Name: Kelela Kai' lani Boreta

Date: 14 Aug 2024

Course: Foundations of Programming: Python

Assignment: Module 7 – Classes and Objects

https://github.com/kboreta/IntroToProg-Python-Mod07

Developing a Python Course Registration Program with Data Classes and Error Handling

Introduction

The task was to create a Python program to manage course registrations for students, focusing on demonstrating the use of constants, variables, and data classes. The program needed to handle input/output operations, perform error handling, and store data in a structured format using JSON. This paper explains the step-by-step process of writing the code for this assignment, including defining data classes, implementing file handling, and ensuring robust user interaction through error handling.

Step-by-Step Code Explanation

1. Script Header and Imports

The script begins with a header that outlines the purpose of the program and tracks changes. The `json` module is imported to handle reading from and writing to a JSON file.

```
# ------------------------------------------------------------------------------------------------------- #
# Title: Assignment07
# Desc: This assignment demonstrates using constants, variables, and #data classes # with
# structured error handling for a course registration program.
# Change Log: (Who, When, What)
#  KBoreta,8/14/2024,Created Script
# ------------------------------------------------------------------------------------------------------- #

import json
```

2. Define Constants

Two constants are defined: `MENU` for displaying the program's options to the user and `FILE_NAME` for the name of the JSON file that stores student data. Constants are immutable and help maintain the program's readability and maintainability.

```
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"
```

3. Define Data Classes

Data classes represent the structure of data being handled. In this program, `Person` is a base class with `first_name` and `last_name` attributes, while `Student` extends `Person` to include `course_name`.

```
class Person:
    def __init__(self, first_name: str = "", last_name: str = ""):
        self.first_name = first_name
        self.last_name = last_name

class Student(Person):
    def __init__(self, first_name: str = "", last_name: str = "", course_name: str = ""):
        super().__init__(first_name, last_name)
        self.course_name = course_name
```

Each class includes properties with simple validation to ensure data integrity, such as checking that names contain only letters.

4. FileProcessor Class

This class contains methods to read from and write to the JSON file. It uses structured error handling to manage potential issues like file not found or JSON decoding errors.

```
class FileProcessor:
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        try:
            with open(file_name, 'r') as file:
                data = json.load(file)
                for item in data:
                    student_data.append(Student(item['first_name'], item['last_name'], item['course_name']))
        except FileNotFoundError as e:
            IO.output_error_messages("File not found.", e)
        except json.JSONDecodeError as e:
            IO.output_error_messages("Error decoding JSON from file.", e)
        except Exception as e:
            IO.output_error_messages("An unexpected error occurred while reading the file.", e)

    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
        try:
            with open(file_name, 'w') as file:
                json.dump([s.to_dict() for s in student_data], file)
            print("Data saved successfully.")
        except Exception as e:
```

```
            IO.output_error_messages("An error occurred while writing to the file.", e)
```

5. IO Class

The `IO` class handles all input/output operations, such as displaying menus and capturing user input. It also includes methods for displaying error messages.

```
class IO:
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        print(f"Error: {message}")
        if error:
            print(f"Exception: {error}")

    @staticmethod
    def output_menu(menu: str):
        print(menu)

    @staticmethod
    def input_menu_choice():
        return input("Please select a menu option: ").strip()

    @staticmethod
    def output_student_courses(student_data: list):
        print("Current Student Registrations:")
        for student in student_data:
            print(f"{student.student_first_name} {student.student_last_name} is registered for
{student.course_name}.")

    @staticmethod
    def input_student_data(student_data: list):
        try:
            first_name = input("Enter the student's first name: ").strip()
            last_name = input("Enter the student's last name: ").strip()
            course_name = input("Enter the course name: ").strip()
            student_data.append(Student(first_name, last_name, course_name))
        except Exception as e:
            IO.output_error_messages("An error occurred while entering student data.", e)
```

6. Main Program Loop

The main body of the script initializes the `students` list and enters a loop to display the menu and handle user input. It uses the `IO` class for interactions and `FileProcessor` for file operations.

```
if __name__ == "__main__":
    students = []
    FileProcessor.read_data_from_file(FILE_NAME, students)
    menu_choice = ""

    while menu_choice != "4":
```

```
IO.output_menu(MENU)
menu_choice = IO.input_menu_choice()

if menu_choice == "1":
    IO.input_student_data(students)
elif menu_choice == "2":
    IO.output_student_courses(students)
elif menu_choice == "3":
    FileProcessor.write_data_to_file(FILE_NAME, students)
elif menu_choice == "4":
    print("Exiting program.")
else:
    print("Invalid option. Please select again.")
```

Summary

This assignment provided a comprehensive exercise in building a Python program that integrates data classes, error handling, and file I/O operations. By structuring the code with classes like `FileProcessor` and `IO`, the program achieves a clean separation of concerns, making it easier to maintain and extend. The use of constants and validation ensures data integrity, while structured error handling provides resilience against runtime errors. This program is an effective demonstration of managing user interactions and data persistence in Python.