Erik Duong

November 24, 2024

Course: Python IT FDN 110

Assignment: 06

https://github.com/erikduong807/IntroToProg-Python-Mod06

Create Course Registration Program

Introduction

This document outlines the steps taken to create a Python program that demonstrates using constants, variables, and print statements to display a message about a student's registration for a Python course. This program will **add the use of functions, classes, and using the separation of concerns pattern.** The program incorporates the **IO** and **FileProcessor** classes to maintain clean separation between user interactions and file operations.

Topic

1. Understanding the Requirements

To begin, I thoroughly reviewed the assignment prompt to identify specific requirements, including the use of constants, variables, user input, string formatting, and file handling. The program needed a menu-driven structure, allowing for student registration, data display, data saving to a JSON file, and program exit.

2. Defining Constants

I defined two constants, MENU and FILE_NAME, to ensure consistency:

- MENU: Holds the program menu options.
- FILE_NAME: Holds the file name "Enrollments.json", which I need to update from csv file to json file to avoid further issue in the code.

3. Declaring Variables

I defined two variables, Students and menu_choice. These minimal variables ensure the program remains focused and memory-efficient.

```
# Define the Data Variables and constants
students: list = []  # a table of student data
menu_choice: str  # Hold the choice made by the user.
```

```
# Save the data to a file
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w")
        for student in students:
            json_data = f"{student['first_name']},{student['last_name']},{student['course_name']}\n"
            file.write(json_data)
        file.close()
    except Exception as e:
        print('Error saving data to file')
        print(e)
    finally:
        if file and not file.closed:
            file.close()
    print("The following data was saved to file!")
    for student in students:
        print(f"Student {student['first_name']} {student['last_name']} is enrolled in {student['course_name']}")
```

4. IO & FileProcess Class

This program demonstrates how to effectively manage a Python application using constants, lists, and classes. By utilizing the **IO** and **FileProcessor** classes, the program maintains a clear separation between user interactions and data processing.

```python
# Processing ------------------------------------- #
class FileProcessor:
    """
     A collection of processing layer functions that work with json files

     ChangeLog: (Who, When, What)
    """

    def read_data_from_file(file_name:str, student_data:list):
        """This function reads data from a json file and loads it into a list of dictionary rows

        ChangeLog: (Who, When, What)
        :return: list
        """
        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            file.close()
        except Exception as e:
            IO.output_error_messages(message="Error: There was a problem with reading the file.", error=e)

        finally:
            if file.closed == False:
                file.close()
        return student_data

    def write_data_to_file(file_name: str, student_data: list):
        """This function writes data from a json file and loads it into a list of dictionary rows

               ChangeLog: (Who, When, What)
               """
        try:
            file = open(file_name, "w")
            json.dump(student_data, file)
            file.close()
            IO.output_student_and_course_name(student_data=student_data)
        except Exception as e:
            message = "Error: There was a problem with writing to the file. "
            message += "Please check that the file is not open by another program."
            IO.output_error_messages(message=message,error=e)
        finally:
            if file.closed == False:
                file.close()
```

```python
# Presentation ------------------------------------- #
class IO:
    """
    A collection of presentation layer functions that manage user input and output

    ChangeLog: (Who, When, What)
    """

    @staticmethod
    def output_error_messages(message: str, error = None):
        """ This function displays the a custom error messages to the user

        ChangeLog: (Who, When, What)
        RRoot,1.3.2030,Created function

        :return: None
```

By separating user interface logic (handled by the **IO** class) from file operations (handled by the **FileProcessor** class), the code becomes **modular** and much easier to understand.