Camea Hoffman

February 26, 2025

Introduction to Programming with Python

Assignment 05

Assignment 05 – Advanced Collections and Error Handling

Introduction

This paper discusses the structure and functionality of my Course Registration Program, which I designed to allow users to register students for courses, view current registrations, and save data using JSON files. The program incorporates exception handling to manage potential errors effectively. By using dictionaries and lists, the script ensures data organization and storage efficiency. Below is a detailed breakdown of my approach and implementation. This was a very challenging assignment!

Code Structure and Implementation

Importing Necessary Modules

The program begins by importing the json module, which is essential for handling data storage and retrieval in a structured format. This allows student registration data to be saved and loaded from a JSON file.

```
import json

FILE_NAME = "Enrollments.json"
```

Defining Constants and Variables

To ensure clarity and maintainability, I defined constants and variables:

- FILE_NAME stores the name of the JSON file where data is saved.
- MENU holds the user interface menu for easy reference.
- Variables such as student_first_name, student_last_name, and course_name are initialized as empty strings to store user input.
- students is an empty list that will hold student registration data as dictionaries.

(see next page for diagram)

```
# 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
# 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.
menu_choice: str  # Hold the choice made by the user.
student_data: dict = {}  # one row of student data (TODO: Change this to a Dictionary)
students: list = []  # a table of student data
file = None
```

Loading Data from a File

At the start of the program, I attempt to open and load data from Enrollments.json. If the file does not exist, an error message is displayed. Exception handling ensures the program does not crash due to missing or corrupt files.

```
# 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("Text 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!\n")
    print("-- Technical Error Message --")
    print(e, e.__doc__, type(e), sep='\n')
finally:
    if file.closed == False:
        file.close()
```

User Interaction Loop

A while True loop presents a menu with four options:

```
# Present and Process the data
while (True):

    # Present the menu of choices
    print(MENU)
    menu_choice = input("What would you like to do: ")
    print()
```

1. **Register a Student**
    - The program prompts the user to enter a student's first and last name, ensuring they contain only alphabetic characters.
    - The course name is then entered, and the data is stored in a dictionary before being appended to the students list.
    - Exception handling is used to catch and display errors if invalid input is detected.

```
# 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) #prints 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("-- Technical Error Message --")
        print(e, e.__doc__, type(e), sep='\n')
    continue
```

2. **Show Current Data**
    - The program iterates through the students list and prints each student's registration details.

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

    # Process the data to create and display a custom message
    print("-"*50)
    for student_data in students:
        message = "{} {} is registered for {}."
        print(message.format( *args: student_data["FirstName"], student_data["LastName"], student_data["CourseName"]))
    print("-"*50)
    continue
```

3. **Save Data to a File**
    - The students list is written to Enrollments.json using json.dump().
    - The last student registered is displayed to confirm successful saving.
    - Exception handling ensures the file-writing process runs smoothly.

```
# Save the data to a file
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w")
        json.dump(students, file)
        file.close()
        print("The following data was saved to file!")
        if students:
            last_student = students[-1]
            message = "{} {} is registered for {}."
            print(message.format( *args: student_data["FirstName"], student_data["LastName"], student_data["CourseName"]))
        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')
    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()
```

4. **Exit the Program**
    - Selecting this option breaks the loop and ends the program.

```
    # 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")
```

Error Handling and Validation

I included multiple exception handling mechanisms to prevent unexpected crashes. I used try-except blocks to catch specific errors like missing files (FileNotFoundError) and invalid data (ValueError). Additionally, I included a safeguard to close files properly in a finally block.

Uploading to GitHub

I will be uploading this document and the script file to my GitHub repository:

https://github.com/CameaHoffman/Python110-Q2-2025

Conclusion

This Course Registration Program demonstrates my understanding of dictionaries, lists, file handling, and exception handling in Python. By structuring data and implementing error handling, I created a functional and user-friendly application (I hope!). This assignment enhanced my ability to work with persistent data storage while ensuring smooth user interactions. I struggled with this assignment a lot, and I am hopeful that the learning curve will show it's benefits in the next Modules.