Name: Lora Berger
Date: November 12, 2024
Course: IT FDN 110 Foundations of Programming: Python
Assignment 05
GitHub: https://github.com/lab80-code/IntroToProg-Python-Mod05

# Python Course Registration Program – Using JSON and Adding in Exception Handling

## Introduction

This week's assignment takes the registration program we updated last week and changes the use of CSV format to JSON format and file as well as add in error handling.

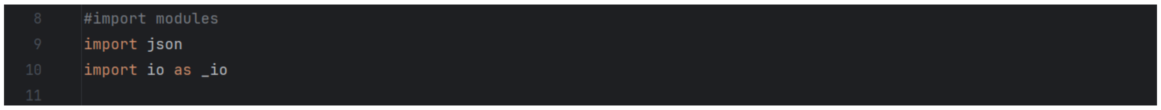
## Lists vs Dictionaries

Lists are commonly used for CSV formatting and are an ordered set of data like a tabular table. When we access the data in lists in Python it is done by the index or position of where it is in the list. Dictionaries though are unordered and utilize a key – value pair to store and access which is the same format as JSON files.

One of the benefits I found during this assignment is that instead of mapping to a position for either the student's first name or last name, I could just name the key and the data was available. This made my script more reader friendly since you see what I am pulling in vs just a number.

## Constants and Variable Changes for JSON and Error Handling

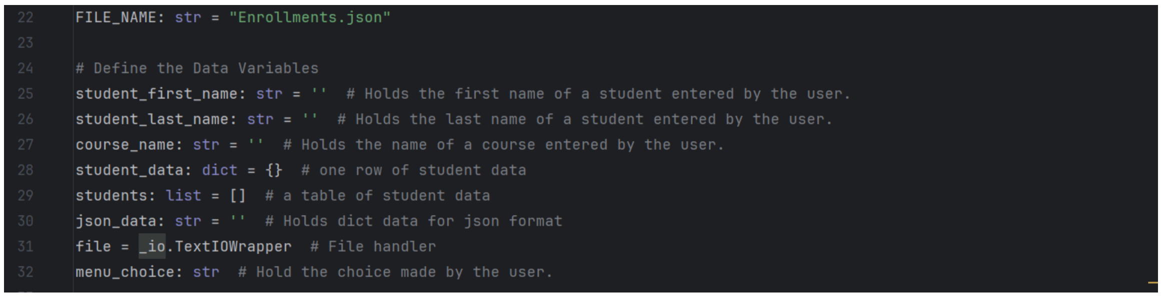
The first change to the script I made was to import the json module which is a tool available specific to working with JSON files. This tool made it easier for me to load as well as to write JSON files. I also imported the io module which will load built-in tools to work with input and output operations. My specific use for this module will be to close files after an error has occurred in either the reading or writing of that file.



```
#import modules
import json
import io as _io

```

Figure 01: screenshot of script showing the modules being imported for use

I then updated my FILE_NAME constant to point to the Enrollments.json file instead of the csv. I also updated my student_data variable definition from list to dictionary and put in the curly brace which is what is used to identify the data for key – value pairs. My final change was to the file variable to instead use the io tool to identify the file itself.



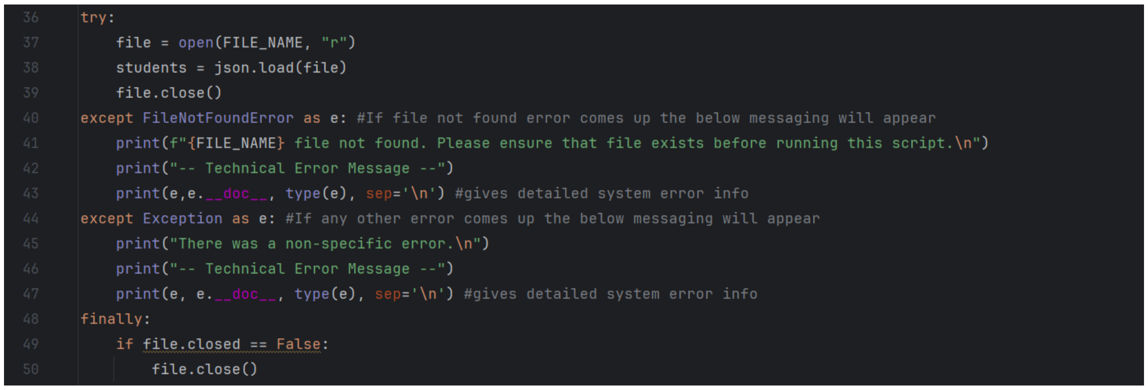
```
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables
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
json_data: str = ''  # Holds dict data for json format
file = _io.TextIOWrapper  # File handler
menu_choice: str  # Hold the choice made by the user.
```

Figure 02: screenshot of script showing constants and variables

## Loading JSON file data with Error Handling

Using the json module, loading data from a json file becomes much easier. Instead of having to identify each element of a line like in the csv file, I only need to use the load command. Once the file is loaded into my students list, I tell the program to close the file.

I wrapped these commands into a try-except block for the error handling portion. The try is what I am asking the program to do, in this case load the json file into my list variable. If an error occurs though the script will move into the except parts of the block. I have called out the specific error of file not found to be able to give the user more information about what the cause of the error may be. When Python is unable to locate the file I specified, it will present a message to the user stating that the FILE_NAME file was not found. If a different error occurs, it will go to the second except statement and give the technical message with documentation and type of message. Once the messages are presented the finally portion of the block will close the file. This code will use the imported io module to complete.



```python
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)
    file.close()
except FileNotFoundError as e: #If file not found error comes up the below messaging will appear
    print(f"{FILE_NAME} file not found. Please ensure that file exists before running this script.\n")
    print("-- Technical Error Message --")
    print(e,e.__doc__, type(e), sep='\n') #gives detailed system error info
except Exception as e: #If any other error comes up the below messaging will appear
    print("There was a non-specific error.\n")
    print("-- Technical Error Message --")
    print(e, e.__doc__, type(e), sep='\n') #gives detailed system error info
finally:
    if file.closed == False:
        file.close()
```

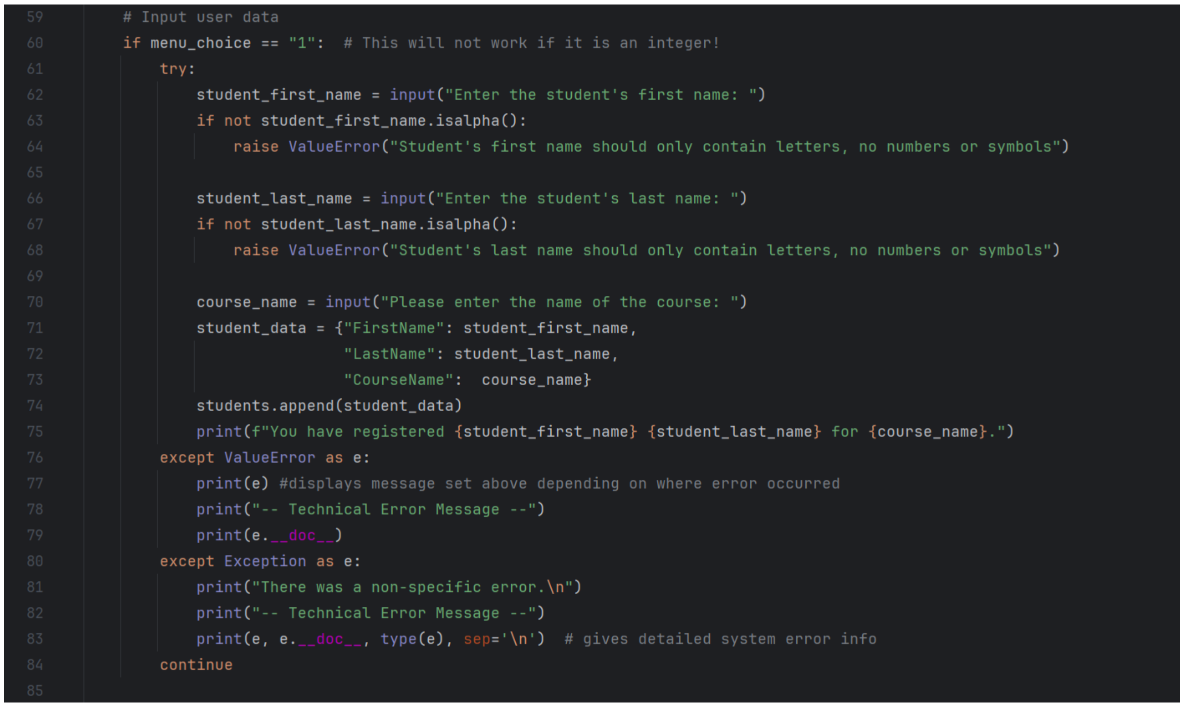
Figure 03: screenshot of json load and error handling try-except block

## Menu Choice Loop Changes for JSON and Error Handling

Choice “1” and “3” sections did have changes to account for using key-value pairs and error handling. Choice “2” had a small change from using list indexing to the key-value pair of the dictionary. Choice “4” had no change to the scripting.

### Choice “1”

The student_data variable was updated to accommodate the key-value pair format. The biggest change though to this section was the error handling. Under the student_first_name input and student_last_name input I put in a statement to check if the data the user provided contained anything other than alphabetic letters. If it did, I put in a statement to raise a ValueError stating that the user should only use letters and go back to the menu to try again. I still put in a catch all except in case a different error occurred so the user could get the information around that error.

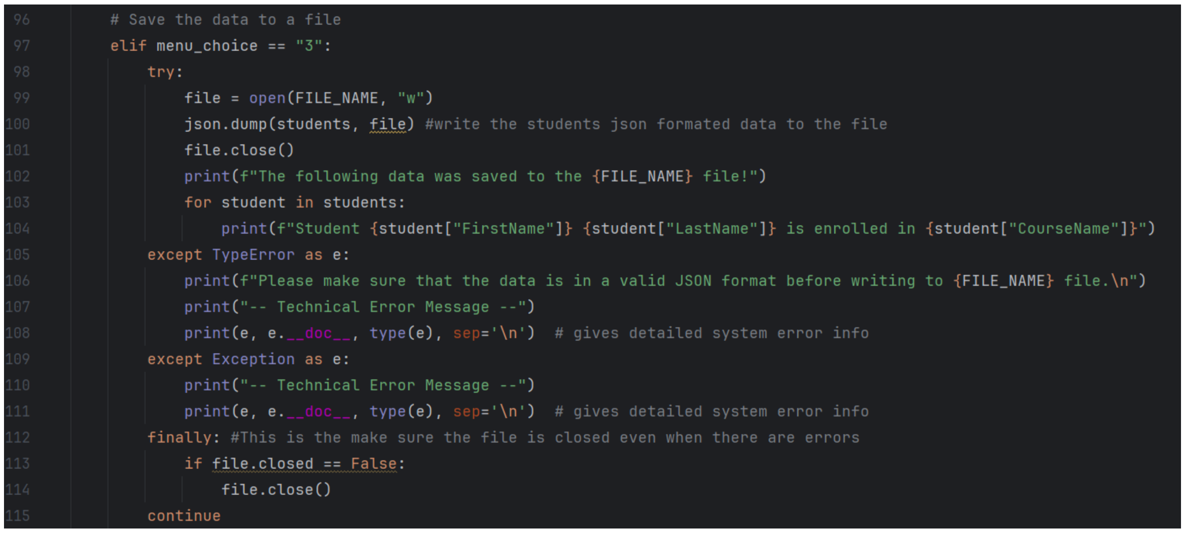


```python
    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        try:
            student_first_name = input("Enter the student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("Student's first name should only contain letters, no numbers or symbols")

            student_last_name = input("Enter the student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError("Student's last name should only contain letters, no numbers or symbols")

            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}.")
        except ValueError as e:
            print(e) #displays message set above depending on where error occurred
            print("-- Technical Error Message --")
            print(e.__doc__)
        except Exception as e:
            print("There was a non-specific error.\n")
            print("-- Technical Error Message --")
            print(e, e.__doc__, type(e), sep='\n')  # gives detailed system error info
        continue

```

Figure 04: screenshot of while loop script first condition

## Choice “3”

Like the loading of the JSON data earlier, writing to a JSON file is just as easy.  Instead of having to iterate through each line to write to the file, I just used the json.dump command and identified the students list as my data to write.  These sections of code I wrapped in a try-except block as well to give detailed error messaging for a possible formatting error or any other error that may occur.  The finally part of the block closes the open file so it does not generate errors later in the script.

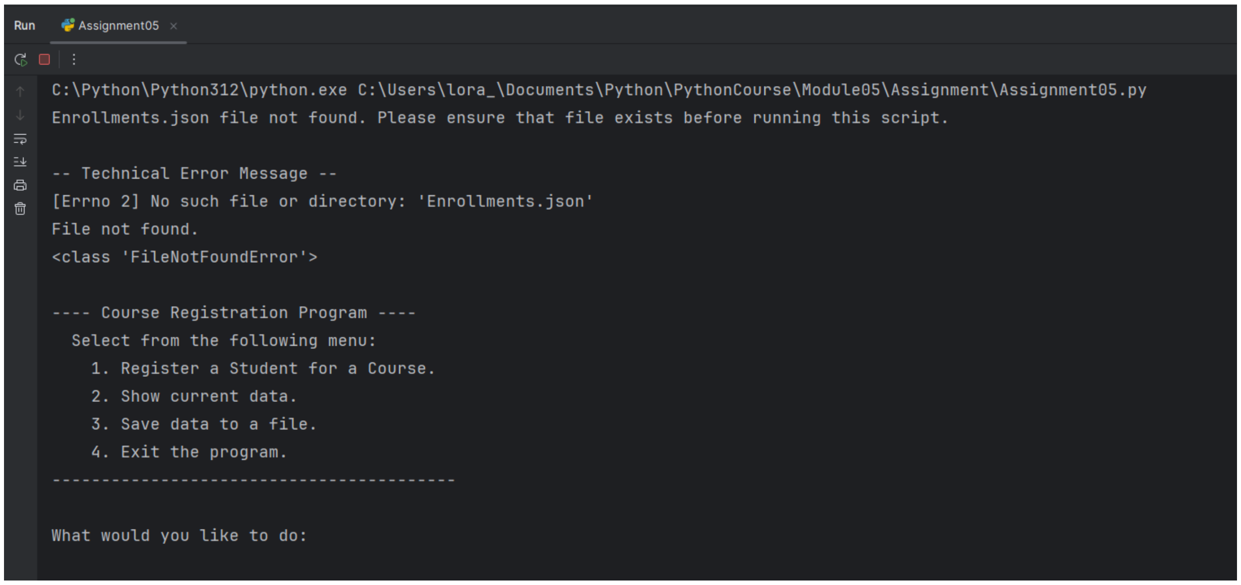


```python
    # Save the data to a file
    elif menu_choice == "3":
        try:
            file = open(FILE_NAME, "w")
            json.dump(students, file) #write the students json formated data to the file
            file.close()
            print(f"The following data was saved to the {FILE_NAME} file!")
            for student in students:
                print(f"Student {student["FirstName"]} {student["LastName"]} is enrolled in {student["CourseName"]}")
        except TypeError as e:
            print(f"Please make sure that the data is in a valid JSON format before writing to {FILE_NAME} file.\n")
            print("-- Technical Error Message --")
            print(e, e.__doc__, type(e), sep='\n')  # gives detailed system error info
        except Exception as e:
            print("-- Technical Error Message --")
            print(e, e.__doc__, type(e), sep='\n')  # gives detailed system error info
        finally: #This is the make sure the file is closed even when there are errors
            if file.closed == False:
                file.close()
        continue
```

Figure 05: screenshot of while loop script third condition

## Testing Program

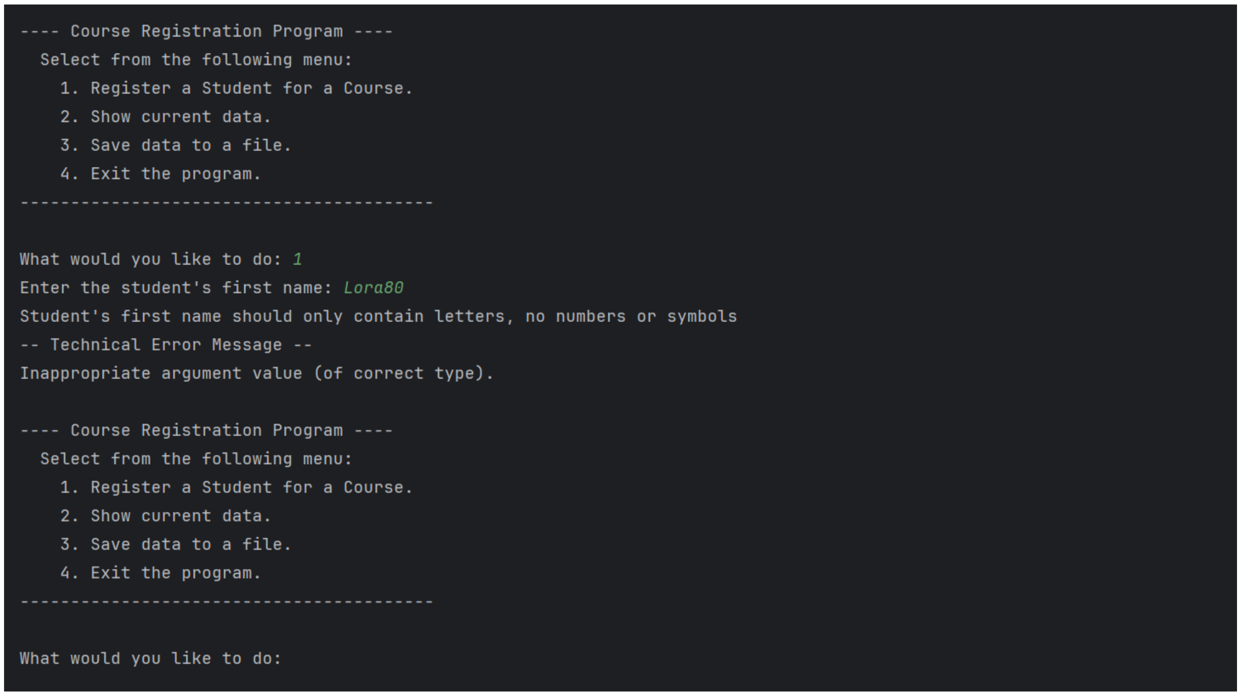
This time when I tested my program I also deliberately caused errors to test the error handling.  My first test was to rename my Enrollments.json file to 1Enrollments.json effectively removing the file.  When I started my program, I immediately received my first successful error handling messaging.



```
Run   Assignment05

C:\Python\Python312\python.exe C:\Users\lora_\Documents\Python\PythonCourse\Module05\Assignment\Assignment05.py
Enrollments.json file not found. Please ensure that file exists before running this script.

-- 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:
```

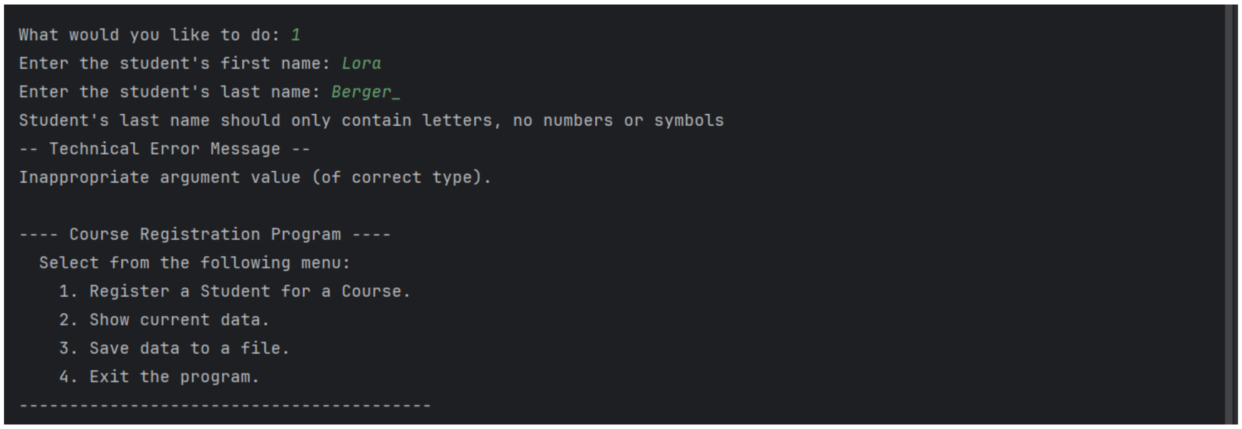
Figure 06: screenshot of testing file not found error in PyCharm

I ended the program, renamed the file back to Enrollments.json and then reran my script.  This time there was no error message and got my menu.  I input 1 and tested out putting in numbers or symbols into both the first name and last name inputs.  Each time I got the expected message and return to menu.



```
---- 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: Lora80
Student's first name should only contain letters, no numbers or symbols
-- Technical Error Message --
Inappropriate argument value (of correct type).

---- 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 07: screenshot of testing error handling for choice 1 first name



```
What would you like to do: 1
Enter the student's first name: Lora
Enter the student's last name: Berger_
Student's last name should only contain letters, no numbers or symbols
-- Technical Error Message --
Inappropriate argument value (of correct type).

---- 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 08: screenshot of testing error handling for choice 1 last name

I chose 1 again but this time I put in the name properly.  I ran through the rest of the selections and confirmed that my registration was presented by choice 2 then written to the file in choice 3.

## Summary

In conclusion, changing from using CSV formatting and lists to the JSON file formatting and dictionaries has made the script more efficient and easier to read due to the key-value pairs vs index numbers.  Adding in the error handling has made it possible to ensure data validity and make it possible to give easier to understand error messaging than what is built into Python.  In addition, by putting in the try-except blocks I made it so the program will not stop at the first error in the script.