

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>

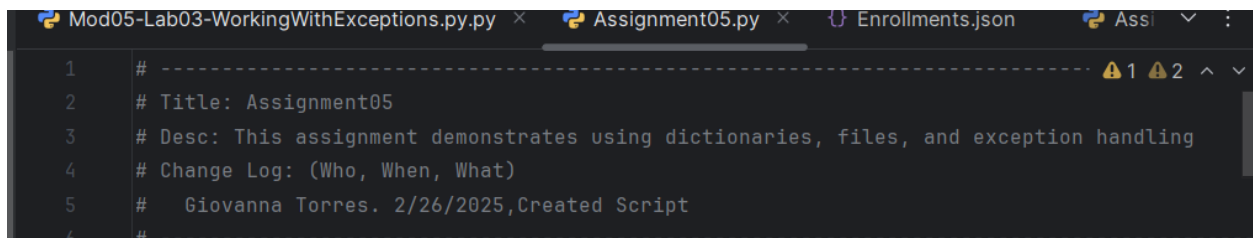
# Advanced Collections and Error Handling

## 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.

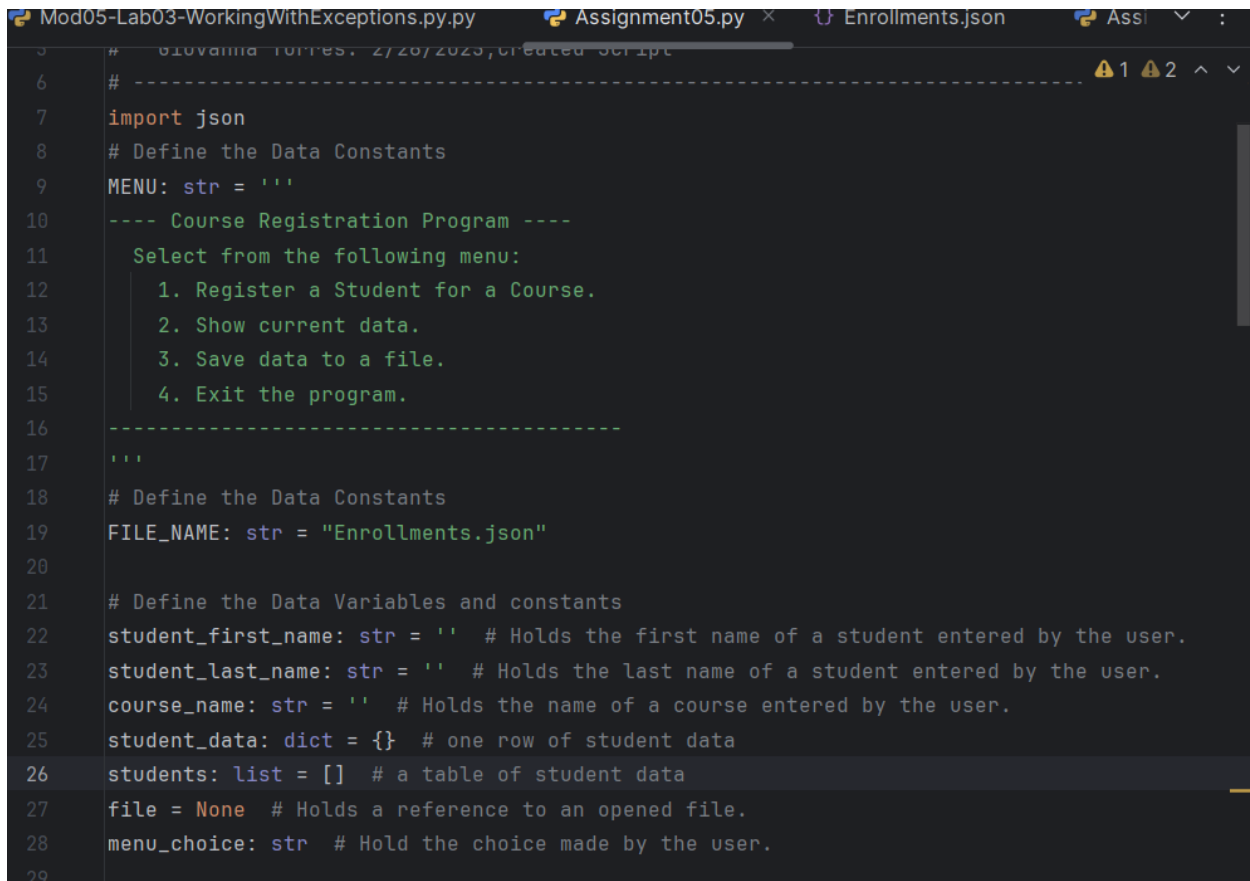


```
Mod05-Lab03-WorkingWithExceptions.py.py x Assignment05.py x Enrollments.json Assi v :
1 # -----
2 # Title: Assignment05
3 # Desc: This assignment demonstrates using dictionaries, files, and exception handling
4 # Change Log: (Who, When, What)
5 #   Giovanna Torres. 2/26/2025, Created Script
6 # -----
```

*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

reategorizing variables as dictionaries, and deleting unnecessary variables left over from Module 4.

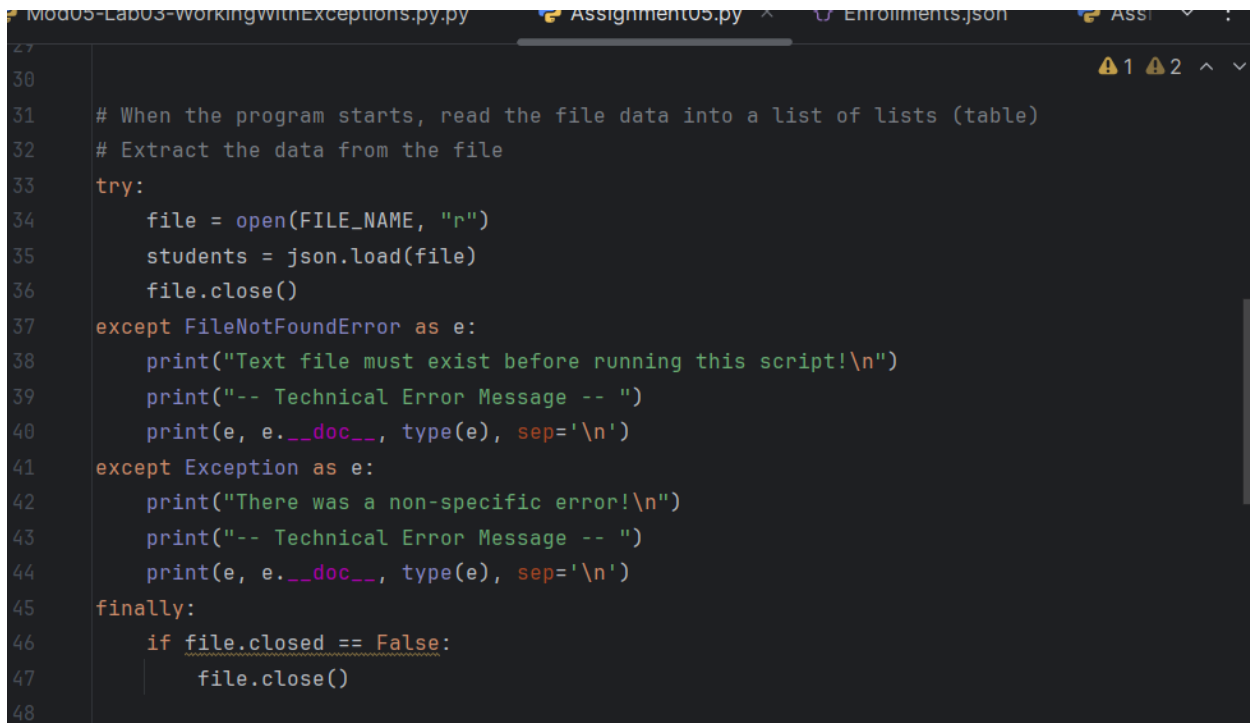


```
Mod05-Lab03-WorkingWithExceptions.py.py  Assignment05.py  x  Enrollments.json  Assi  v  :
5  # Giovanni Torres: 2/26/2023, created script
6  # -----
7  import json
8  # Define the Data Constants
9  MENU: str = '''
10  ---- Course Registration Program ----
11  Select from the following menu:
12  1. Register a Student for a Course.
13  2. Show current data.
14  3. Save data to a file.
15  4. Exit the program.
16  -----
17  '''
18  # Define the Data Constants
19  FILE_NAME: str = "Enrollments.json"
20
21  # Define the Data Variables and constants
22  student_first_name: str = '' # Holds the first name of a student entered by the user.
23  student_last_name: str = '' # Holds the last name of a student entered by the user.
24  course_name: str = '' # Holds the name of a course entered by the user.
25  student_data: dict = {} # one row of student data
26  students: list = [] # a table of student data
27  file = None # Holds a reference to an opened file.
28  menu_choice: str # Hold the choice made by the user.
29
```

*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.

A screenshot of a code editor window with a dark theme. The editor shows a Python script with line numbers 27 through 48 on the left margin. The code is as follows:

```
27  
30  
31 # When the program starts, read the file data into a list of lists (table)  
32 # Extract the data from the file  
33 try:  
34     file = open(FILE_NAME, "r")  
35     students = json.load(file)  
36     file.close()  
37 except FileNotFoundError as e:  
38     print("Text file must exist before running this script!\n")  
39     print("-- Technical Error Message -- ")  
40     print(e, e.__doc__, type(e), sep='\n')  
41 except Exception as e:  
42     print("There was a non-specific error!\n")  
43     print("-- Technical Error Message -- ")  
44     print(e, e.__doc__, type(e), sep='\n')  
45 finally:  
46     if file.closed == False:  
47         file.close()  
48
```

The code uses a try-except-finally block to handle file opening and JSON loading. It catches a FileNotFoundError and a general Exception, providing specific error messages and the exception details. A finally block ensures the file is closed if it was successfully opened.

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.

```

Mod05-Lab03-WorkingWithExceptions.py.py Assignment05.py x Enrollments.json Assl v :
50 # Present and Process the data
51 while True:
52
53     # Present the menu of choices
54     print(MENU)
55     menu_choice = input("What would you like to do: ")
56
57     # Input user data
58     if menu_choice == "1": # This will not work if it is an integer!
59         try:
60             student_first_name = input("Enter the student's first name: ")
61             if not student_first_name.isalpha():
62                 raise ValueError("The first name should not contain numbers.")
63             student_last_name = input("Enter the student's last name: ")
64             if not student_last_name.isalpha():
65                 raise ValueError("The last name should not contain numbers.")
66             course_name = input("Please enter the name of the course: ")
67             student_data = {"FirstName": student_first_name,
68                             "LastName": student_last_name,
69                             "Course": course_name}
70             students.append(student_data)
71             print(f"You have registered {student_first_name} {student_last_name} for {course_name}")
72             continue
73         except ValueError as error:
74             print(error)
75             print("-- Technical Error Message -- ")
76             print(error.__doc__)
77             print(error.__str__())
78
79         except Exception as error:
80             print("There was a non-specific error!\n")
81             print("-- Technical Error Message -- ")
82             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.

```
Mod05-Lab03-WorkingWithExceptions.py.py Assignment05.py x Enrollments.json Assignment05-Starter.py
82
83     # Present the current data
84     elif menu_choice == "2":
85
86         # Process the data to create and display a custom message
87         print("-"*50)
88         for student in students:
89             print(f"Student {student['FirstName']} {student['LastName']} is enrolled in {student['Course']}")
90         print("-"*50)
91         continue
92
```

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.

```
Mod05-Lab03-WorkingWithExceptions.py.py Assignment05.py x Enrollments.json Assignment05-Starter.py
94     elif menu_choice == "3":
95         try:
96             file = open(FILE_NAME, "w")
97             json.dump(students, file)
98             file.close()
99             print("The following data was saved to file!")
100             for student in students:
101                 print(f"Student {student['FirstName']} {student['LastName']} is enrolled in {student['Course']}")
102             continue
103         except TypeError as error:
104             print("Please check that the data is a valid JSON format\n")
105             print("-- Technical Error Message -- ")
106             print(error, error.__doc__, type(error), sep='\n')
107         except Exception as error:
108             print("-- Technical Error Message -- ")
109             print("Built-In Python error info: ")
110             print(error, error.__doc__, type(error), sep='\n')
111         finally:
112             if file.closed == False:
113                 file.close()
114
115         # Stop the loop
116         elif menu_choice == "4":
117             break # out of the loop
118         else:
119             print("Please only choose option 1, 2, 3, or 4")
120
121     print("Program Ended")
```

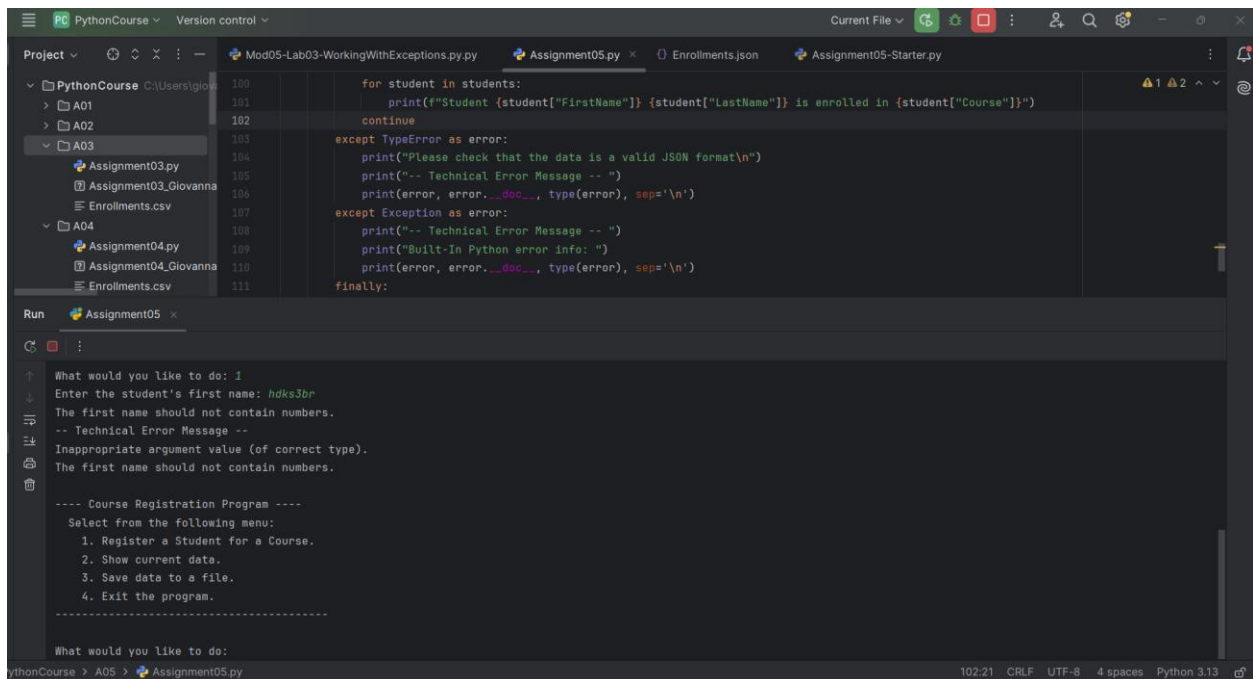
Figure 6 - Options 3 and 4 writing to JSON file and exiting the program with error handling.

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.



The screenshot shows a Python IDE with a project named 'PythonCourse'. The file explorer on the left shows a directory structure with folders A01, A02, A03, and A04. Inside A03, there are files 'Assignment03.py', 'Assignment03\_Giovanna', and 'Enrollments.csv'. Inside A04, there are files 'Assignment04.py', 'Assignment04\_Giovanna', and 'Enrollments.csv'. The main editor window shows a Python script with the following code:

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

The Run window shows the output of the script:

```
What would you like to do: 1
Enter the student's first name: hdk33br
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:
```

The status bar at the bottom shows the file path 'pythonCourse > A05 > Assignment05.py', the line and column number '102:21', the encoding 'CRLF', the file type 'UTF-8', the indentation '4 spaces', and the Python version 'Python 3.13'.

The screenshot shows the VS Code editor with the file `Assignment05.py` open. The code defines a function `is_enrolled` that checks if a student is enrolled in a specific course. It uses a `for` loop to iterate over a list of students and prints a message if a student is enrolled. The function also includes exception handling for `TypeError` and `Exception`. The terminal window shows the program's output, including prompts for student name and last name, and a menu for course registration.

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

Run Assignment05 x

```
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.

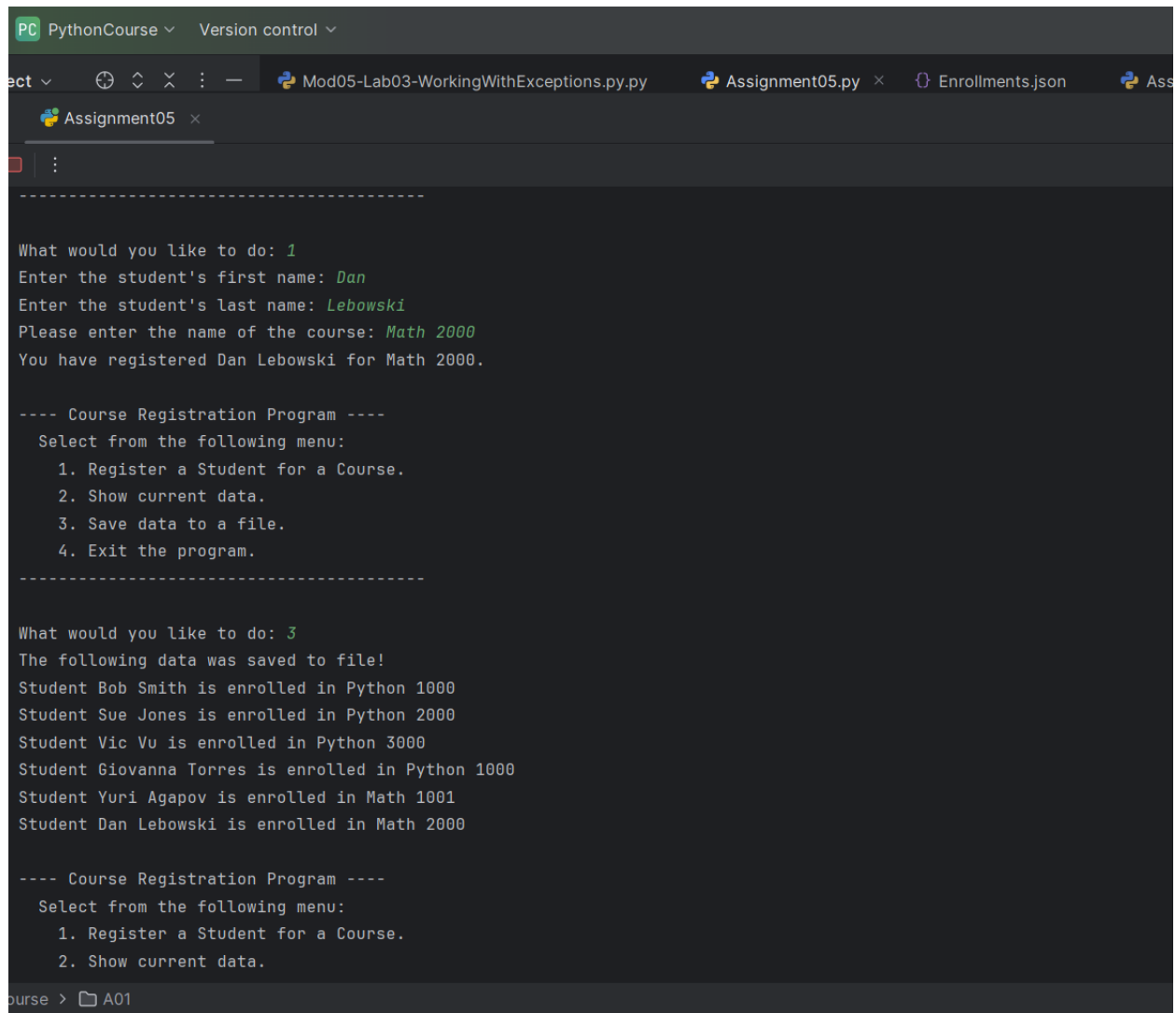
---- Course Registration Program ----
Select from the following menu:
1. Register a Student for a Course.
2. Show current data.
3. Save data to a file.
4. Exit the program.
-----
What would you like to do:
```

The screenshot shows the VS Code editor with the file `Assignment05.py` open. The code defines a function `is_enrolled` that checks if a student is enrolled in a specific course. It uses a `for` loop to iterate over a list of students and prints a message if a student is enrolled. The function also includes exception handling for `TypeError` and `Exception`. The terminal window shows the program's output, including prompts for student name and last name, and a menu for course registration.

```
1 # -----
2 # Title: Assignment05
3 # Desc: This assignment demonstrates using dictionaries, files, and exception handling
4 # Change Log: (Who, When, What)
5 #   Giovanni Torres. 2/26/2025, Created Script
6 # -----
7 import json
8 # Define the Data Constants
9 MENU: str = ''
10 ---- Course Registration Program ----
11 Select from the following menu:
12 1. Register a Student for a Course.
```

Run Assignment05 x

```
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:
```



```
PC PythonCourse Version control
Mod05-Lab03-WorkingWithExceptions.py.py Assignment05.py x Enrollments.json Ass
Assignment05 x

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.



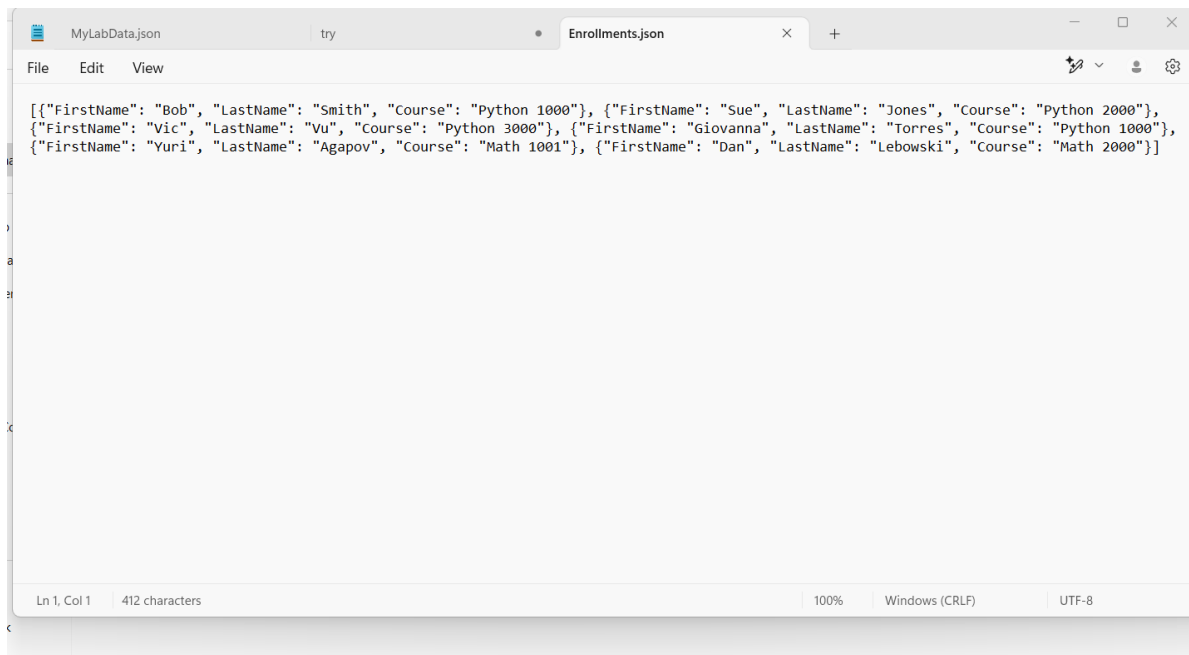


Figure 8 - Outputs in JSON file.

```
Command Prompt
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.