Alvin Jappeth Isip
November 16, 2024
IT FDN 110 A – Foundations of Programming: Python
Assignment 05

Advance Collections and Error Handling

Introduction

On this module, it was discussed writing data to a file from dictionary and reading data from a file into a dictionary. It was also discussed the JSON (JavaScript Object Notation) file and how Python's json module can simplify data handling. This module also discussed error handling and it's importance. And lastly, how can we share, discuss, collaborate using GitHub.

Module 5 Requirements

I did open the module 5 notes and the assignment to see the requirements for this assignment. The assignment is more likely with assignment 4 but additional requirements:

1. The constant **FILE_NAME: str** is set to the value "Enrollments.json"
2. Creating this variables:

- **json_data: str** is set to empty string.
- **student_data: dict** is set to and empty list
-

3. Data collected for menu choice 1 is added to a two-dimensional list table (list of dictionaries).

4. When the program starts, the contents of the "Enrollments.json" are automatically read into a two-dimensional list table (a list of dictionary rows). (**Tip:** Make sure to put some starting data into the file or you will get an error!)

5. On menu choice 3, the program opens a file named "Enrollments.json" in write mode using the open() function. It writes the content of the students variable to the file using the dump() function, then file is closed using the close() method. Then displays what was stored in the file.

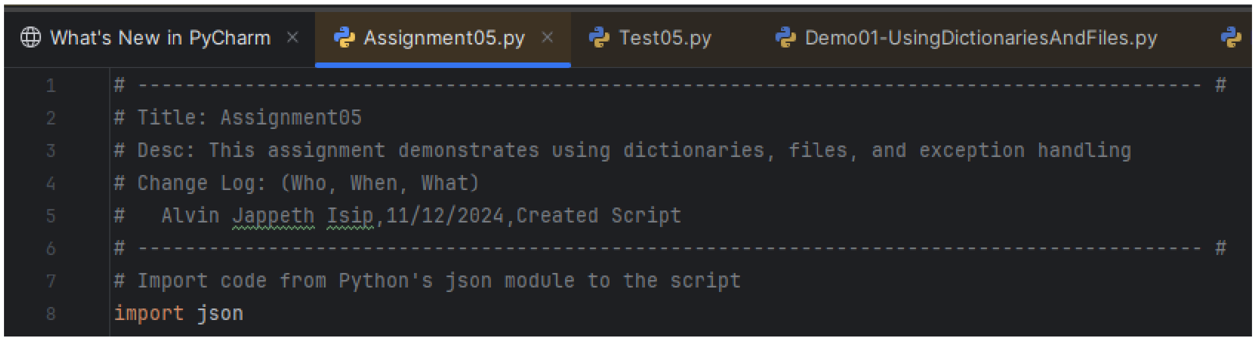
Error Handling

Compared to previous assignments, this assignment requires error handling. Error Handing improves your scripts by managing errors you may not have control over in any other way.

- The program provides structured error handling when the file is read into the list of dictionary rows.
- The program provides structured error handling when the user enters a first name.
- The program provides structured error handling when the user enters a last name.
- The program provides structured error handling when the dictionary rows are written to the file.

Writing Script

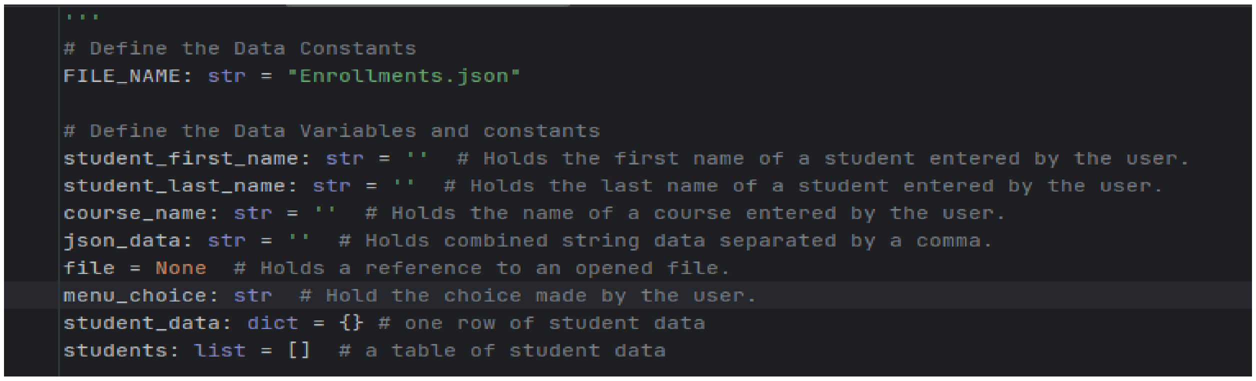
To start my coding, I opened the Assignment05-Starter.py and saved it as Assignment05.py. I did changed the header script to reflect what I'll be doing for this assignment.



```python
# ------------------------------------------------------------------------------------------ #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   Alvin Jappeth Isip,11/12/2024,Created Script
# ------------------------------------------------------------------------------------------ #
# Import code from Python's json module to the script
import json
```

Figure 1. Header Script

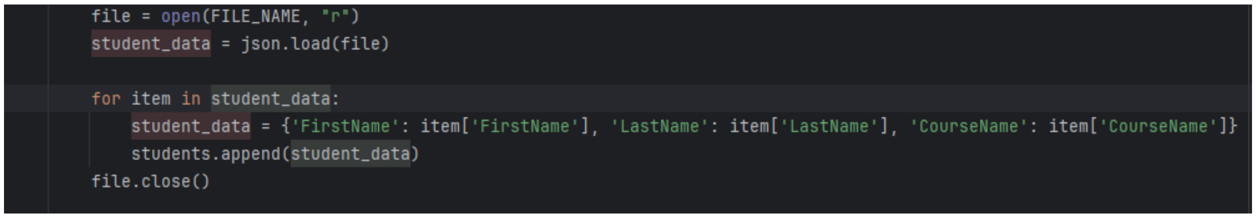
I added the needed variables and constants.



```python
'''
# 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.
json_data: str = ''  # Holds combined string data separated by a comma.
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
student_data: dict = {} # one row of student data
students: list = []  # a table of student data
```

Figure 2. Variables and constants

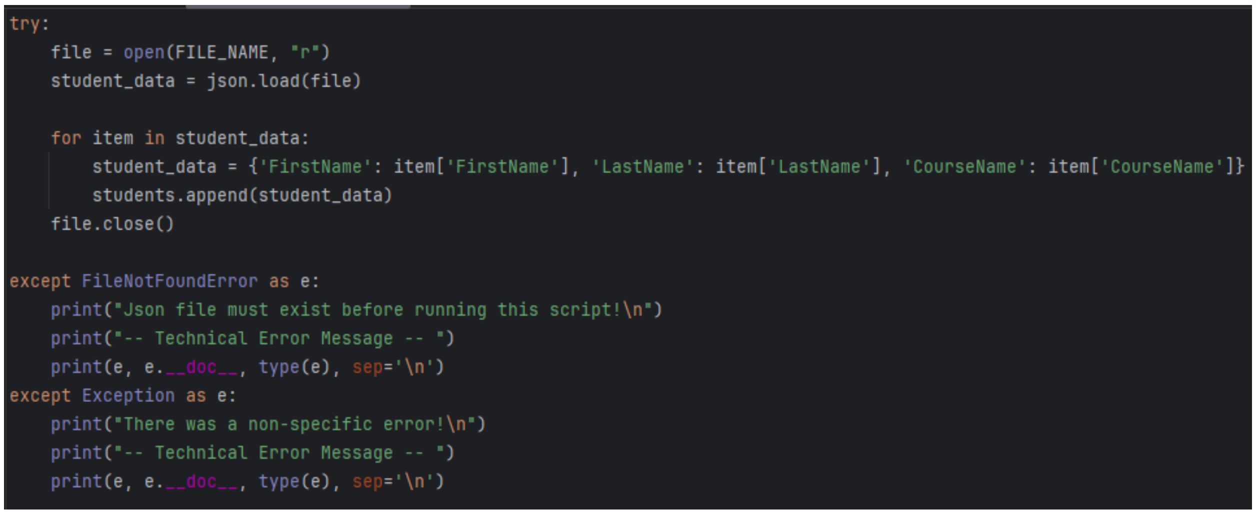
Since it is on the requirement that the json file needs to be readed first, I tested that script first since I think it will be easy to code. I did write the script below to read the json file. It looks like a simple code but took me a while since I'm not familiar with this programming but using the notes and videos, also sample program, I was able to write this.



```
    file = open(FILE_NAME, "r")
    student_data = json.load(file)

    for item in student_data:
        student_data = {'FirstName': item['FirstName'], 'LastName': item['LastName'], 'CourseName': item['CourseName']}
        students.append(student_data)
    file.close()
```

Figure 3. Reading json file

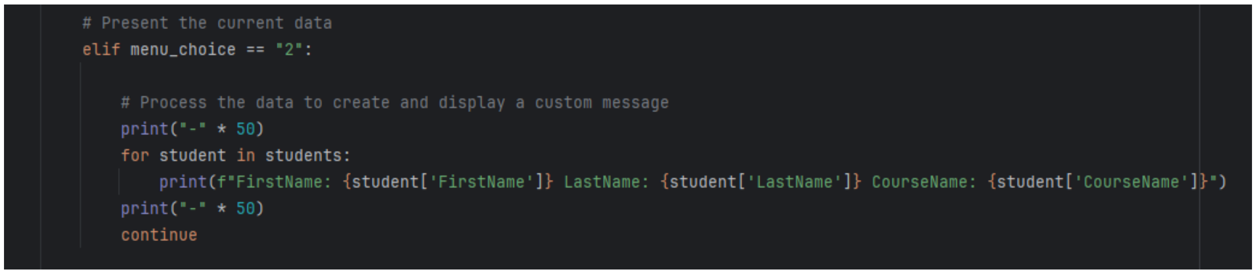
Written also is the error handling when opening the json file.



```
try:
    file = open(FILE_NAME, "r")
    student_data = json.load(file)

    for item in student_data:
        student_data = {'FirstName': item['FirstName'], 'LastName': item['LastName'], 'CourseName': item['CourseName']}
        students.append(student_data)
    file.close()

except FileNotFoundError as e:
    print("Json 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')
```

Figure 4. Error handling for the input file

Since the requirement for menu 2 is almost the same with reading the json file, I wrote this code for menu 2.

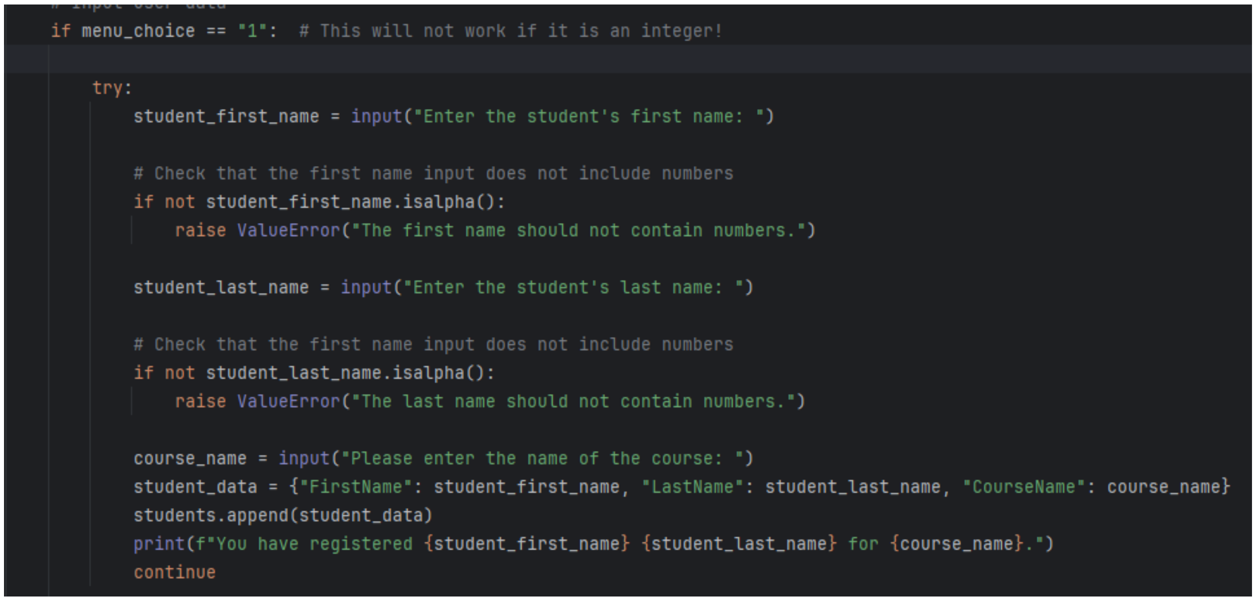


```
    # Present the current data
    elif menu_choice == "2":

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

Figure 5. Menu 2 script

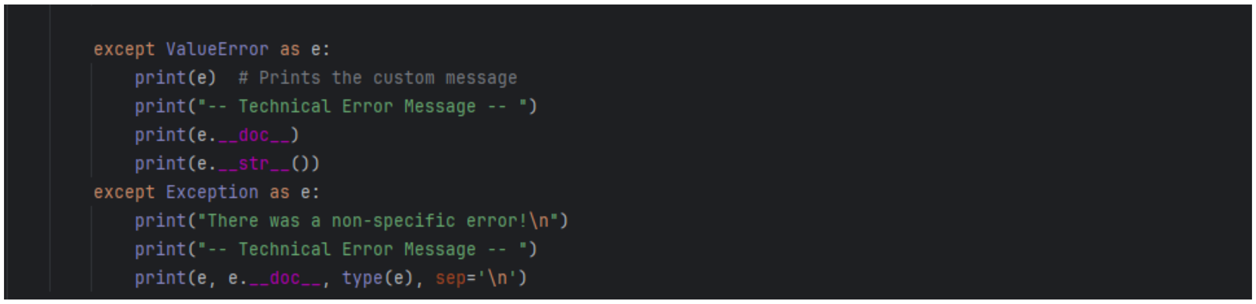
The next thing that I wrote is the script for menu 1. Still used the input() function to ask for users input but now, there is an error handling for entering student first name and student last name. Isalpha() means that the input should just be letters only.

Also, I still used the append to append the entered data to the students list.



```
if menu_choice == "1":  # This will not work if it is an integer!

    try:
        student_first_name = input("Enter the student's first name: ")

        # Check that the first name input does not include numbers
        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: ")

        # Check that the first name input does not include numbers
        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, "CourseName": course_name}
        students.append(student_data)
        print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
        continue
```

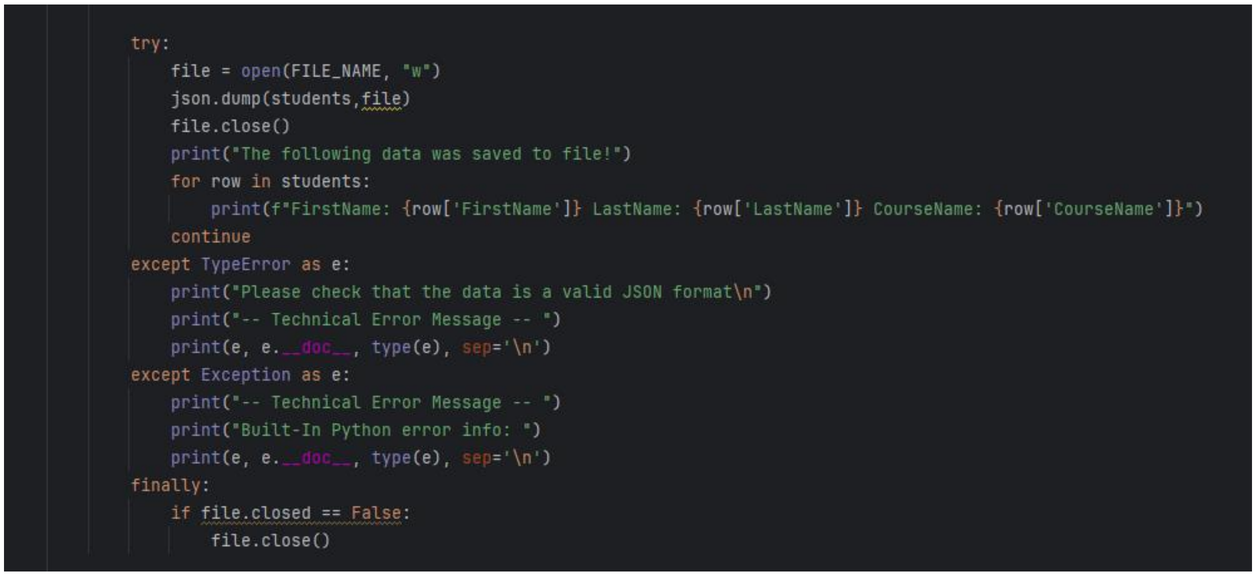
Figure 6. Menu 1



```
    except ValueError as e:
        print(e)  # Prints the custom message
        print("-- Technical Error Message -- ")
        print(e.__doc__)
        print(e.__str__())
    except Exception as e:
        print("There was a non-specific error!\n")
        print("-- Technical Error Message -- ")
        print(e, e.__doc__, type(e), sep='\n')
```

Figure 7. Error handling in Menu 1

For menu 3, I still used the open() function to open the file and W to write on the json file. I used the json.dump() function to write the table list (which contains dictionaries) to the JSON file. I also write error handling when the format is not in JSON format.

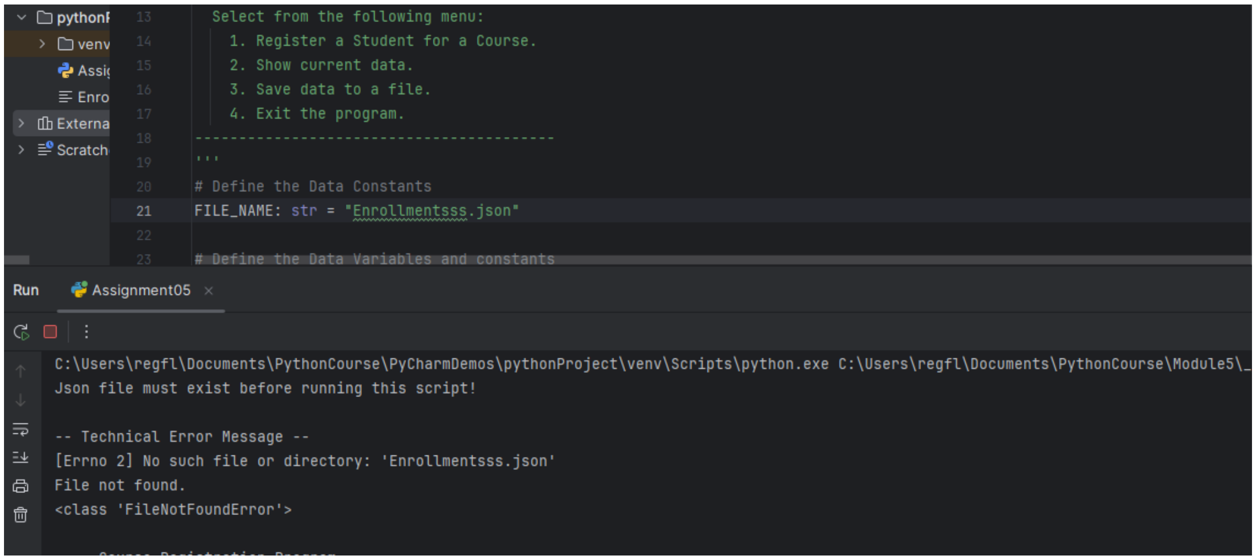


```
try:
    file = open(FILE_NAME, "w")
    json.dump(students,file)
    file.close()
    print("The following data was saved to file!")
    for row in students:
        print(f"FirstName: {row['FirstName']} LastName: {row['LastName']} CourseName: {row['CourseName']}")
    continue
except TypeError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

Figure 8. Menu 3 with error handling

Testing the error handling

I tested first if the error handling when reading the json file is working. I changed the Enrollments.json to Enrollmentsss.json to see if it will be catched.

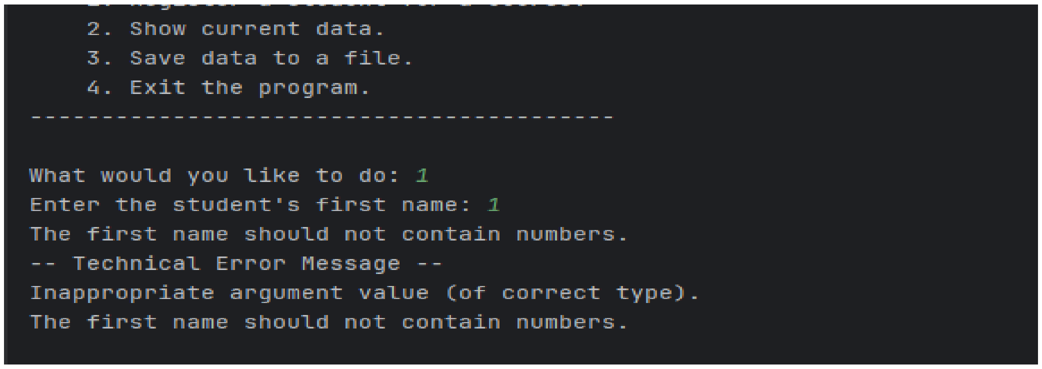


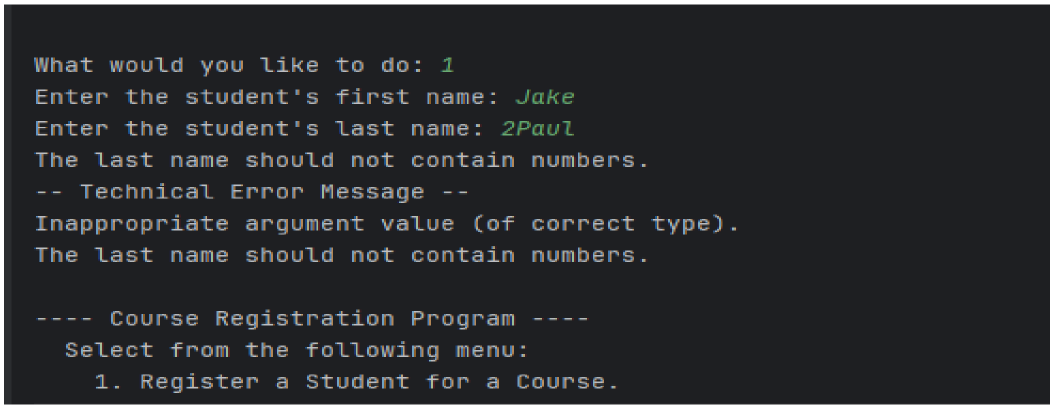
```
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
----------------------------------------
'''
# Define the Data Constants
FILE_NAME: str = "Enrollmentsss.json"

# Define the Data Variables and constants
```

```
C:\Users\regfl\Documents\PythonCourse\PyCharmDemos\pythonProject\venv\Scripts\python.exe C:\Users\regfl\Documents\PythonCourse\Module5\_
Json file must exist before running this script!

-- Technical Error Message --
[Errno 2] No such file or directory: 'Enrollmentsss.json'
File not found.
<class 'FileNotFoundError'>
```

Figure 9. Error when file is not found

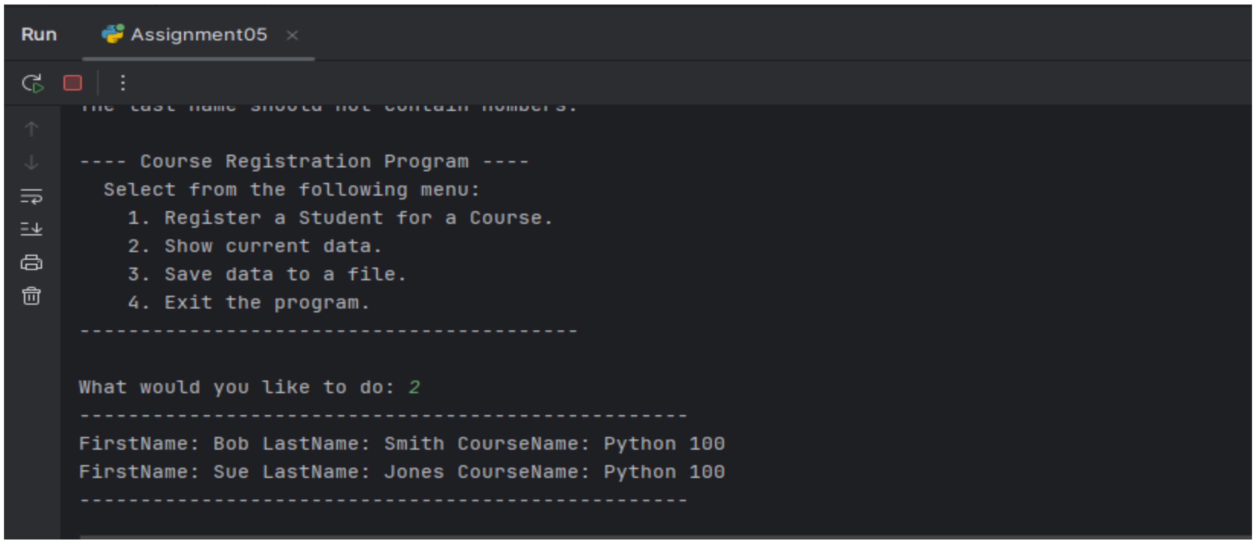
Now I tested the menu 1 if I will put number on the student first and last name.


```
    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
The first name should not contain numbers.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The first name should not contain numbers.
```
Figure 10. Error when student first name contains number


```
What would you like to do: 1
Enter the student's first name: Jake
Enter the student's last name: 2Paul
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.
```
Figure 11. Error when student last name contains number

Testing the Script

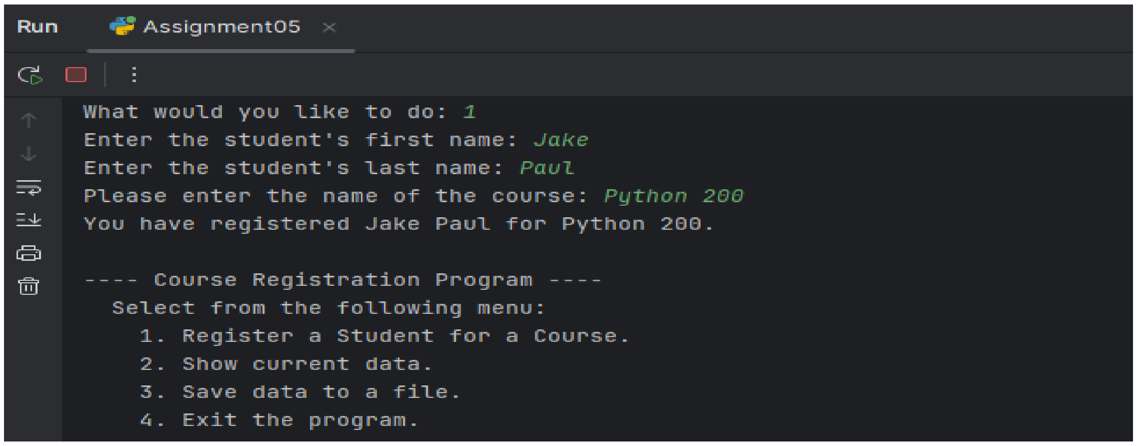
After testing the error handling, I then proceeded to testing the whole script. I started with menu 2.



```
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: 2
--------------------------------------------------
FirstName: Bob LastName: Smith CourseName: Python 100
FirstName: Sue LastName: Jones CourseName: Python 100
--------------------------------------------------
```

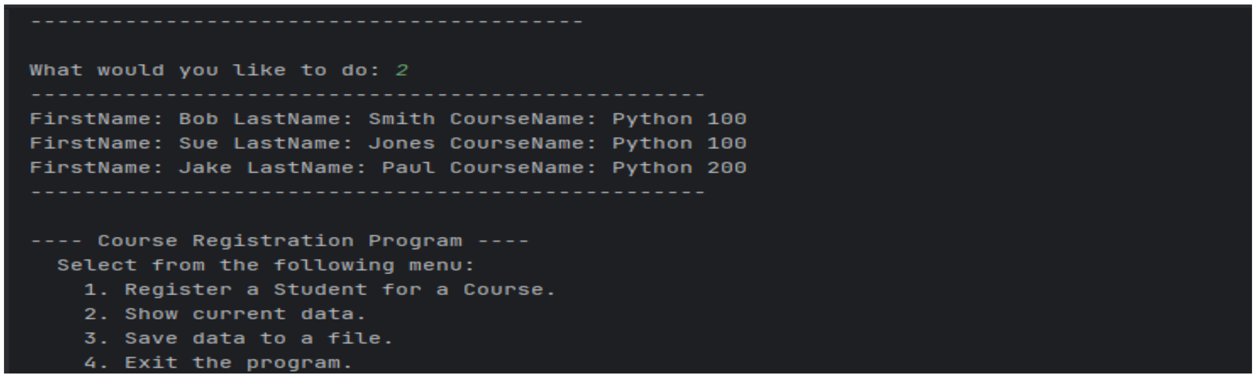
Figure 12. Menu 2 – Showing current data

Then I added one record for the menu 1 and went back to menu 2 to see that the data is added on the list.



```
What would you like to do: 1
Enter the student's first name: Jake
Enter the student's last name: Paul
Please enter the name of the course: Python 200
You have registered Jake Paul for Python 200.

---- 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.
```

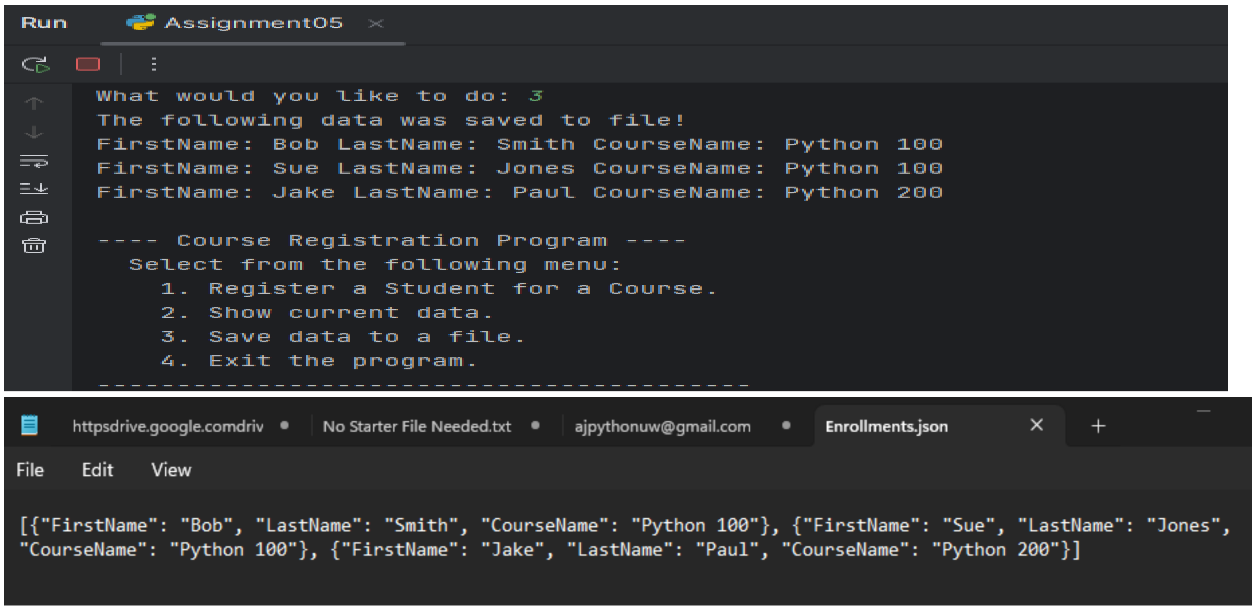
Figure 13. Menu 1 – Entering new record



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

What would you like to do: 2
--------------------------------------------------
FirstName: Bob LastName: Smith CourseName: Python 100
FirstName: Sue LastName: Jones CourseName: Python 100
FirstName: Jake LastName: Paul CourseName: Python 200
--------------------------------------------------

---- 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.
```

Figure 14. Menu 2 – Showing current data

Then chose menu 3 to save the data.

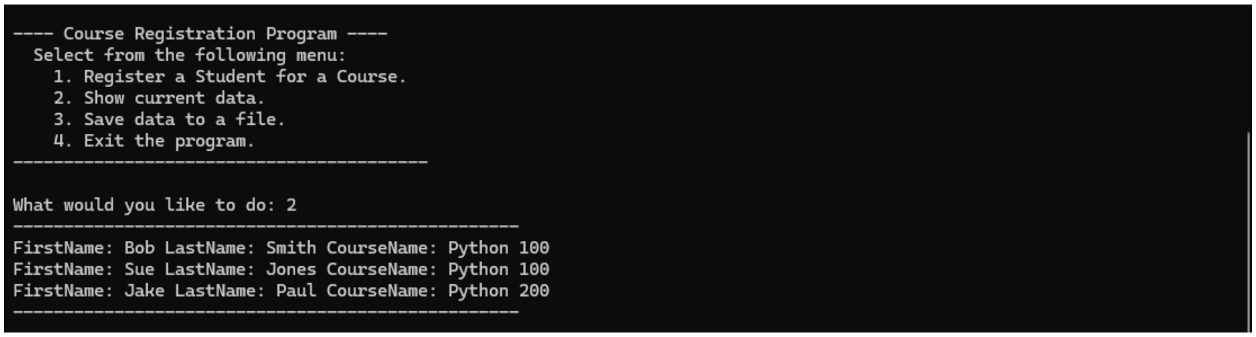


```
What would you like to do: 3
The following data was saved to file!
FirstName: Bob LastName: Smith CourseName: Python 100
FirstName: Sue LastName: Jones CourseName: Python 100
FirstName: Jake LastName: Paul CourseName: Python 200

---- 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.
----------------------------------------
```

```
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"}, {"FirstName": "Sue", "LastName": "Jones",
"CourseName": "Python 100"}, {"FirstName": "Jake", "LastName": "Paul", "CourseName": "Python 200"}]
```

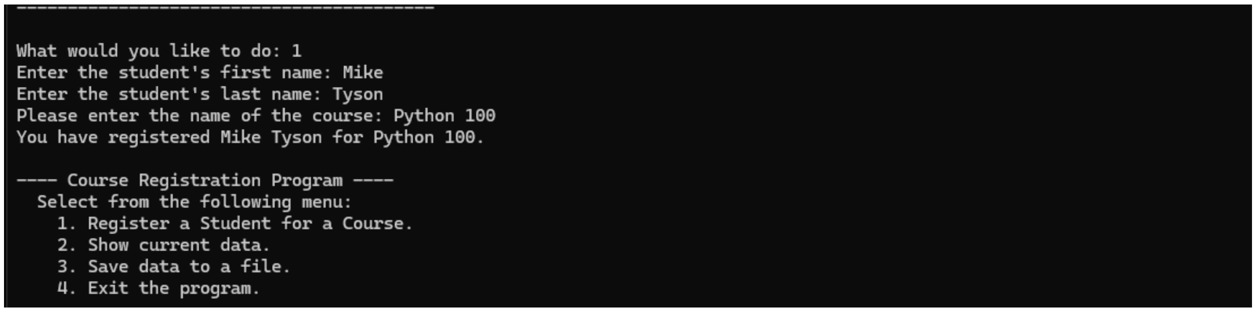
Figure 15. Data added on the json file.

Testing using CMD



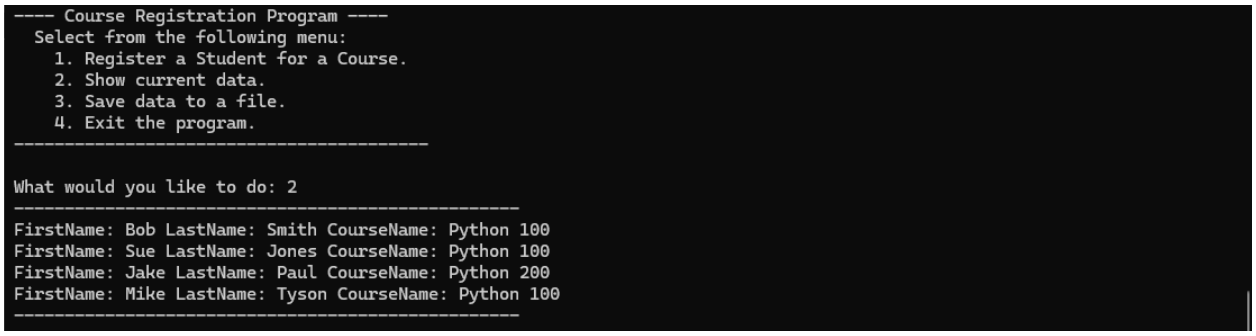
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------

What would you like to do: 2
--------------------------------------------------
FirstName: Bob LastName: Smith CourseName: Python 100
FirstName: Sue LastName: Jones CourseName: Python 100
FirstName: Jake LastName: Paul CourseName: Python 200
--------------------------------------------------
```

Figure 16. Menu 2 – CMD



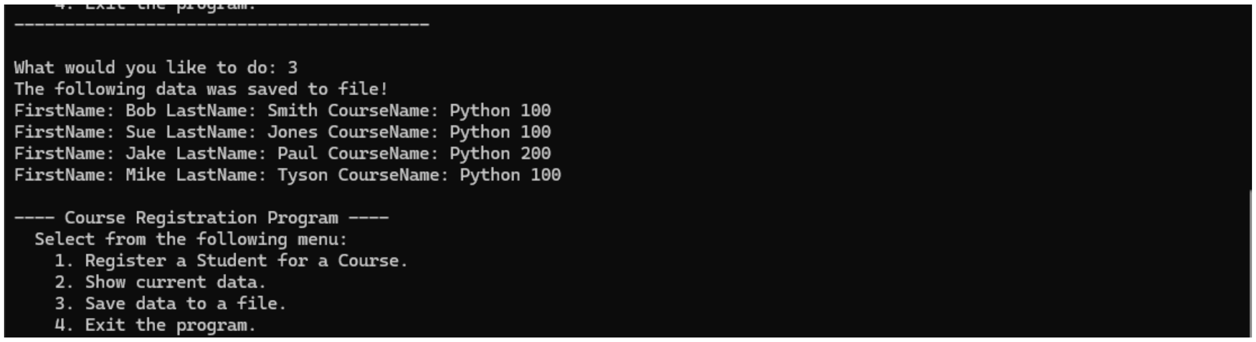
```
-----------------------------------------

What would you like to do: 1
Enter the student's first name: Mike
Enter the student's last name: Tyson
Please enter the name of the course: Python 100
You have registered Mike Tyson for 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.
```

Figure 17. Menu 1 – CMD



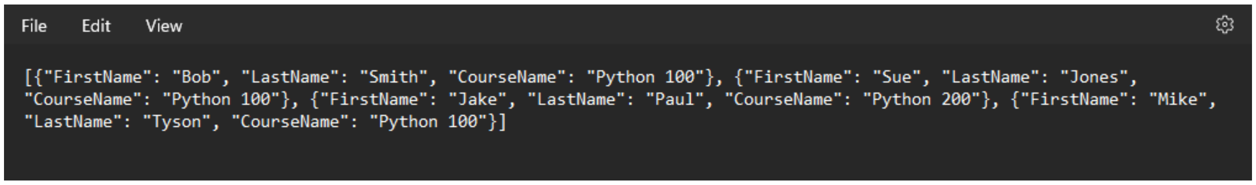
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------

What would you like to do: 2
--------------------------------------------------
FirstName: Bob LastName: Smith CourseName: Python 100
FirstName: Sue LastName: Jones CourseName: Python 100
FirstName: Jake LastName: Paul CourseName: Python 200
FirstName: Mike LastName: Tyson CourseName: Python 100
--------------------------------------------------
```

Figure 18. Menu 2 – CMD



```
    4. Exit the program.
-----------------------------------------

What would you like to do: 3
The following data was saved to file!
FirstName: Bob LastName: Smith CourseName: Python 100
FirstName: Sue LastName: Jones CourseName: Python 100
FirstName: Jake LastName: Paul CourseName: Python 200
FirstName: Mike LastName: Tyson CourseName: 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.
```

Figure 19. Menu 3 – CMD


File Edit View

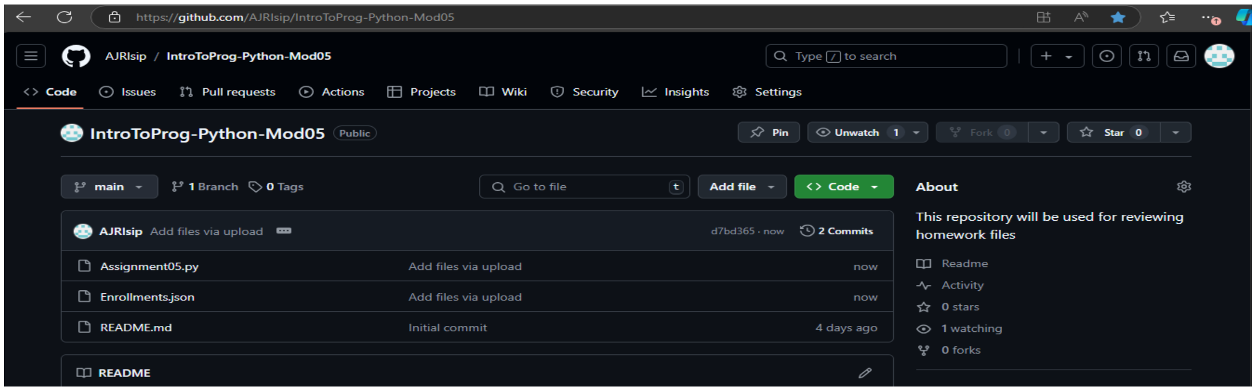
```
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"}, {"FirstName": "Sue", "LastName": "Jones",
"CourseName": "Python 100"}, {"FirstName": "Jake", "LastName": "Paul", "CourseName": "Python 200"}, {"FirstName": "Mike",
"LastName": "Tyson", "CourseName": "Python 100"}]
```

Figure 20. JSON file

Uploading in GitHub

Once I know that the script is working, I then uploaded the python file, knowledge document and the json file to the GitHub repository:

[AJRIsip/IntroToProg-Python-Mod05: This repository will be used for reviewing homework files]



AJRIsip Add files via upload d7bd365 · now 2 Commits

Assignment05.py	Add files via upload	now
Enrollments.json	Add files via upload	now
README.md	Initial commit	4 days ago

README

About

This repository will be used for reviewing homework files

Readme
Activity
0 stars
1 watching
0 forks

Conclusion

To summarize Assignment 05, I was able to learn to read data from a json file and write data to a json file. I was able to learn also the error handling scripts in Python. Also, I was able to create and upload files to GitHub.