Jeisson Guerrero Estrada

02/14/24

Python 100

Assignment 05

https://github.com/Jxeoninov8/IntroToProg-Python-Mod05

Collections & Error Handling

Introduction

I created a program to demonstrate my proficiency in using dictionaries, handling files, and implementing exception handling. The script reads data from a JSON file into the 'students' list, validates user input, and adds it to the list. It allows users to display the current data and save it to the file. I designed a basic course registration system with error handling and file operations, demonstrating my programming skills.

Library Import

To enable handling of JSON data, the 'json' library is imported. This library provides essential functionality for interacting with JSON-formatted information, which is a key element in the course registration program (Figure 1).

```
# Library import json
import json
```

FIGURE 1

Data Constants

Using defined constants like 'MENU' and 'FILE_NAME' improves the script's readability and flexibility. 'MENU' is responsible for the user interface and 'FILE_NAME' stores enrollment data in JSON format (Figure 2).

```
# 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'  # Change to Json file
```

FIGURE 2

Data Variables & Constants

This section defines variables such as 'student_first_name', 'student_last_name', and 'course_name' to store user inputs. 'students' is introduced as a list to store student data while 'FILE_NAME' represents the filename (Figure 3).

```
# 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 3

Data Reading & Initialization

When the script is started, it tries to read data from the designated file ('Enrollments.json'). The 'try' block handles the file reading process, while the 'except' block is responsible for scenarios where the file is missing. To ensure the file is properly closed, whether it is read or created, the 'finally' block is executed (refer to Figure 4).

```
# 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('The file does not exist.\n')
    print('---Technical Information---')
    print(e, e.__doc__, type(e), sep='\n')
    file = open(FILE_NAME, 'w')
    json.dump(students, file)
finally:
    if not file.close:
        file.close()
```

FIGURE 4

Menu Loop

The program runs in a continuous loop, displaying a menu for the user to select an action related to the course registration system (Figure 5).

```
# Present and Process the data
```

```
while True:

    # Present the menu of choices

    print(MENU)
    menu_choice = input('What would you like to do: ‘)
```

FIGURE 5

Menu Choices

The menu provides well thought-out options. Option 1 enables student registration while ensuring input validation and handling of potential errors. Option 2 displays the current data and includes error handling for potential key errors. Option 3 saves the data to a file and displays the saved information, while also managing potential exceptions. Finally, option 4 gracefully terminates the program when selected (refer to Figure 6).

```
# Input user data

if menu_choice == '1':
    try:
        student_first_name = input('Enter the student\'s first name: ')
        if not student_first_name.isalpha():
            raise ValueError('Student first name can only contain
alphanumeric characters')
        student_last_name = input('Enter the student\'s last name: ')
        if not student_last_name.isalpha():
```

```
            raise ValueError('Student last name can only contain alphanumeric
characters')

        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}  # Change list to a
dictionary

        students.append(student_data)

        print('-' * 50)

        print(f'You have registered {student_first_name} {student_last_name}
for {course_name}.')

        print('-' * 50)

    except ValueError as e:

        print('Unknown Error\n')

        print('---Technical Information---')

        print(e, e.__doc__, type(e), sep='\n')

    continue


# Present the current data


elif menu_choice == '2':


    # Process the data to create and display a custom message

    print('-' * 50)

    try:

        for student in students:

            print(f'{student["first_name"]},{student["last_name"]},
{student["course_name"]}\n')

    except KeyError as e:

        print('Invalid data found\n')

        print('---Technical Information---')

        print(e, e.__doc__, type(e), sep='\n')

        print('-' * 50)

        continue
```

```
# Save the data to a file

elif menu_choice == '3':
    try:
        print('-' * 50)
        file = open(FILE_NAME, 'w')
        json.dump(students, file)
        file.close()
        print('Data has been saved to file!')
        for student in students:
            print(f'{student["first_name"]},{student["last_name"]},
{student["course_name"]}\n')
        print('-' * 50)
    except Exception as e:
        print('Unknown Error')
        print('---Technical Information---')
        print(e, e.__doc__, type(e), sep='\n')
    continue

# Stop the loop

elif menu_choice == '4':
    break
else:
    print('Option invalid please  choose from options 1, 2, or 3')
```

FIGURE 6

Exception Handling

Exception handling is a crucial part of the script. The 'try', 'except', and 'finally' blocks proficiently manage file-related exceptions, including situations where the file cannot be found. Furthermore, the script uses error handling to validate input during student registration, handle unexpected key errors during data display, and deal with unforeseen errors during data processing.

Summary

This is a Python course registration program that utilizes dictionaries, file operations, and meticulous exception handling to create a user-friendly and error-tolerant experience. It features constants for streamlined menu presentation and file management, allowing users to register students, display data, save information, and exit the program. With thoughtful implementation and error management, this program is a commendable example of a reliable Python script for course registration.