Sid Tolbert
11/11/2024
IT FDN 110 A
Assignment05

Assignment 5 – Advanced Collections and Error Handling

Introduction

This assignment focused on advanced Python concepts, including collections and error handling. The task was to modify a starter file to meet requirements, handle JSON data, and implement error handling to make the program more robust and user-friendly.

Doing the Assignment

From the starter file it looks like the input/output requirements are already met. I tested this by commenting out the file code outside of the While Loop and adding some print statements inside of the menu 1 If Statement. I could see that the requirements were met.

```
What would you like to do: 1
Enter the student's first name: Sue
Enter the student's last name: Salias
Please enter the name of the course: Python 100
You have registered Sue Salias for Python 100.
['Sue', 'Salias', 'Python 100']
[['Vic', 'Vu', 'Python 100'], ['Sue', 'Salias', 'Python 100']]
```

Figure 1: Input Output Verification

I reset all the lines to what they originally were in the starter file. Then I started modifying them to fulfill the requirements.

```
# Importing JSON
import json

# 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: Import JSON and declare variables.

I then made a For Loop to iterate through the content of the JSON file and append the data to the students list. The data is formatted into a dictionary and each dictionary is a place in the list. I put in some error handling in case the JSON file wasn't existing:

```
# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
# Error handling to
try:
    file = open(FILE_NAME, "r")
    for row in file:
        data: list = row.split(",")
        row = {"First Name": data[0], "Last Name": data[1], "Course Name": data[2].strip()}
        students.append(row)
    file.close()
except FileNotFoundError as e:
    print("JSON File with content must exist before running this script.")
```

Figure 3: Placing the JSON file data into the students table.

I then modified the student data list to be a dictionary in the first menu option:

```
student_data = {"First Name": student_first_name, "Last Name": student_last_name,"Course Name": course_name}
```

I replaced all the instances of the variable 'csv_data' with 'json_data'. Then I changed all the list indices to be the right dictionary keys:

```
json_data = f"{student['First Name']},{student['Last Name']},{student['Course Name']}\n"
```

For the first menu choice I did exception handling so the user couldn't put numbers in the student's first or last names:

```
# Input user data
# Error handling if the user tries to use numbers in the first name.
if menu_choice == "1":  # This will not work if it is an integer!
    try:
        student_first_name = input("Enter the student's first name: ")

        # Student name can't have numbers
        if student_first_name.isnumeric():
            raise ValueError("Student Name can't have numbers in it.")

        student_last_name = input("Enter the student's last name: ")

        # Student name can't have numbers
        if student_last_name.isnumeric():
            raise ValueError("Student Name can't have numbers in it.")

        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}.")
        continue
    except ValueError as e:
        print("There can't be numbers in a student's name.")
```

Figure 4: Menu choice 1 error handling.

I added in error handling for Menu 3 option. It should be a catch-all error message.

```
# Save the data to a file
# Error handling for all error types
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w")
        for student in students:
            json_data = f"{student['First Name']},{student['Last Name']},{student['Course Name']}\n"
            file.write(json_data)
        file.close()
        print("The following data was saved to file!")
        for student in students:
            print(f"Student {student['First Name']} {student['Last Name']} is enrolled in {student['Course Name']}")
        continue
    except Exception as e:
        print("Erroneous Behavior!")
        print("Built-In Python error info: ")
        print(e, e.__doc__, type(e), sep='\n')
```

Figure 5: Menu 3 with error handling.

Conclusion

Through this assignment, I enhanced the program by integrating JSON handling, organizing data into dictionaries, and adding error handling. These improvements ensured the program's functionality and resilience, reinforcing the importance of writing reliable and efficient code.