Anthony Cabral

11/15/2024

IT FDN 110 A Au 24

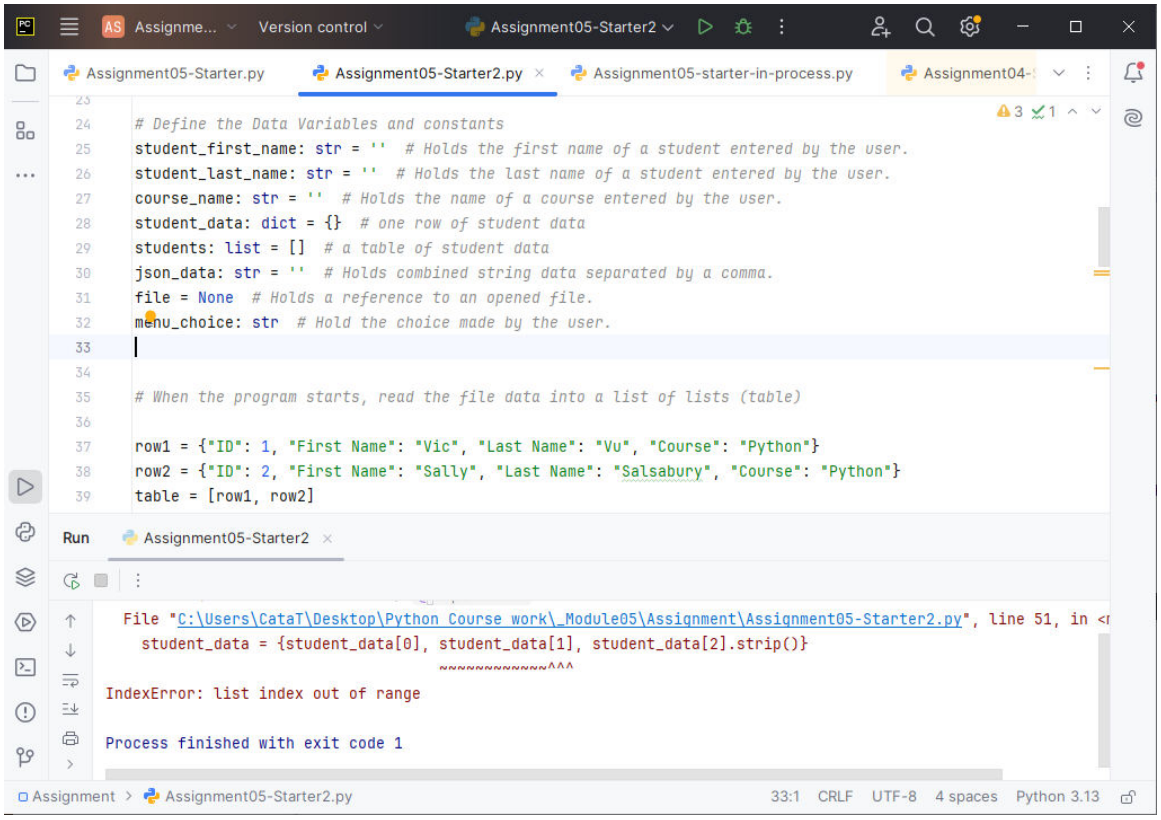
Module 5

# Module 5 Breakdown

## Process Introduction

Choice Adding correct variables and constants. Remove trace of CSV from the script and replace with JSON. Imported json at the start of the script.

I restarted the class. On my own, as in Ive redone the course modules because Im not getting as good a grip as I thought I had by this module. The last 3 I mean. Anyways heres module 5, it starts off really bad.




This part makes me a little hesitant because I'm adding a list of dicts and writing them to the JSON file. Just so the program can run.

```python
# When the program starts, read the file data into a list of lists (table)

row1 = {"ID": 1, "First Name": "Vic", "Last Name": "Vu", "Course": "Python"}
row2 = {"ID": 2, "First Name": "Sally", "Last Name": "Salsabury", "Course": "Python"}
table = [row1, row2]

file = open("data.json", 'w')
json.dump(table, file)
file.close()


# Extract the data from the file
file = open(FILE_NAME, "r")
for row in file.readlines():
    # Transform the data from the file
```

```
  File "C:\Users\CataT\Desktop\Python Course work\_Module05\Assignment\Assignment05-Starter2.py", line 51, in <
    student_data = {student_data[0], student_data[1], student_data[2].strip()}
                                     ~~~~~~~~~~~~^^^
IndexError: list index out of range

Process finished with exit code 1
```

So now Ive added it as suggested by the assignment. But the instructions say "Make sure to put some starting data into the file or you will get an error!" So starting data placed inside the json file? Or the script adding starting data, aka the list of dicts? Seemed unclear, so I figured I'd do both.

```python
row1: dict[str, str] = {"ID": 1, "First Name": "Vic", "Last Name": "Vu", "Course": "Python"}
row2: dict[str, str] = {"ID": 2, "First Name": "Sally", "Last Name": "Salsabury", "Course": "Python"}
students = [row1, row2]

file = open("data.json", 'w')
json.dump(students, file)
file.close()


# Extract the data from the file
file = open(FILE_NAME, "r")
for row in file.readlines():
    # Transform the data from the file
    #student_data = row.split(',')
    #student_data = {student_first_name, student_last_name, course_name.strip()}
    # Load it into our collection (list of lists)
```

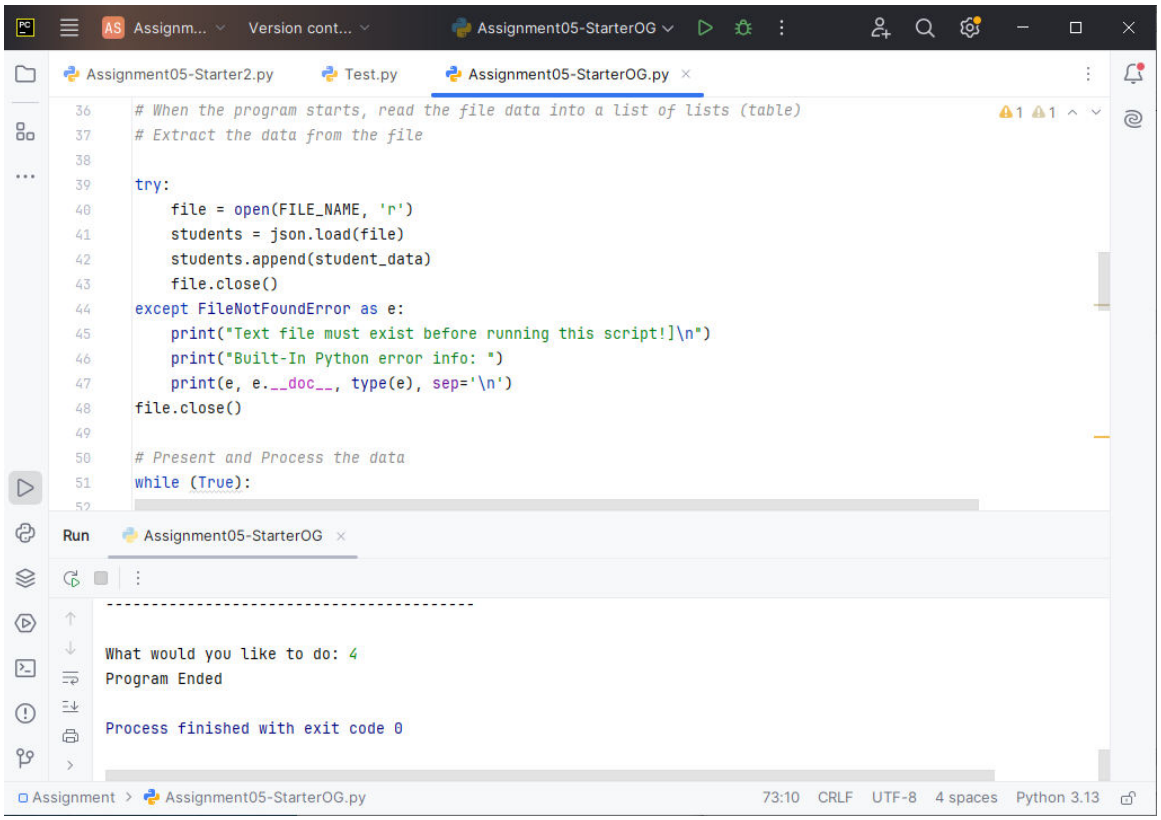
Problems | File 9 | Project Errors | Server-Side Analysis New | Vulnerable Dependencies

Assignment05-Starter2.py C:\Users\CataT\Desktop\Python Course work_Module05\Assignment 9 problems

- Redeclared 'students' defined above without usage :39
- Redeclared 'file' defined above without usage :41
- Expected type 'SupportsWrite[str]', got 'TextIO' instead :42
- Expected type 'dict', got 'set[str]' instead :68
- Redeclared 'student' defined above without usage :86
- Redeclared 'student' defined above without usage :87
- Expected type 'SupportsWrite[str]', got 'TextIO' instead :88

Also went back and wrote the dict lines correctly. String/string, as opposed to string float. Makes sense. Opens the file, dumps it to json, closes. Now I can add a try/except line and change the write to read as the assignment suggests.

Ok so the code has changed and I've added my first try. First name and last name have been added as key values, as has course. Student data is correctly formatted. I hope. It works. And then it appends the data. Now as for the "try":




I used the format from the labs:

```
except Exception as e:
    print("Error! Please check you are not dividing by zero.\n")
    print("-- Technical Error Message -- ")
    print(e)          # Print the exception object (typically includes the error message)
    print(type(e))    # Print the type of the exception object
    print(e.__doc__)  # Print the documentation string of the exception type
    print(e.__str__())
```

Using the file not found handler. This example shows the zero error, because I found it redundant to share the exact same code I just used. File not found seemed most appropriate here.

---

Ok so I looked back and realized the dicts at the start were pointless. I've removed them. The start has the correct formatting and it works now. The earlier screenshots look messy. This is much better. I had to fix everything. The Json file kept throwing errors, probably because it didnt NOT have starting data, the starting data just wasnt correctly formatted. I had to restart from scratch twice. Now its working again.

```python

import json

# 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.
student_data: dict = {}  # 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.


# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file

try:
    file = open(FILE_NAME, 'r')
    students = json.load(file)
    students.append(student_data)
    file.close()
except FileNotFoundError as e:
    print("Text file must exist before running this script!]\n")
```

Im going straight to option 3 to add the dump function. On the way there I added the dictionary keys and rewrote them about 200 times, getting keyErrors while doing so. It was very fun. Heres the result:

```python
    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!
        student_first_name = input("Enter the student's first name: ")
        student_last_name = input("Enter the student's last name: ")
        course_name = input("Please enter the name of the course: ")

        #Dictionary keys

        student_data = {"FirstName": student_first_name, "LastName": student_last_name, "CourseName": course
        students.append(student_data)
        print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
        continue

    # Present the current data
    elif menu_choice == "2":
        print("=" *50)
        #student_data = {"FirstName": student_first_name},{"LastName": student_last_name},{"CourseName": cou
        print(f"Student {student_first_name} {student_last_name} is registered for {course_name}.")
        print("=" *50)
        continue

        # Process the data to create and display a custom message
        #print("-"*50)
        #for student in students:
```

I watched and rewatched the lab and module videos to find out why I kept getting keyErrors, and Im still not sure why. I hope this can be elaborated on for my review. Anyways here is the option 3 requirement:

```python
        #print("-"*50)
        #for student in students:
            #print(f"Student {student[0]} {student[1]} is enrolled in {student[2]}")
        #print("-"*50)
        #continue

    # Save the data to a file
    elif menu_choice == "3":
        file = open(FILE_NAME, "w")
        json.dump(students, file)
        file.close()
        print("The following data was saved to file!")
        for student in students:
            print(f"Student {student_first_name} {student_last_name} is registered for {course_name}.")
        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")
```

## Action Time

Now I need error handling.

```python
    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("The last name should not contain numbers.")
        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("Built-In Python error info: ")
        print(e, e.__doc__, type(e), sep='\n')
        try:
            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.")
        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")
```

I added the error handling. Just for the first name and last name. Alphabetical characters only. That was much easier to parse then the KeyErrors.

Adding the last error handling:

```python
        #continue

    # Save the data to a file
    try:
        elif menu_choice == "3":
        file = open(FILE_NAME, "w")
        json.dump(students, file)
        file.close()
        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()

            print("The following data was saved to file!")
            for student in students:
                print(f"Student {student_first_name} {student_last_name} is registered for {course_name}.")
            continue

    # Stop the loop
```

Checking if false, checking for valid json format.

# Conclusion

Oh no.

```
Run    Assignment05-StarterOG
Student Antoneh Caboolski is registered for Math.
Student Antoneh Caboolski is registered for Math.
Student Antoneh Caboolski is registered for Math.
Student Antoneh Caboolski is registered for Math.
Student Antoneh Caboolski is registered for Math.
Student Antoneh Caboolski is registered for Math.
Student Antoneh Caboolski is registered for Math.
```

This is my achilles. Why does it duplicate the print statement? I had this issue last time. This is a new script taken from the module zip. Its really funny. Its really upsetting. Anyways error handling is not too

bad.