ARSENE HYACINTHE DINA NGOLLO
08/07/2024
IT FDN 110 A
Assignment 06
https://github.com/Andybowell/IntroToProg-PythonMod06

Python Program for Student Registration: Learning Functions, Classes, and the Separation of Concerns Programming Pattern

Introduction

This week's assignment focuses on creating a Python script using the PyCharm IDE to manage student registration data. We will demonstrate the use of constants, variables, print statements, and string formatting to display registration messages. Building on Assignment 5, this task will reinforce our understanding of basic Python syntax and concepts through practical application.

Additionally, we will learn three techniques to enhance our scripts: functions, classes, and the separation of concerns programming pattern. The assignment aims to improve our skills with elements such as while loops, programming menus, conditional logic, and data processing using dictionaries. Furthermore, we will ensure data integrity through proper error handling and validation, managing exceptions like file not found, JSON decoding errors, and invalid user inputs. This holistic approach will deepen our technical skills and enhance our ability to write clean, maintainable code.

Creating the Program

In Assignment 6, we advance from the previous work by enhancing the code's organization and maintainability by applying functions, classes, and the separation of concerns programming patterns. This assignment builds on the knowledge from Assignment 5 and introduces more structured and advanced programming techniques to improve script functionality and clarity. We started by defining two primary classes: FileProcessor and IO. Each class has descriptive docstrings that outline their specific roles and functionalities. Additionally, all functions within these classes include detailed docstrings to ensure clarity and ease of maintenance. The use of@staticmethod decorators simplifies method access, and exception handling is managed through dedicated error message functions.
Key functions in the program include:

1. output_error_messages(message: str, error: Exception = None): Displays custom error messages.
2. output_menu(menu: str): Presents the menu of choices to the user.
3. input_menu_choice(): Retrieves the user's menu selection.
4. output_student_courses(student_data: list): Outputs the list of student courses.
5. input_student_data(student_data: list): Collects student details from the user.
6. read_data_from_file(file_name: str, student_data: list): Reads data from a JSON file.
7. write_data_to_file(file_name: str, student_data: list): Writes data to a JSON file.

The program's implementation involved setting essential constants for consistency. The MENU constant was defined to include formatted menu options, while FILE_NAME specified the JSON file ("Enrollments.json") for storing student data.
Variables were initialized to handle user input and data management. student_first_name, student_last_name, and course_name were initialized as empty strings to store user inputs. The json_data string was set to empty for holding combined string data, and the file was initially set to None to reference the opened file. A menu_choice string was prepared for user selections, and student_data was created as an empty dictionary. The students' list was established to maintain student records.
The program starts by reading the "Enrollments.json" file into the students' list, handling file operations in read mode, and parsing data into a list of dictionaries. Error handling is incorporated to manage file reading issues such as file not found or JSON decoding errors.
A while loop manages user interaction, presenting a menu with four main options:
1. Register a Student for a Course: Prompts the user for student details, formats them into a dictionary, and appends them to the students' list. Error handling ensures valid input.
2. Show Current Data: Displays the list of student registrations by iterating through the students list.
3. Save Data to a File: Opens the "Enrollments.json" file in write mode and saves the students list using JSON.dump(). Error handling checks for valid JSON format.
4. Exit the Program: Prints a termination message and exits the loop to end the program.

Testing confirmed the program's functionality across different environments, ensuring it accurately displays and saves student data. The program was validated both in PyCharm and from the console or terminal. Please refer to the figures below for illustration.

Assignment06.py Enrollments.json

```json
[
 {
  "FirstName": "Bob",
  "LastName": "Smith",
  "CourseName": "Physic 101"
 },
 {
  "FirstName": "Sue",
  "LastName": "Jones",
  "CourseName": "C++ 100"
 }
]
```

Figure 1-Enrollments.json

Assignment06.py Enrollments.json

```python
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06_Starter
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   RRoot,1/1/2030,Created Script
#   Arsene Hyacinthe Dina Ngollo,08/06/2024,<Activity>
# ------------------------------------------------------------------------------------------ #
import json


# 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.csv"
```

IO › input_student_data() › try

Figure 2-Importing and Menu

```python
# Define the Data Constants
# FILE_NAME: str = "Enrollments.csv"
FILE_NAME: str = "Enrollments.json"

# 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
csv_data: str = ''  # Holds combined string data separated by a comma.
json_data: str = ''  # Holds combined string data in a json format.
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.


2 usages
class FileProcessor:  # function that works with json file
    """
    A collection of processing layer functions that work with Json files

    ChangeLog: (Who, When, What)
    RRoot,1.1.2030,Created Class
```

Figure 3- defining
Constants, variables and creation of FileProcessor class

```python

    1 usage
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        """
        Reading data from Json files

        ChangeLog: (Who, When, What)
        RRoot,1.1.2030,Created Class
        """
        try:
            file = open(file_name, "r")
            student_data = json.load(file)

            file.close()
        except FileNotFoundError as e:
            IO.output_error_messages("Text file must exist before running this script!", e)
        except Exception as e:
            IO.output_error_messages("There was a non-specific error!", e)
        finally:
            if not file.closed:
                file.close()
        return student_data
```

Figure 4. creating the read_data_from_file function inside the FileProcessor class

Assignment06.py × {} Enrollments.json

```python
    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
        """
                Writing data to Json files

                ChangeLog: (Who, When, What)
                RRoot,1.1.2030,Created Class
                """

        try:
            file = open(file_name, "w", )
            json.dump(student_data, file, indent=1)

            file.close()
            print("The following data was saved to file!")
            for student in student_data:
                print(f'Student {student["FirstName"]} '
                      f'{student["LastName"]} '
                      f'is enrolled in {student["CourseName"]}')
        except TypeError as e:
            IO.output_error_messages("Please check that the data is a valid JSON format", e)
        except Exception as e:
            IO.output_error_messages("There was a non-specific error!", e)
        finally:
            if not file.closed:
                file.close()
        return student_data

```

Figure 5. creating the write_data_to_file function inside the FileProcessor class

Assignment06.py × {} Enrollments.json

```python

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

        ChangeLog: (Who, When, What)
        RRoot,1.1.2030,Created Class
        RRoot,1.2.2030,Added menu output and input functions
        RRoot,1.3.2030,Added a function to display the data
        RRoot,1.4.2030,Added a function to display custom error messages
        """

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

                ChangeLog: (Who, When, What)
                RRoot,1.3.2030,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')

```

Figure 6. Creation of IO class and creating the output_error_messages function

```python
    1 usage
    @staticmethod
    def output_menu(menu: str):
        """ This function displays the menu of choices to the user

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

            :return: None
            """

        # Present the menu of choices
        print("="*100)  # make the format look nicer
        print(menu)
        print("="*100)  # make the format look nicer
```

Figure 7. Creating the output_menu function

```python
    1 usage
    @staticmethod
    def input_menu_choice():
        """ This function gets a menu choice from the user

            :return: string with the users choice
            """
        choice = "0"
        try:
            choice = input("What would you like to do: ")
            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 e to avoid the technical messag
        return choice
```

Figure 8. Creating the input_menu_choice function

```python
    1 usage
    @staticmethod
    def output_student_courses(student_data: list):
        """ This function displays the student course to the user

            ChangeLog: (Who, When, What)
            RRoot,1.3.2030,Created function
            RRoot,1.4.2030,Added code to toggle technical message off if no exception object is passed

            :return: None
            """
        # Process the data to create and display a custom message
        print("-" * 100)
        for student in student_data:
            print(f'Student {student["FirstName"]} '
                  f'{student["LastName"]} '
                  f'is enrolled in {student["CourseName"]}')
        print("-" * 100)
```

Figure 9. Creating the output_student_courses function

Assignment06.py × {} Enrollments.json

```python
    1 usage
    @staticmethod
    def input_student_data(student_data: list):
        """ This function gets the first name, last name, and course from the user

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

            :return: str
            """
        try:
            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("That value is not the correct type of data!", e)
        except Exception as e:
            IO.output_error_messages("There was a non-specific error!", e)
        return student_data
```

Figure 10. Creating the input_student_data function

Assignment06.py × {} Enrollments.json

```python
# reading data from file
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

# Present and Process the data
while True:

    # Present the menu of choices
    IO.output_menu(menu=MENU)

    # make a choice out of the menu
    menu_choice = IO.input_menu_choice()

    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        IO.input_student_data(student_data=students)
        continue

    # Present the current data
    elif menu_choice == "2":
        # Process the data to create and display a custom message
        IO.output_student_courses(student_data=students)
        continue

    # Save the data to a file
    elif menu_choice == "3":
        FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
        continue

```

Figure 11. Re-organizing the functions

```python
    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop
    else:
        print("Please only choose option 1, 2, or 3")

print("Program Ended")
```

Figure 12. Re-organizing the functions

```
---- 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.
----------------------------------------


==========================================================================================
What would you like to do: 2
------------------------------------------------------------------------------------------
Student Bob Smith is enrolled in Physic 101
Student Sue Jones is enrolled in C++ 100
------------------------------------------------------------------------------------------
==========================================================================================
```

Figure 13. Testing on PyCharm IDE

```
---- 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.
----------------------------------------


==========================================================================================
What would you like to do: 1
Enter the student's first name: Ming
Enter the student's last name: Vu
Please enter the name of the course: Art 101
You have registered Ming Vu for Art 101.
==========================================================================================
```

Figure 14. Testing on PyCharm IDE (suite)

```
---- 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.
----------------------------------------


==========================================================================================
What would you like to do: 3
The following data was saved to file!
Student Bob Smith is enrolled in Physic 101
Student Sue Jones is enrolled in C++ 100
Student Ming Vu is enrolled in Art 101
==========================================================================================
```

Figure 15. Testing on PyCharm IDE (suite)

```
---- 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.
----------------------------------------


==========================================================================================
What would you like to do: 4
Program Ended

Process finished with exit code 0
```

Figure 16. Testing on PyCharm IDE (suite)

```
Assignment06 — python3 Assignment06.py — 109×28
Last login: Tue Jul 30 19:14:48 on ttys000
(base) arsenengollo@arsenes-MacBook-Air ~ % cd Desktop/python_work/Summer_python_Class/Module/_Module06/Assignment/Assignment06
(base) arsenengollo@arsenes-MacBook-Air Assignment06 % python3 Assignment06.py
==========================================================================================

---- 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.
----------------------------------------

==========================================================================================
What would you like to do: 2
------------------------------------------------------------------------------------------
Student Bob Smith is enrolled in Physic 101
Student Sue Jones is enrolled in C++ 100
Student Ming Vu is enrolled in Art 101
------------------------------------------------------------------------------------------
==========================================================================================
```

Figure 17. Testing on terminals

```
Assignment06 — python3 Assignment06.py — 109×28
==========================================================================================

---- 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.
----------------------------------------

==========================================================================================
What would you like to do: 1
Enter the student's first name: Mario
Enter the student's last name: Bros
Please enter the name of the course: Gym 101
You have registered Mario Bros for Gym 101.
==========================================================================================
```

Figure 18. Testing on terminals (suite)

```
Assignment06 — python3 Assignment06.py — 109×28

---- 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.
-----------------------------------------

=========================================================================================
What would you like to do: 3
The following data was saved to file!
Student Bob Smith is enrolled in Physic 101
Student Sue Jones is enrolled in C++ 100
Student Ming Vu is enrolled in Art 101
Student Mario Bros is enrolled in Gym 101
=========================================================================================
```

Figure 19. Testing on terminals (suite)

```
---- 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.
-----------------------------------------

=========================================================================================
What would you like to do: 4
Program Ended
(base) arsenengollo@arsenes-MacBook-Air Assignment06 %
```

Figure 20. Testing on terminals (suite)

```
Enrollments.json
[
  {
   "FirstName": "Bob",
   "LastName": "Smith",
   "CourseName": "Physic 101"
  },
  {
   "FirstName": "Sue",
   "LastName": "Jones",
   "CourseName": "C++ 100"
  },
  {
   "FirstName": "Ming",
   "LastName": "Vu",
   "CourseName": "Art 101"
  },
  {
   "FirstName": "Mario",
   "LastName": "Bros",
   "CourseName": "Gym 101"
  }
]
```

Figure 21. Enrollments.json after running the program in the terminal

Summary

In this Python assignment, we built on the foundational work from previous tasks by enhancing our script with functions, classes, and the separation of concerns programming patterns. This approach improved code organization and maintainability while deepening our understanding of Python syntax and programming concepts.
We began by structuring our code with two primary classes, FileProcessor and IO, each equipped with clear documentation. This structure, along with the use of static methods and well-defined functions, streamlined our script and enhanced its readability. Key functions such as output_error_messages, output_menu, input_menu_choice, output_student_courses, input_student_data, read_data_from_file, and write_data_to_file was implemented to handle various tasks and ensure robust error management.

This assignment solidified our grasp of Python programming by incorporating advanced techniques and error handling, preparing us for more complex programming challenges in future tasks.