Name: Celia Louie
Date: November 13, 2023
Course: IT FDN 110 B
GitHubURL: https://github.com/celialouie/IntroToProg-Python

Assignment 05 - Advanced Collections and Error Handling

Introduction

For this assignment, I created a Python program that builds on what we've learned so far and adds on the use of data processing using dictionaries, exception handling, and json files for capturing students' course registration.

Creating a program

I started out by using the starter file provided and filling out the script header. This assignment builds on Assignment 4 so I kept the data constants and variables the same and added student_data and set it as an empty dictionary. I also imported json as we will be using the json module in this assignment.

```python
# ------------------------------------------------------------------------------------------ #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   Celia Louie, 11/13/2023, Created Script
# ------------------------------------------------------------------------------------------ #

# Define the Data Constants
import json

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.
----------------------------------------
'''
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables and constants
student_first_name: str = ''
student_last_name: str = ''
course_name: str = ''
csv_data: str = ''
file = None
menu_choice: str
student_data: dict = {}
students: list = []
```

Figure 1: Added script header and defined constants and variables.

I wanted the program to read the json file data in the beginning so that the data could be displayed later on. I used the json module with the open() function with “r” to open and read the file and the load() method. I included some structured error handling (Try-Except) to manage and customize error statements in a more organized way for the user. This eror handling accounts for if the file is not found and a generic exception with some print statements to inform the user. The variable “e” is intended to reference the automatically generated Exception object and the “e.__doc___” code will output a documentation string and type of exception. I then closed the file.

```python
# When the program starts, read the file data
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)
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 not file.closed:
        file.close()
```

Figure 2: Read in json file with eror handling

The menu of options in this assignment are similar to what we did in Assignment 3 and 4 so I will highlight the new additions. Menu choice 1 prompts the user to input the student’s first name, student’s last name, and the course course name. I included isalpha() to check that the input for the student’s first and last name doesn’t contain any numbers. I added the user’s input as a dictionary to the empty dictionary student_data we defined earlier in the script.

```python
    # Input user data
    if menu_choice == "1":
        try:
            # Check that first and last name input only contain letters
            student_first_name = input("Enter the student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("The first name should only contain letters")
            student_last_name = input("Enter the student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError("The last name should only contain letters")
        except Exception as e:
            print("There was a non-specific error\n")
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep="\n")
        course_name = input("Enter the name of the course: ")
        student_data = {"student_first_name": student_first_name,
                        "student_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

```

Figure 3: Prompt user for input with error handling

For menu choice 2, to present the data I used the print function and a for loop to print out the data with a custom message.

```python
    # Present the data
    elif menu_choice == "2":
        # Process the data to create and display a custom message
        print("-"*50)
        for student in students:
            print(f'Student {student["student_first_name"]} {student["student_last_name"]} is enrolled in {student["course_name"]}')
        print("-"*50)
        continue

```

Figure 4: Present the data

On menu choice 3, I wanted to save the inputs from menu choice 1 to the json file named “Enrollments.json”. I saved the data using the json module and included some error handling for if the file doesn’t yet exist or there’s a non-specific error to let the user know. Then I closed the file.

```python
    # Save the data to a file
    elif menu_choice == "3":
        try:
            file = open(FILE_NAME, "w")
            json.dump(students, file)
            print("Registration has been saved")
        except FileNotFoundError as e:
            print("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 not file.closed:
                file.close()
                continue

```

Figure 5: Save the data to json file

Testing

I tested the program to make sure it ran correctly by taking the user's input, displaying the inputs, saving the input as a json file, showing error messaging as expected, and allowing for multiple registrations. I ran the program in both PyCharm and terminal.

```
main.py   Enrollments.json
[{"student_first_name": "Celia", "student_last_name": "Louie", "course_name": "Python 100"}, {"student_first_name": "Vic", "student_last_name": "Vu", "course_name": "Python 100"}]
```

Figure 7: Confirming the program saves the input to a json file

Summary

I created a program using lists and files to display a menu option for the user to input a student's registration for a Python course with dictionaries, error handling, and json files. I tested the program to make sure it runs correctly in PyCharm.