Nolan Davis

November 14, 2023

IT FDN 110

Assignment 05

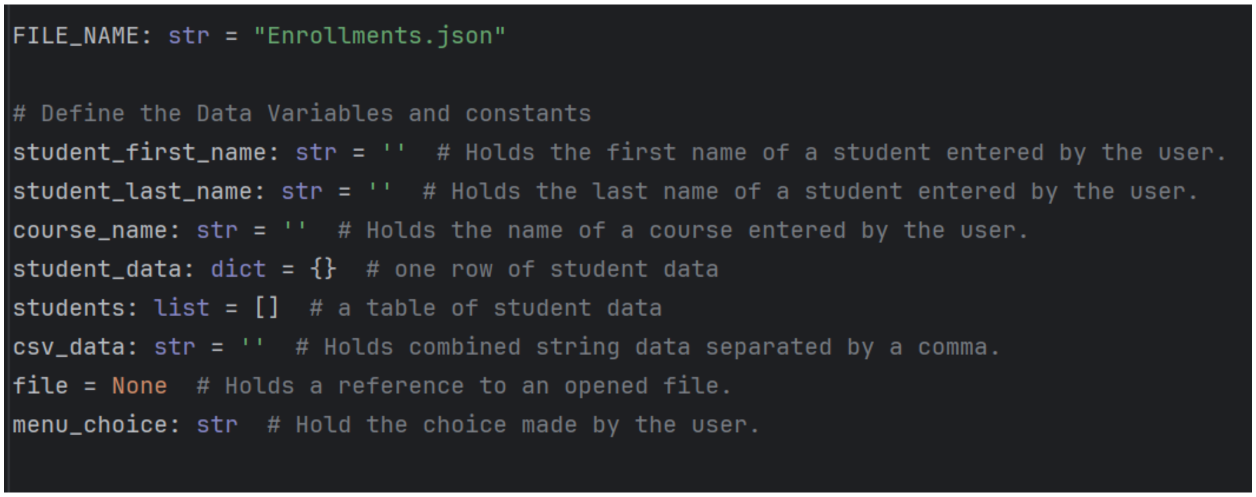
JSON files and Error Handling

Introduction

This week introduced some more advanced concepts like error handling and working with a different data format called JSON. Error handling adds a layer of complexity but also allows for much more control in how the user interacts with the program and being able to limit what they can input in certain circumstances. JSON seems like a very useful data format that includes some very helpful functions in Python to make our lives easier.

Creating the Program

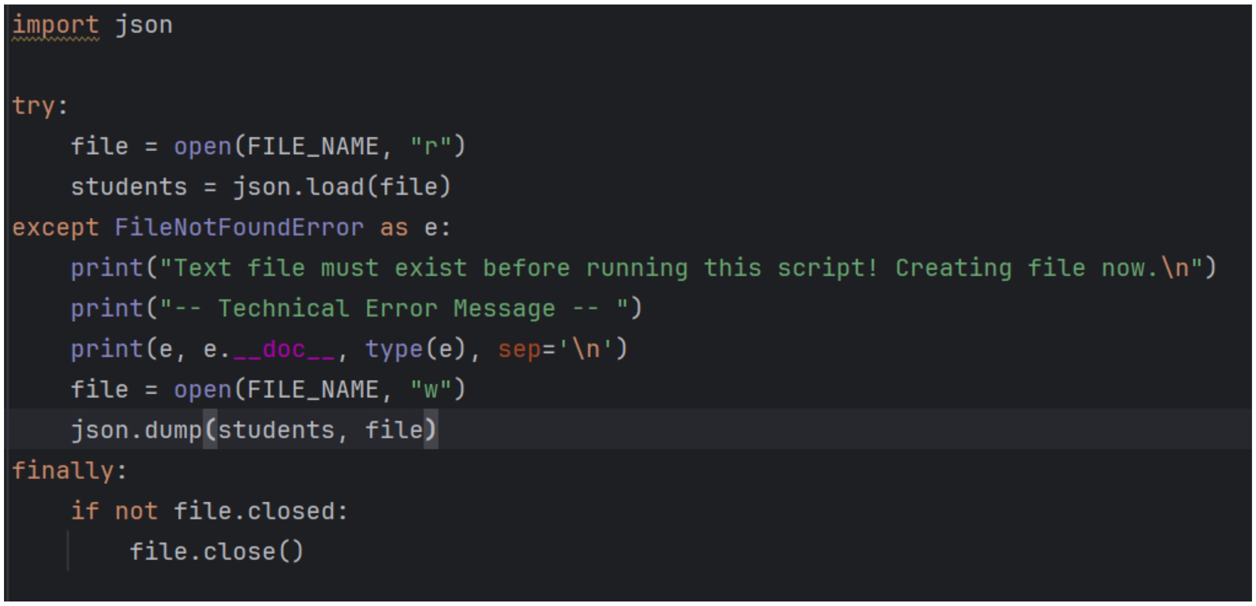
To begin, I updated the starter python file to change the FILE_NAME to be ".json" instead of ".csv". I also had to change the student_data variable to be a dictionary class, and changed the square brackets to curly brackets. See Figure 1.



```
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
csv_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.
```

Figure 1 – Updated FILE_NAME and student_data

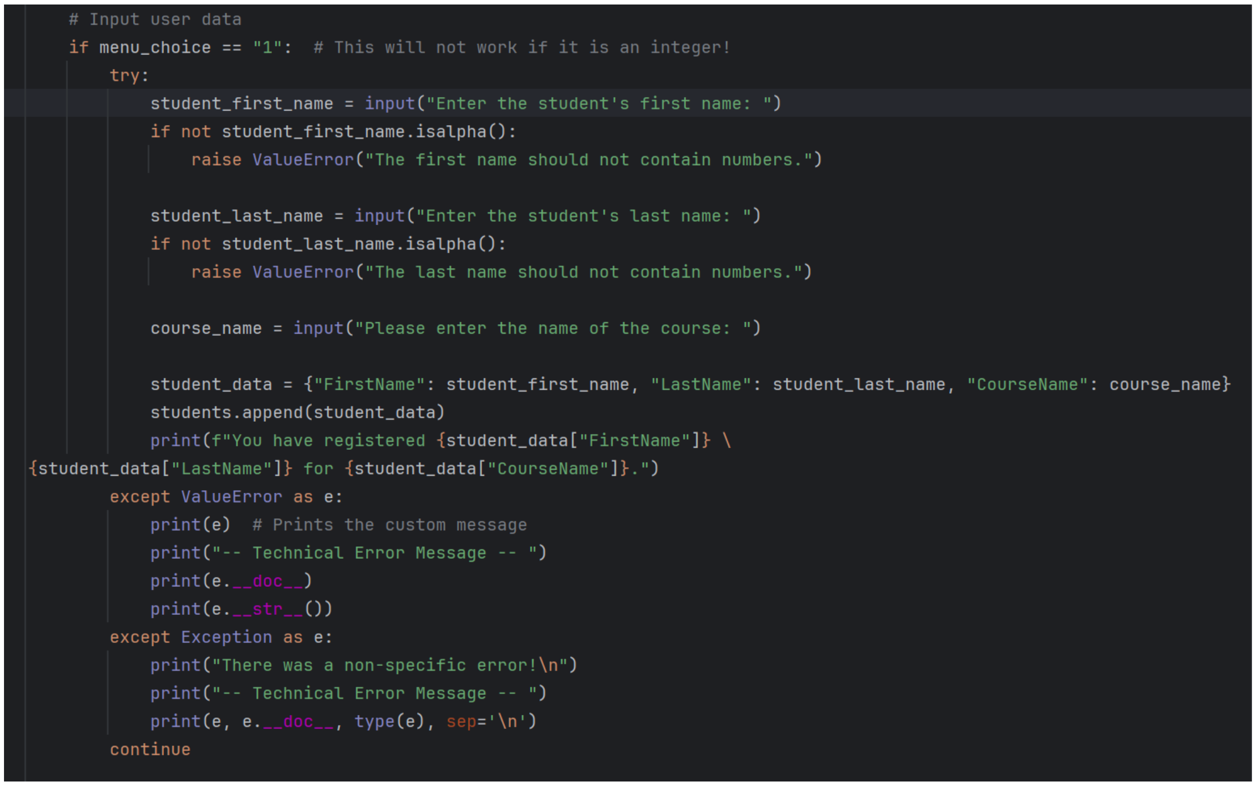
To continue using .json throughout the program I used the "import json" line to import the module. Thanks to the functionality included with that module, I was able to replace the "For" loop to "json.load(file)" to dump all of the dictionary information from the "Enrollments.json" file into my students list. I added a "try:" line and a FileNotFoundError exception to handle the case where "Enrollments.json" file didn't exist, and in that case it opens the File_Name under write() to create it and writes the empty students list into the file. See Figure 2.



```python
import json

try:
    file = open(FILE_NAME, "r")
    students = json.load(file)
except FileNotFoundError as e:
    print("Text file must exist before running this script! Creating file now.\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
    file = open(FILE_NAME, "w")
    json.dump(students, file)
finally:
    if not file.closed:
        file.close()
```

Figure 2 – Reading the .json file and handling FileNotFoundError

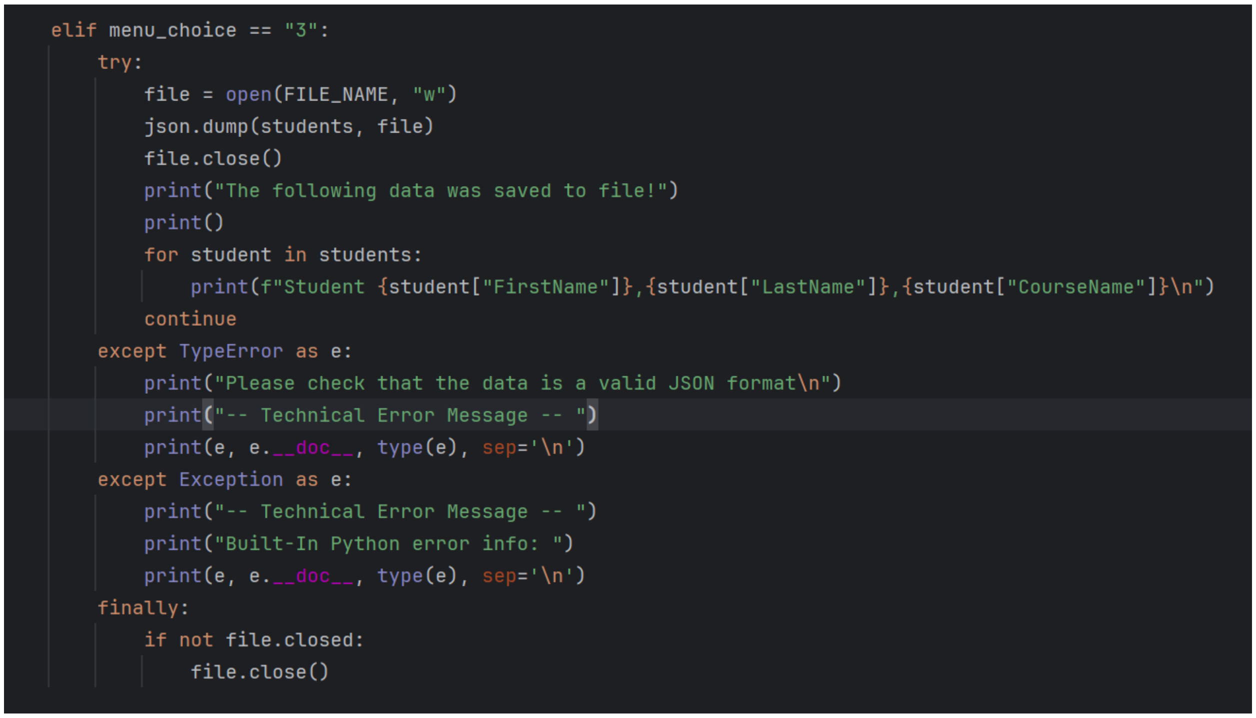
For Menu Choice 1, I added another “try” line to handle the ValueError exception if the user inputs a number into the first or last name of the student. Then I updated student_data to use the keys for a .json file instead of indexes from a list. See Figure 3. I copied the student_data line to update the print() function for Menu Choice 2 as well.



```python
    # 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, "CourseName": course_name}
            students.append(student_data)
            print(f"You have registered {student_data["FirstName"]} \
{student_data["LastName"]} for {student_data["CourseName"]}.")
        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')
        continue
```

Figure 3 – Menu Choice 1 with error handling for user input.

For Menu Choice 3, I copied the same print statement F-String from choice 2 and added error handling for TypeError and general exception like we did the Mod05 labs and demos. See Figure 4.

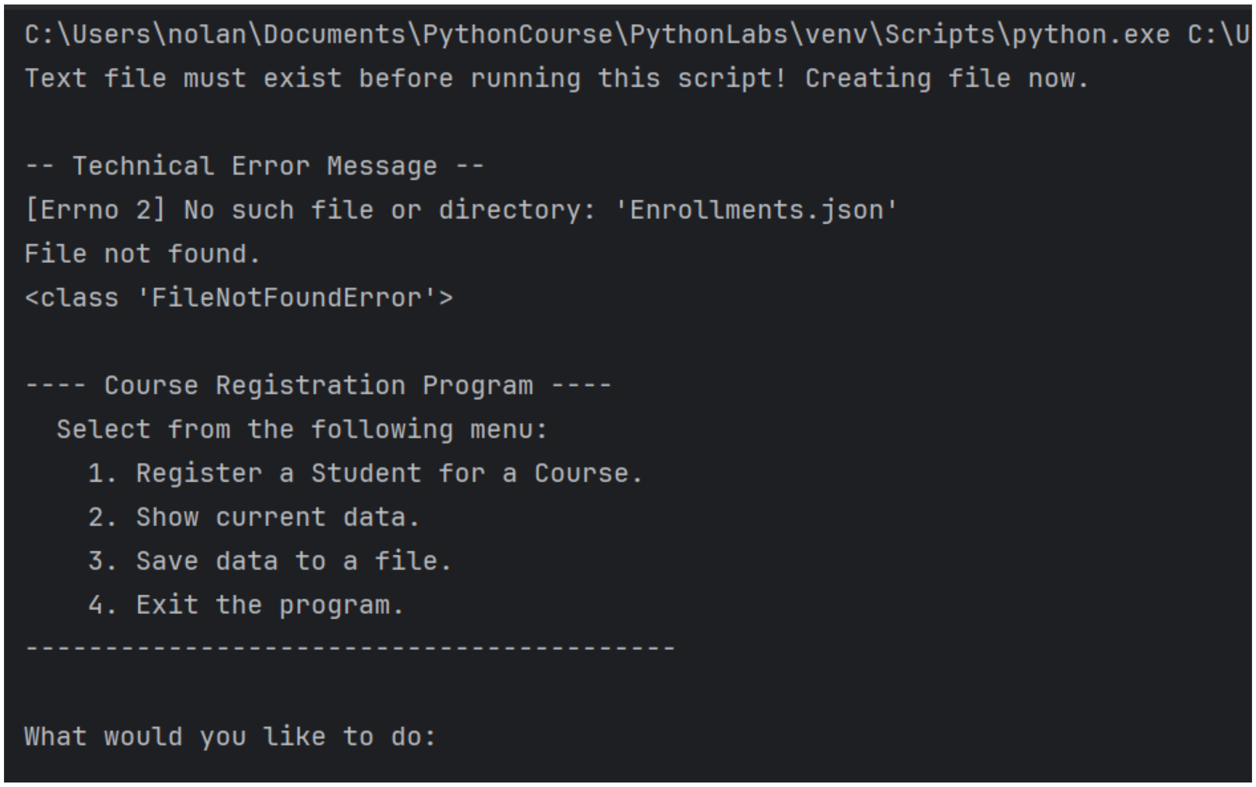


```
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w")
        json.dump(students, file)
        file.close()
        print("The following data was saved to file!")
        print()
        for student in students:
            print(f"Student {student["FirstName"]},{student["LastName"]},{student["CourseName"]}\n")
        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 not file.closed:
            file.close()
```

Figure 4 – Error handling in case data does not match JSON format.

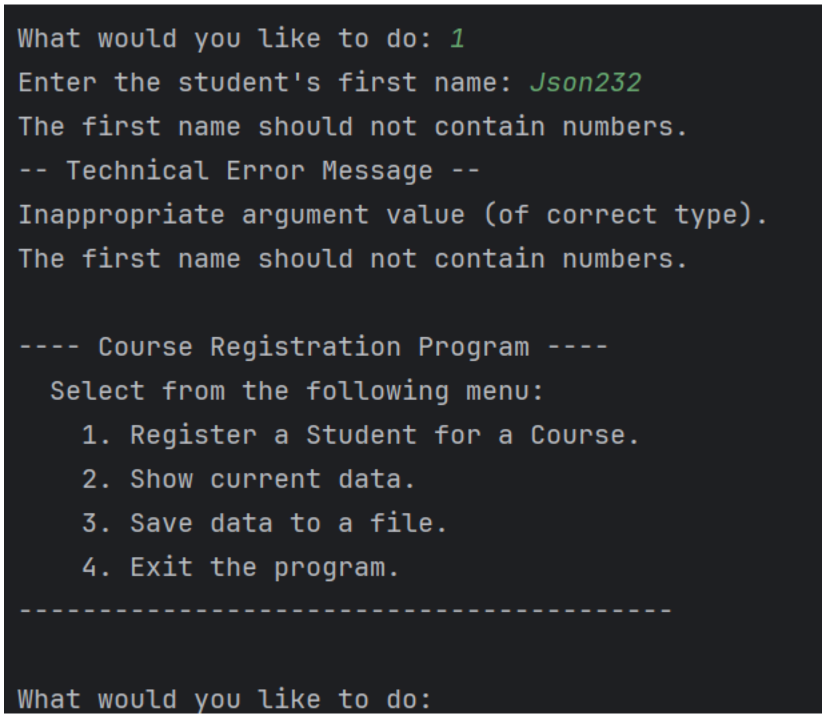
Testing the program

For testing, I ran through the normal procedure of adding a student, verifying the student was added to the list, and then saving to the Enrollments.json file. Once that was done, I focused on verifying that the error handling that was added to the code worked as intended. Figure 5 verified that when I removed the Enrollments.json file from the working folder that the program created a file with the same name. Figure 6 verified that the program would not allow the user to input a number in the first or last name fields for menu choice 1. Figure 7 verified that the data saved correctly for menu choice 3.



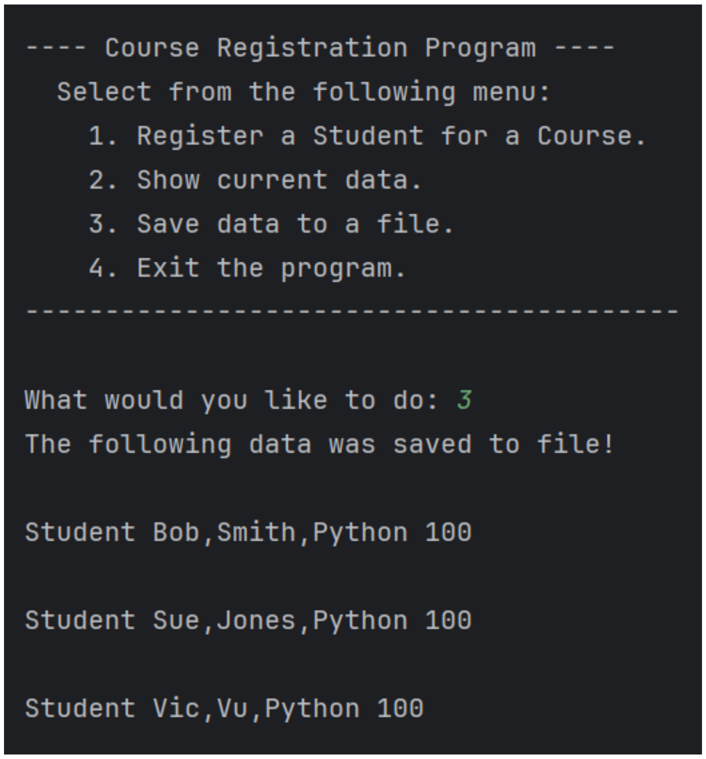
```
C:\Users\nolan\Documents\PythonCourse\PythonLabs\venv\Scripts\python.exe C:\U
Text file must exist before running this script! Creating file now.

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

---- 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:
```

Figure 5 – Error handling for FileNotFoundError, and creating a new file instead.



```
What would you like to do: 1
Enter the student's first name: Json232
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:
```

Figure 6 – Error Handling if user inputs a number in the first name for menu choice 1.



```
---- 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,Python 100

Student Sue,Jones,Python 100

Student Vic,Vu,Python 100
```

Figure 7 – Verifying data saves to Enrollments.json correctly.

Summary

Error handling added a significant amount of length to the program, but it makes up for that in the amount of control that it gives the programmer. Working with JSON files also feels more intuitive using dictionary keys rather than indexes. I can see how much simpler that will become as lists get longer and more complex. This was another steep learning curve week but the functionality of the programs is also becoming more clear and controlled.