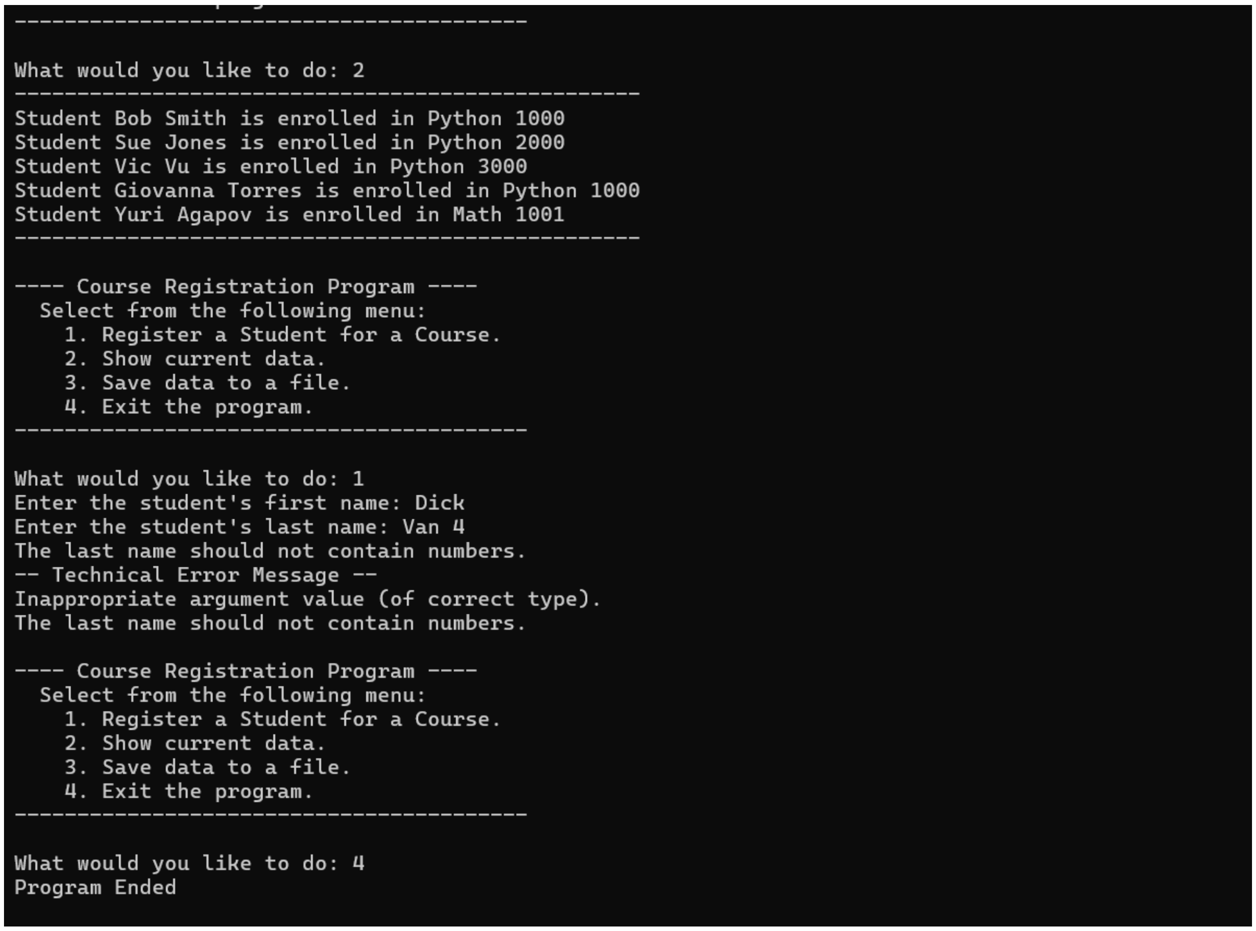


```
[{"FirstName": "Bob", "LastName": "Smith", "Course": "Python 1000"}, {"FirstName": "Sue", "LastName": "Jones", "Course": "Python 2000"},
{"FirstName": "Vic", "LastName": "Vu", "Course": "Python 3000"}, {"FirstName": "Giovanna", "LastName": "Torres", "Course": "Python 1000"},
{"FirstName": "Yuri", "LastName": "Agapov", "Course": "Math 1001"}, {"FirstName": "Dan", "LastName": "Lebowski", "Course": "Math 2000"}]
```

Figure 8 - Outputs in JSON file.



```
C:\Users\giova\OneDrive\Documents\Python\PythonCourse\A03>cd C:\Users\giova\OneDrive\Documents\Python\PythonCourse\A05

C:\Users\giova\OneDrive\Documents\Python\PythonCourse\A05>python Assignment05.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: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 1000
Student Sue Jones is enrolled in Python 2000
Student Vic Vu is enrolled in Python 3000
Student Giovanna Torres is enrolled in Python 1000
Student Yuri Agapov is enrolled in Math 1001
--------------------------------------------------

---- 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: D7h
The first name should not contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The first name should not contain numbers.

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
```



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

What would you like to do: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 1000
Student Sue Jones is enrolled in Python 2000
Student Vic Vu is enrolled in Python 3000
Student Giovanna Torres is enrolled in Python 1000
Student Yuri Agapov is enrolled in Math 1001
--------------------------------------------------

---- 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: Dick
Enter the student's last name: Van 4
The last name should not contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The last name should not contain numbers.

---- 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 - Testing in Command Prompt window.

Summary

During this assignment, I learned about managing lists and dictionaries, as well as manipulating JSON files and doing some basic error handling. This is very good information, but I think the biggest takeaway from this module is setting up my GitHub account, as this will be very useful in my job as a Product Manager to work with the same tools my dev team uses to push and pull code to and from repositories to make my product available. This is invaluable information for me to be able to navigate through the code base and knowing how our repos are structured, and for sharing my own repos in my resume in the future.