Tom Lowder

11/11/2024

IT FDN 110 A

Assignment05

Collections of data

Introduction

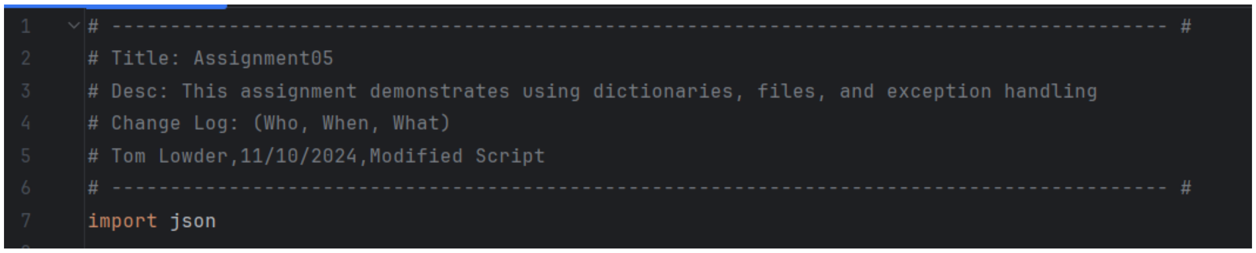
Assignment05 was based off information learned in module 05. Module 05 explored the concept of managing data with dictionaries. The role of indexes and keys were explained as well as the process of reading and writing to files from a dictionary using JSON file format. Then, structured error handling was clarified using the recommended Try-Except structure. Finally, network and cloud file sharing are explained with a in depth analysis of GitHub.

Creating the program

The instructor provided Assignment05-Starter.py file. Assignment05-Starter.py represents real-world use of modifying another developer's code. The main objective of Assignment05 was to convert the CSV list collection formatting to a JSON dictionary collection format while also incorpoarting error handling. The remainder of the paper describes modfications made to the program to achieve the task.

The header change log was modified because the program was modifiied. This included changing the change log's who, what, when to "Tom Lowder", "11/11/2024" , "Modified program".

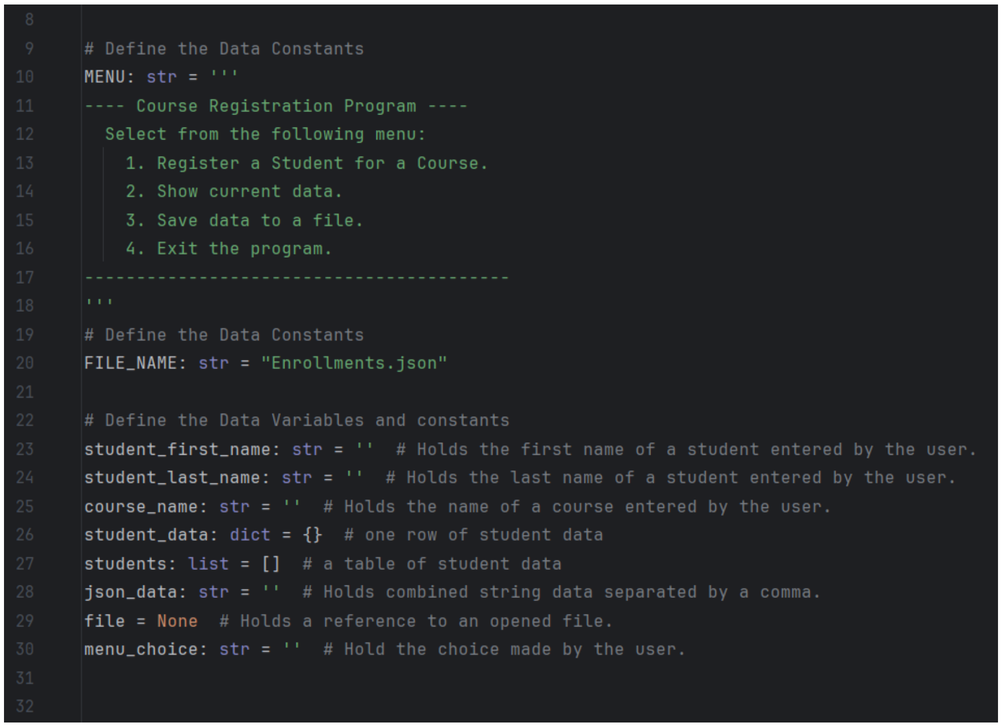
Since we are using JSON data, import json was used to import Python's json module. This was included right after the header as shown below



```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
# Tom Lowder,11/10/2024,Modified Script
# ------------------------------------------------------------------------------------------ #
import json
```

Figure 1. Header and import json

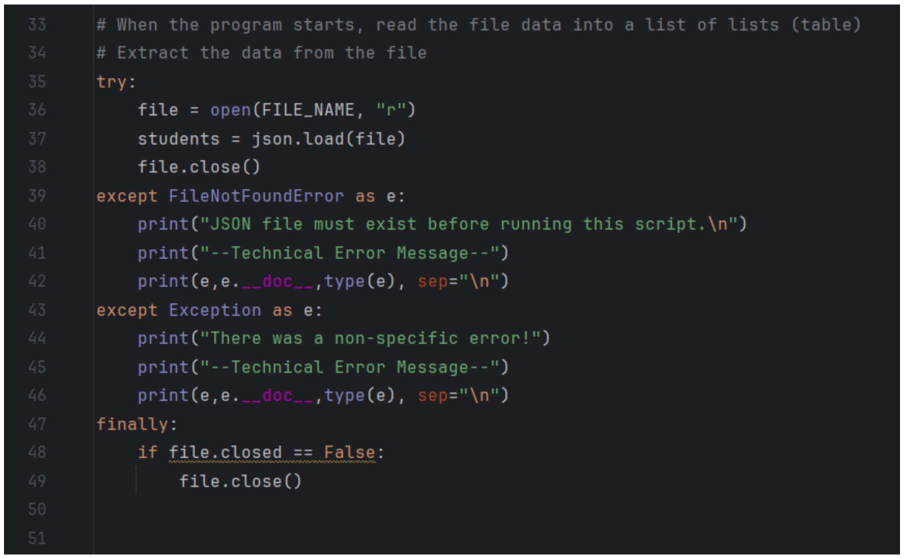
Most of the constants and variables were defined in the starter file. Constant modifications included changing FILE_NAME value Enrollments.csv to Enrollments.json to manage data in JSON format. Variable modifications included changing *csv_data* to *json_data,* changing *student_data* to dict type and setting it to empty dictionary {}. *Student_data* represented one row of student data in dictionary format. While *students* represented a two-dimensional list table of *student_data* or list of dictionaries.



```python

# 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"

# 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.


```

Figure 2. Defined constants and variables

Try-except was used in conjunction with opening, reading "Enrollments.JSON" extracting the data and loading the data to variable *students*. The first except catches the FileNotFoundError error if it occurs. It prints a more user-friendly error message "JSON file must exist before running this script". It then prints additional information about the error. This information includes the error message, documentation string, and type of exception (FileNotFoundError). The second exception is a catch-all for any other exceptions that may occur. It provides a general error message, "There was a non-specific error!". It then displays the error message, documentation string, and the type of the exception. The finally block checks to see if the file is not closed and if it is not closed, it closes the file.

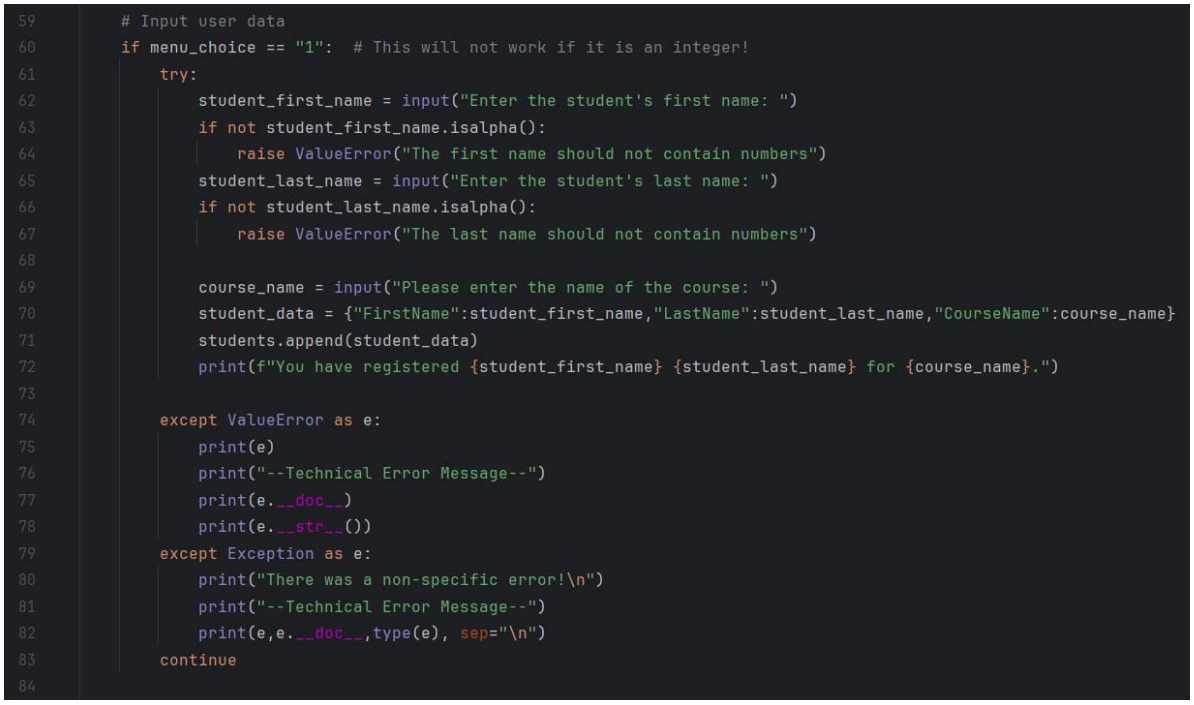


```python
# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)
    file.close()
except FileNotFoundError as e:
    print("JSON file must exist before running this script.\n")
    print("--Technical Error Message--")
    print(e,e.__doc__,type(e), sep="\n")
except Exception as e:
    print("There was a non-specific error!")
    print("--Technical Error Message--")
    print(e,e.__doc__,type(e), sep="\n")
finally:
    if file.closed == False:
        file.close()


```

Figure 3. Try-except and open json file for read

Try-except was included in the if menu_choice == "1" section of the program. The custom exception was used after *student_first_name* and *student_last_name* input to catch a specific ValueError exception that is raised if the input is not alphabetic. The first except ValueError as e: then prints both the custom error message (e) that was raised with the ValueError, "--Technical error Message—", the docstring (e.__doc__), and the string representation (e.__str__()) of the exception. The second except Exception as e: block is a catch-all for handling any other exceptions (i.e., exceptions that are not ValueError).

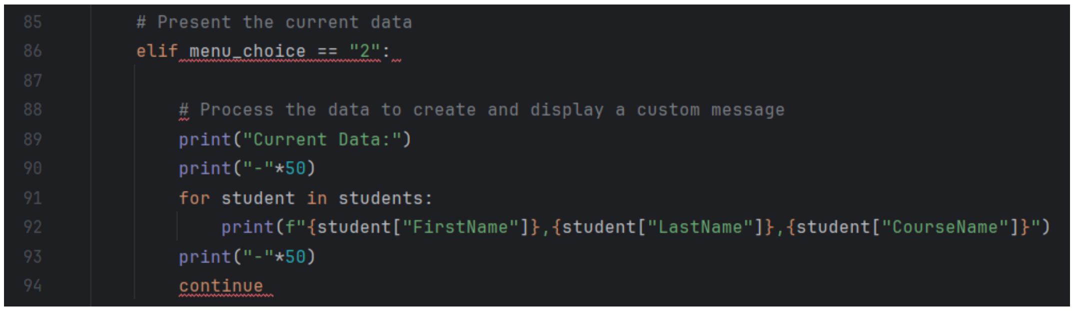
Student_data and *students* were modified in this section of the program as well. *Student_data* was set to dictionary formatting, while students.append(student_data) was used to add *student_data* dictionaries to the *students* list.



```
    # 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_first_name} {student_last_name} for {course_name}.")

        except ValueError as e:
            print(e)
            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 4. Try-except and student_data and students modification

For if menu_choice == "2" section of the program, a for loop was used to examine the dictionary keys in the two-dimensional list table *students.* The output was the dictionary values in comma separated string format.



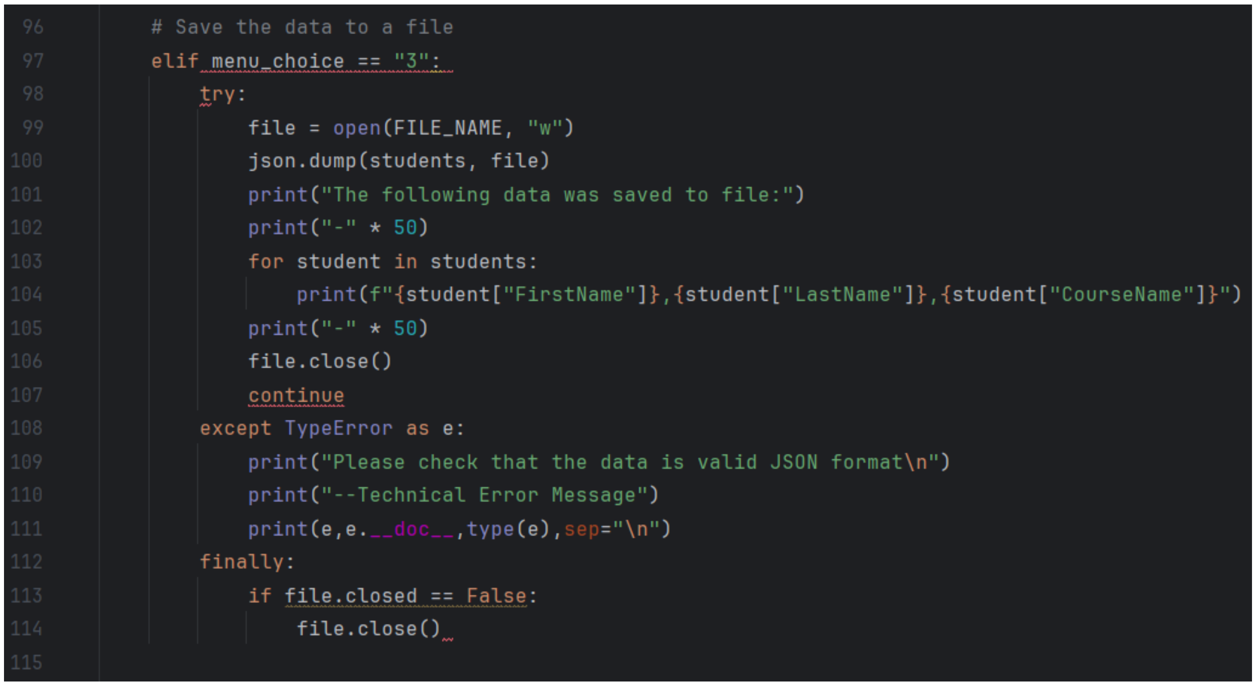
```
    # Present the current data
    elif menu_choice == "2":

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

Figure 5. For loop with dictionary keys

Try-except was included in the if menu_choice == "3" section of the program. The first except catches the TypeError error if it occurs. It prints a more user-friendly error message "Please check that the data is valid JSON format". It then prints additional information about the error. This information includes the error message, documentation string, and type of exception (TypeError). The finally block checks to see if the file is not closed and if it is not closed, it closes the file.

Json.dump(students, file) was used to "dump" the *students* data to the file.

Additionally, a for loop was used to examine the dictionary keys in the two-dimensional list table *students* to output the dictionary values in comma separated string format.



```
    # Save the data to a file
    elif menu_choice == "3":
        try:
            file = open(FILE_NAME, "w")
            json.dump(students, file)
            print("The following data was saved to file:")
            print("-" * 50)
            for student in students:
                print(f"{student["FirstName"]},{student["LastName"]},{student["CourseName"]}")
            print("-" * 50)
            file.close()
            continue
        except TypeError as e:
            print("Please check that the data is valid JSON format\n")
            print("--Technical Error Message")
            print(e,e.__doc__,type(e),sep="\n")
        finally:
            if file.closed == False:
                file.close()
```

Figure 6. Try-except and open json file for write

Testing the program

The program was tested in PyCharm and the command prompt. Tests included:

- The program takes the user's input for a student's first, last name, and course name.
- The program displays the user's input for a student's first, last name, and course name.
- The program saves the user's input for a student's first, last name, and course name to a coma-separated string file.
- The program allows users to enter multiple registrations (first name, last name, course name).
- The program allows users to display multiple registrations (first name, last name, course name).
- The program allows users to save multiple registrations to a file (first name, last name, course name).

The program successfully ran and saved to Enrollments.json in both PyCharm and the command prompt.

Lastly, I had to create a GitHub account and open a repository to save my program and this assignment on so others can review my work. This task helped mebetter understand how GitHub works.

Summary

With the resources provided in module 05 I was able to create the program. The program demonstrates my new understanding of modifying existing files for the use and management of data in JSON format. I also learned the use of exceptions in a program to catch errors. Finally, I learned the fundamentals of GitHub and how to create a repository and save files to it.