Camea Hoffman

March 6, 2025

Introduction to Programming with Python

Assignment 06

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

Assignment 06 – Functions

Introduction

This program is designed to manage student course registrations using Python functions, structured error handling, and JSON file operations. It allows users to register students, view registrations, and save data to a file. The program follows a modular approach, separating concerns into processing and presentation layers.

Code Explanation

1. **Program Structure**
    - The script starts by defining constants, including a menu for user interaction and the JSON file name for storing data.
    - The students list holds the student registration data.

```
import json

# define 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 file
  4. Exit the program
  """

FILE_NAME: str = "Enrollments.json"

# define program variables

menu_choice: str = ""
students: list = []
```

2. **Processing Layer (FileProcessor Class)**

```
class FileProcessor:  2 usages
    """ A collection of processing layer functions that work with Json files

    ChangeLog (Who, When, What)
    CamegaHoffman, 03.05.25,Created Class"""
```

- read_data_from_file(): Reads student data from a JSON file and handles errors if the file is missing or corrupted.

```
@staticmethod  1 usage
def read_data_from_file(file_name:str, student_data: list):

    """Extract and read data from Json file

        ChangeLog (Who, When, What)
        CamegaHoffman, 06.03.25, Created Function

        :return: student data"""

    try:
        file = open(file_name, "r")
        student_data = json.load(file)
        file.close()
    except FileNotFoundError as e:
        IO.output_error_messages( message: "Text file must exist before running the script!", e)
    except Exception as e:
        IO.output_error_messages( message: "There was a non-specific error!", e)
    finally:
        if file.closed == False:
            file.close()
    return student_data
```

- write_data_to_file(): Saves student registration data to a JSON file, displaying a confirmation message with the last saved entry. It includes error handling to ensure data integrity.

```
@staticmethod  1 usage
def write_data_to_file(file_name: str, student_data: list):
    """Write user input data to Json file

        ChangeLog (who, when, what)
        CamegaHoffman, 03.06.25, Created fuction

        :return: None """

    try:
        file = open(file_name, "w")
        json.dump(student_data, file)
        file.close()
        print("Data saved to file!")
        if students:
            last_student = students[-1]
            print(f"The last entry saved was {last_student['FirstName']} {last_student['LastName']} is registered
    except TypeError as e:
        IO.output_error_messages( message: "Please check that the data is in valid JSON format", e)
    except Exception as e:
        IO.output_error_messages( message: "There was a non-specific error!", e)
    finally:
        if file.closed == False:
            file.close()
```

3. **Presentation Layer (IO Class)**

```
class IO:  11 usages
    """A collection of presentation layer functions that manage user input and output

    ChangeLog (Who, When, What)
    CamegHoffman, 03.05.25, Created Class
    CamegHoffman, 03.05.2025,Added menu output and input functions
    CamegHoffman, 03.05.2025,Added a function to display the data
    CamegHoffman, 03.05.2025,Added a function to display custom error messages"""
```

- output_error_messages(): Displays error messages, differentiating between general and technical errors.

```
@staticmethod  7 usages
def output_error_messages(message: str, error: Exception = None):
    """This function displays the custom error messages to the user
        ChangeLog: (Who, When, What)
        CamegHoffman,03.05.2025,Created function)

        :return: None
        """
    print(message, end="\n\n")
    if error is not None:
        print("-- Technical Error Message --")
        print(error, error.__doc__, type(error), sep='\n')
```

- output_menu(): Prints the main menu for user interaction.

```
@staticmethod  1 usage
def output_menu(menu: str):
    """This function displays the menu to the user
    ChangeLog: (Who, When, What)
    CamegHoffman, 03.05.25, Created function)

    :return: None"""

    print()
    print(menu)
    print() #adding extra space to make it look nicer
```

- input_menu_choice(): Gets user input for menu selection, validating it to prevent invalid choices.

```python
@staticmethod  1 usage
def input_menu_choice():
    """This function gets a menu choice from the user
    ChangeLog: (Who, When, What)
    CameaHoffman, 03.05.25, Created function

    :return: string with the users choice"""

    choice = "0"
    try:
        choice = input("Enter your menu choice number: ")
        if choice not in ("1","2","3","4"): # Note these are strings
            raise Exception ("Please choose only 1, 2, 3, or 4")
    except Exception as e:
        IO.output_error_messages(e.__str__())
        #not passing the exception object to avoid the technical message

    return choice
```

- input_student_data(): Collects student details, ensuring names contain only letters. The data is stored in a dictionary and appended to the student list.

```python
@staticmethod  1 usage
def input_student_data(student_data: list):
    """This function gets information from user

        ChangeLog: (Who, When, What)
        CameaHoffman, 03.06.25, Created Function

        :return: string with user's input data
        """

    try: #input student data

        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 = {"FirstName": student_first_name,
                   "LastName": student_last_name,
                   "CourseName": course_name}
        student_data.append(student)
        print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")

    except ValueError as e:
        IO.output_error_messages( message: "-- Technical Error Message -- ", e)
    except Exception as e:
        IO.output_error_messages( message: "Error: There was a problem with your entered data.", e)
        return student_data
```

- output_student_courses(): Displays all registered students and their courses in a formatted list.

```
@staticmethod  1 usage
def output_student_courses(student_data: list):
    """ This function displays the student registration list

        ChangeLog: (Who, When, What)
        CameaHoffman, 03.05.25, Created functions

        :return: displays current full registration list of students and courses """
    print()
    print("-" * 50)
    for student in student_data:
        message = "{} {} is registered for {}."
        print(message.format( *args: student["FirstName"], student["LastName"], student["CourseName"]))
    print("-" * 50)
    print()
```

-

4. **Main Program Execution**

   - The program loads existing student data from the JSON file.

```
# Beginning of the main body of this script
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)
```

   - It continuously displays the menu and processes user choices, allowing registration, data viewing, and file saving.

```
while True:

    IO.output_menu(menu=MENU)

    menu_choice = IO.input_menu_choice()

    if menu_choice == "1": #input user data
        IO.input_student_data(student_data=students)
        continue

    elif menu_choice == "2": #display current data
        IO.output_student_courses(student_data=students)
        continue

    elif menu_choice == "3": #save data
        FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
        continue
```

   - The program exits when the user selects option 4.

```
elif menu_choice == "4": # exit program
    print("The program will now exit.")
    break
```

Summary

This script demonstrates function-based programming with structured error handling and file operations in Python. By organizing logic into processing and presentation layers, the program ensures clarity, maintainability, and user-friendly interactions.