Sashi Chintala

November 17th, 2024

Course name: IT FDN 110 A Au 24: Foundations Of Programming: Python

Assignment 05 – Advanced Collections and Error Handling

GitHub: https://github.com/SashiChintala/IntroToProg-Python-Mod05

Advanced Collections and Error Handling

Introduction

In this module, I learned about the differences between using a list and a dictionary when working with .csv and .json files. I will share my learnings about the .json files basics structure and how the data is organized using the curly braces {} for objects and square brackets [] for arrays, and how its more human-readable compared to .csv files. Notes helped me understand how to fix the bugs in the code using Try-Except block and to give a smooth experience to the end users instead of breaking the program due to an error. Lastly, I learned about sharing code files for team collaboration and how it helps to build the code when working in a team.

Creating the program

Thanks to Professor Root as he gave us the starter script from previous module 04 assignment and I started building it, adding two things. In our module03 and 04 program from the previous week, we took student information and used it for student registration. Information regarding interactive console using the prompt and using the four menu choices.

1. Collect user input using the menu choice 1 and register the student for a course
2. Menu choice two shows the information
3. Menu choice three takes care of writing or saving the information to the .json file
4. And menu choice 4 to end the program

Constants

We are using two constants for our program,

1. **Constant MENU: str** is set to the value:
   ---- 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.

   ---
2. **FILE_NAME: str** and is set to the value “Enrollments.json” to save information to the csv file

Figure 1.1: *Constants*

```
# 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"
```

Variables

Please see Figure 1.2. which tells what all variables are used in our program mostly similar to module 04. Student_first_name to store the student's first name, student_last_name to store the student's last name. Course name, json_data, menu_choice are set to empty string. File is an object and is initialized to 'None', and as of this point, it's not pointed to any file as of now. To remove the errors to read the file, I added the data to enrollments.json file. We will track all the students and individuals using the students multidimensional list as we have a list of strings for students, which includes first name, last name and course name.

***Figure 1.2**: list of variables used*

```
# 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 [str,str] = {}  # one row of student data
students: list = []  # a table of student data
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.
parts: list[str]
```

Reading from a file:

The program will start opening the file enrollments.json to hold existing student data, and will read all the lines the file processed by splitting the comma-separated values to the respective columns first_name, last_name, course_name and saved to a dictionary. If the files doesn't exist we will create a new file. Initially the enrollments.json is set to empty string as shown in figure 1.2. But each line in the files will be saved into a dictionary and added to the students list. Data is saved back

.json file. Students variable is a list and it saves first and last name of the student along with the course name data dictionaries. Variable student_data holds the data temporarily before its added to the students list. We will use for loop to read the rows from the file object. We are using a list of string parts to split the rows for students first, last, and course name.

```
try:
    file = open(FILE_NAME, "r")
    for row in file.readlines():
        # Transform the data from the file
        parts = row.strip().split(',')
        student_first_name = parts[0]
        student_last_name = parts[1]
        course_name = parts[2]
        student_data = {'first_name': student_first_name,'last_name': student_last_name,'course_name': course_name}
        # Load it into our collection (list of lists)
        students.append(student_data)
except FileNotFoundError:
    print('File not found. Creating a new file...')
    open(FILE_NAME, "w")
except Exception as e:
    print('Unknown exception. Resetting roster...')
    students = []
    print(type(e), e, sep='\n')
finally:
    if file and not file.closed:
        file.close()
```

Figure 1.3 *Reading from a file*

Error handling

In this module we learned how to handle the errors using the try-except blocks. Please see Figure 1.3, we will create a new file in the enrollments.json doesn't exist and to move further we continue the program by not stopping it. Also, we will validate the student inputs to be alphabetic and will display a message requesting user to enter only alphabets using the str. Isalpha().

```
# 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("Student first name must be alphanumeric")
        student_last_name = input("Enter the student's last name: ")
        if not student_last_name.isalpha():
            raise ValueError("Student last name must be alphanumeric")
        course_name = input("Please enter the name of the course: ")
        student_data = {'first_name': student_first_name,'last_name': student_last_name,'course_name': 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)
```

Figure1.4 *error handling*

The program also shows how to collect input from students and double checks for errors before writing to the file. We will ensure only data is accepted for students by first name and last name as alphabetical and not numeric. Used a while true loop and it executes the program repeatedly till the condition is met with the menu choices. We use input prompts enabling users to enter the information from menu choice 1-4 till we use option 4 to break the program.

Processing

1. On menu choice 1, the program prompts the user to enter the student's first name and last name, followed by the course name, using the input () function and stores the inputs in the variables (student's first name and last name, followed by the course name). The print function will display the student registration details.
2. On menu choice 2, the presents a coma-separated string by formatting the collected data using the print () function. We are using a loop for getting student data from students multidimensional list for first name, last name, and course name, and print the data.
3. On menu choice 3, program opens the file named “Enrollments.json” in write mode using the open () function. It writes the content of the json_data variable to the file using the write() function, then file is closed using the close() method. Then displays what was stored in the file. Using \n we will write the data in the new line instead of deleting the existing. Similar to the above step, we will use for loop for write the student details to the file for multiple students.

4. On the menu, 4, program prints the message that we are ending the program and stops the while loop and ends the program.

We are using elif or else if statements for menu choices, when choice 1 is not met, elif statement for choice 2 becomes true and prints the data, and same with choice 3, 4. When none of the conditions are met which are not 1-4 choice options, it executes the code in ‘else’ block and takes action to exit the program.

Figure 1.5 and 6: *shows input and output () functions, menu choices 1-4, try-except block and error handling*

Assignment05.py

```
64
65      # Input user data
66      if menu_choice == "1":  # This will not work if it is an integer!
67          try:
68              student_first_name = input("Enter the student's first name: ")
69              if not student_first_name.isalpha():
70                  raise ValueError("Student first name must be alphanumeric")
71              student_last_name = input("Enter the student's last name: ")
72              if not student_last_name.isalpha():
73                  raise ValueError("Student last name must be alphanumeric")
74              course_name = input("Please enter the name of the course: ")
75              student_data = {'first_name': student_first_name,'last_name': student_last_name,'course_name': course_name}
76              students.append(student_data)
77              print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
78          except ValueError as e:
79              print(e)
80
81      # Present the current data
82      elif menu_choice == "2":
83
84          # Process the data to create and display a custom message
85          print("-"*50)
86          for student in students:
87              print(f"Student {student['first_name']} {student['last_name']} is enrolled in {student['course_name']}")
88          print("-"*50)
89          continue
90
91      # Save the data to a file
92      elif menu_choice == "3":
93          try:
94              file = open(FILE_NAME, "w")
95              for student in students:
96                  json_data = f"{student['first_name']},{student['last_name']},{student['course_name']}\n"
97                  file.write(json_data)
98              file.close()
99              print("The following data was saved to file!")
100             for student in students:
101                 print(f"Student {student['first_name']} {student['last_name']} is enrolled in {student['course_name']}")
102         except Exception as e:
103             print('Error saving data to the file')
104         continue
```

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

print("Program Ended")
```

Testing the program

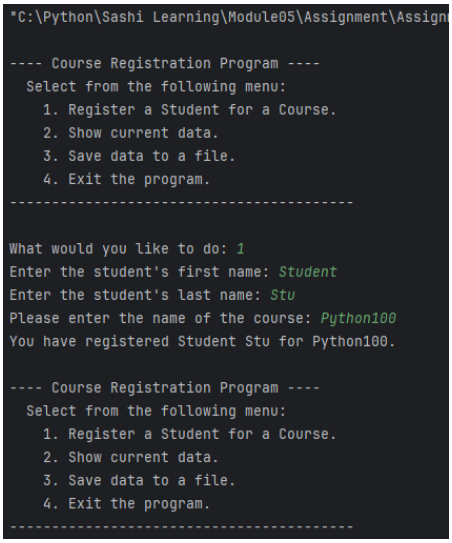
- Please see the figures 1.5 and 1.6, by selecting the menu choice 1, our program takes the user's input for a student's first, last name, and course name and updates to dictionaries instead of lists and ensure the keys or variables are consistent/
- By selecting the menu choice 2, our program displays the user's input for a student's first, last name, and course name.

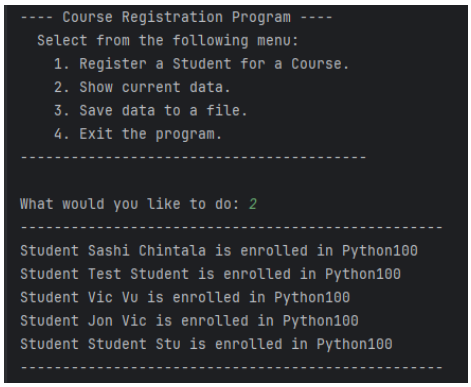
- By selecting the menu choice 3, our program saves the user's input for a student's first, last name, and course name to the json file to reflect the keys
- By selecting the menu choice 4, our program exists.
- Our program allows us to register or display or save multiple student registrations with first name, last name, and course name as shown in the results screenshots.
- We can run in PyCharm and from the command prompt console

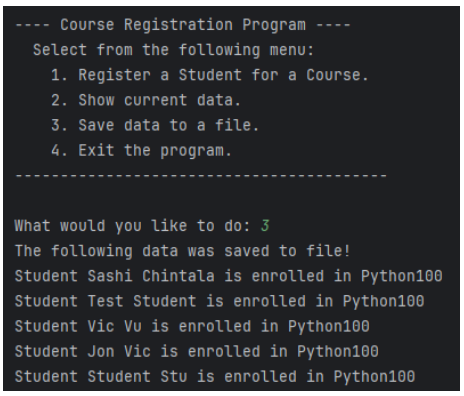
Result from IDLE

Results from PyCharm IDE for menu choice 1-4 from the two screenshots, and json output.

Figures 2.1 -2.4 *– menu choice 1 -4, registers students, displays, writes, and exits*







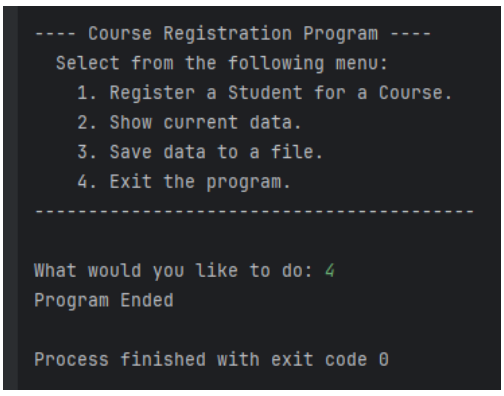
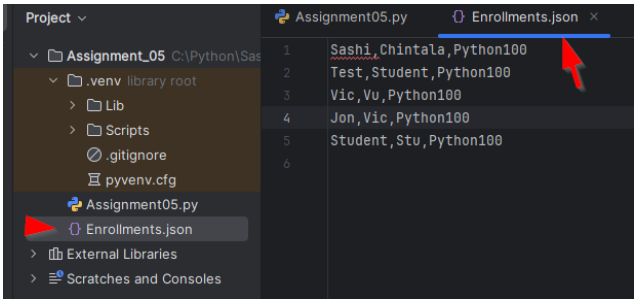


Figure 2.5 shows the enrollments.json file data



Figiure 2.6 shows the message when user enters numeric, instead of stopping the program we will give an opportunity to the user to enter again. This is the new thing I learned from this module.

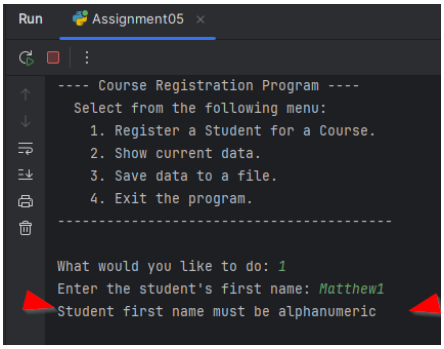
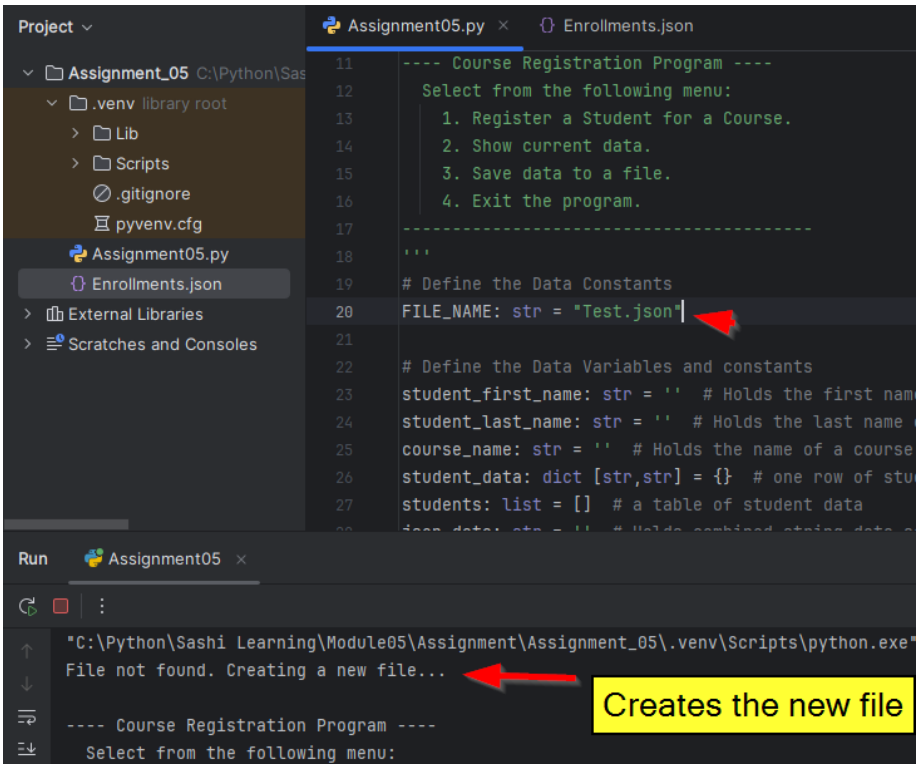


Figure 2.7 File not found error helps when a file doesn't exist, but it creates a new file to continue with the program.



Testing the program from Command prompt on Windows

I was able to test the same from command prompt.

Summary:

I enjoyed learning the differences between the JSON and CSV files and their real time use cases. I am grateful for the notes and videos shared by Professor Root Learned the new way how developers handle the errors using the try-except block and it helped in writing the program, in PowerShell we use similar concept try and catch. Lastly, I got to know how to share the code in GitHub and create the repository. Though its overwhelming to learn all these new topics I am trying to embrace the learning experience, it's a quote I read somewhere 😊