Camea Hoffman

February 26, 2025

Introduction to Programming with Python

Assignment 05

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

Assignment 05 – Advanced Collections and Error Handling

Introduction This program is a Course Registration System that allows users to register students for courses, display current student registrations, and save the data to a file. It uses dictionaries to store student information, JSON for data persistence, and exception handling to manage errors.

Breakdown of the Code

1. **Header and Imports:**
    - The script begins with a comment block describing the assignment and logging changes.
    - The json module is imported to handle file operations.

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   CameaHoffman,02/26/25,Created Script
# ------------------------------------------------------------------------------------------ #

import json

FILE_NAME = "Enrollments.json"
```

2. **Constants and Variables:**
    - FILE_NAME is defined to store the filename where student data is saved.
    - MENU holds the menu text for user interaction.
    - Variables such as student_first_name, student_last_name, and course_name store user inputs.
    - student_data is a dictionary representing a single student, while students is a list containing multiple students.
    - Example student data is preloaded into student_table for initial demonstration. (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 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
students: list = []  # a table of student data
file = None

student_row1: dict = {"FirstName": "Vic", "LastName": "Vu", "CourseName": "Python"}
student_row2: dict = {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Java"}
student_table: list = [student_row1, student_row2]
```

3. **File Handling (Reading Data at Startup):**

    - The program attempts to open and read the JSON file into the students list.
    - If the file does not exist, a FileNotFoundError is handled, prompting the user that the file must exist.
    - A general exception handler is also included for unexpected errors.

```
# 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()
```

4. **User Interaction Loop:**

    - The program continuously displays a menu and waits for user input.

- The menu_choice variable stores the user's selection.

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

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

5. **Option 1: Registering a Student:**

- Prompts the user to enter a first name, last name, and course name.
- Input validation ensures names contain only alphabetic characters.
- The student's data is stored as a dictionary and appended to the students list.
- If invalid input is detected, the program raises and handles a ValueError.

```
# 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
```

6. **Option 2: Display Current Student Data:**

- Iterates over students and prints each student's details in a formatted output.

```
# 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:
            print(f"{student_data["FirstName"]} {student_data["LastName"]} is enrolled in " + \
                  f"{student_data["CourseName"]}")
    print("-"*50)
    continue
```

7. **Option 3: Save Data to File:**

    - Opens the file in write mode and saves the students list in JSON format.
    - Displays a confirmation message showing the last student added.
    - Includes exception handling for TypeError (invalid data format) and other unexpected errors.

```
# 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]
            print(f"{student_data["FirstName"]} {student_data["LastName"]} is registered for {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()
```

8. **Option 4: Exit Program:**

    - The loop terminates when the user selects this option.

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

9. **Error Handling:**
    - Various try-except blocks handle possible input and file-related errors to ensure smooth execution.

Summary

This program efficiently manages student course registrations using dictionaries and lists, emphasizing data validation and error handling. JSON ensures persistent data storage, while structured exception handling enhances robustness. The menu-driven interface provides a seamless user experience.

This is my second version of the program. After revisiting my code and watching Mr. Root's Q&A video for Module 6, I realized I hadn't fully grasped dictionary usage. I reworked my approach in this assignment to improve dictionary implementation.