Nick Greco
Nov 16, 2024
Foundations of Programming: Python
Assignment06
https://github.com/NBGreco/IntroToProg-Python-Mod06

Functions, Classes, and Structured Error Handling in Python

Introduction

This document presents an in-depth overview of a Python script developed for managing course registrations. The script features a menu-driven interface that allows users to register students, display current enrollments, and save data to a JSON file. The script builds on previous assignments by incorporating functions, classes, error handling strategies, and the separation of concerns, which all contribute to overall maintainability and robustness. Specific examples are presented to demonstrate its execution in both PyCharm IDE and Windows Command Prompt environments.

Creating the Script

Overview

The script implements a course registration system enabling users to register students, display current enrollments, and save the data to a JSON file through structured functions. It is organized into two main classes, “FileProcessor” for managing file operations and “IO” for handling user input and output, which promotes modularity and maintainability throughout the code. The ‘Assignment06’ script is shown below in Figure 1.

```
# ------------------------------------------------------------------------ #
# Title: Assignment06
# Desc: This assignment demonstrates using functions with structured error
#         handling.
# Change Log: (Who, When, What)
#   N.Greco, 11/15/2024, Created Script
#   N.Greco, 11/16/2024, Updated Output Formatting in Functions
# ------------------------------------------------------------------------ #
import json

```

```
# Define the data constants.
FILE_NAME: str = "Enrollments.json" # Removed old Enrollments.csv reference.
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 variables.
menu_choice: str        # Hold the choice made by the user.
students: list = []     # Holds a table of student data.

# Removed all of these variables and used them locally instead:
   # course_name: str = ''  # Holds the name of a course entered by the user.
   # csv_data: str = ''  # Holds combined string data separated by a comma.
   # file = None  # Holds a reference to an opened file.
   # json_data: str = ''  # Holds combined string data in a json format.
   # student_data: dict = {}  # Holds one row of student data.
   # 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.

# Processing ------------------------------------------------------------- #
class FileProcessor:
   """
   A collection of processing layer functions that work with JSON files.

   ChangeLog: (Who, When, What)
   N.Greco, 11/15/2024, Created Class
   """


   @staticmethod
   def read_data_from_file(file_name: str, student_data: list):
       """
       A function that extracts data from a JSON file.

       ChangeLog: (Who, When, What)
       N.Greco, 11/15/2024, Created Function
       N.Greco, 11/16/2024, Updated Output Formatting

       :return: A list with student data
       """
       try:
```

```
            file = open(file_name, "r")
            student_data = json.load(file)
            file.close()
        except FileNotFoundError as e:
            IO.output_error_messages("\n!!! JSON file must exist " \
                    "before running this script. !!!", e)
        except Exception as e:
            IO.output_error_messages("\nThere was a non-specific error!", e)
        finally:
            if not file.close():
                file.close()
        return student_data


    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
        """
        A function that writes data to a JSON file.

        ChangeLog: (Who, When, What)
        N.Greco, 11/15/2024, Created Function
        N.Greco, 11/16/2024, Updated Output Formatting

        :return: A list with student data
        """
        try:
            file = open(file_name, "w")
            json.dump(student_data, file)
            file.close()
            IO.output_student_courses(student_data)
        except TypeError as e:
            IO.output_error_messages("\nPlease check that data is valid JSON \
                formatting!", e)
        except Exception as e:
            IO.output_error_messages("\nThere was a non-specific error!", e)
        finally:
            if not file.close():
                file.close()


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

    ChangeLog: (Who, When, What)
    N.Greco, 11/15/2024, Created Class
    """
```

```python
    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        """
        A function that displays a custom error message to the user.

        ChangeLog: (Who, When, What)
        N.Greco, 11/15/2024, Created Function

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


    @staticmethod
    def output_menu(menu: str):
        """
        A function that displays the menu of choices to the user.

        ChangeLog: (Who, When, What)
        N.Greco, 11/15/2024, Created Function

        :return: None
        """
        print(menu)


    @staticmethod
    def input_menu_choice():
        """
        A function that requests the menu choice from the user.

        ChangeLog: (Who, When, What)
        N.Greco, 11/15/2024, Created Function
        N.Greco, 11/16/2024, Updated Output Formatting

        :return: None
        """
        choice = "0"
        try:
            choice = input("What would you like to do? ")
            if choice not in ("1", "2", "3", "4"):
                raise Exception("\n!!! Please choose a menu option " \
                                "(1, 2, 3, or 4). !!!")
        except Exception as e:
```

```
            IO.output_error_messages(e.__str__())
        return choice


    @staticmethod
    def output_student_courses(student_data: list):
        """
        A function that displays current student data.

        ChangeLog: (Who, When, What)
        N.Greco, 11/15/2024, Created Function
        N.Greco, 11/16/2024, Updated Output Formatting

        :return: None
        """
        # Process the data to create and display a custom message
        print("\n" + "-" * 60)
        for student in student_data:
            print(f"\t{student["FirstName"]} {student["LastName"]} " \
                  f"is enrolled in {student["CourseName"]}")
        print("-" * 60)


    @staticmethod
    def input_student_data(student_data: list):
        """
        A function that requests student data from the user.

        ChangeLog: (Who, When, What)
        N.Greco, 11/15/2024, Created Function
        N.Greco, 11/16/2024, Updated Output Formatting

        :return: None
        """
        try:
            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!")

            course_name = input("Please enter the name of the course: ")
            student_data = {"FirstName": student_first_name,
                            "LastName": student_last_name,
                            "CourseName": course_name}
            students.append(student_data)
```

```
                        print(f"\nYou  have  registered  {student_first_name}
{student_last_name}" \
                  f" for {course_name}.")
        except ValueError as e:
            IO.output_error_messages('\nThat value is not the correct type' \
                  'of data!', e)
        except Exception as e:
            IO.output_error_messages("\nThere was a non-specific error!", e)
        return students


# Main body of script. Starts by reading in JSON file data.
students = FileProcessor.read_data_from_file(file_name = FILE_NAME, \
      student_data = students)

# Repeat the following tasks.
while True:

    # Present the menu of choices and request user selection.
    IO.output_menu(menu = MENU)
    menu_choice = IO.input_menu_choice()

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

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

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

    # Stop the loop and exit the program.
    elif menu_choice == "4":
        print("\n" + "-" * 35)
        print("*** Exiting Program. Thank you! ***")
        print("-" * 35)
        break  # out of the loop
```

Figure 1: 'Assignment06' Python Script

Key Functionalities & Concepts Demonstrated

The script effectively utilizes JSON file read and write operations, functions, classes, error handling, and separation of concerns to achieve its objectives of managing student course registrations efficiently. A description of how each element contributes to the overall functionality follows.

JSON File Read

The “FileProcessor” class contains a static method “read_data_from_file”, which attempts to read student data from a JSON file (‘Enrollments.json’). It uses the json.load() function to parse the contents into a list of dictionaries, allowing easy access to the student data.

JSON File Write

Additionally, the “FileProcessor” class has another static method, “write_data_to_file”, which saves the current list of student data back to the JSON file using json.dump(). This ensures that any updates to the student list are persisted between program executions.

Functions

The script is structured around several well-defined functions that encapsulate specific tasks, such as “output_menu”, “input_student_data”, and “output_student_courses” in the “IO” class, and the file handling functions in the “FileProcessor” class. This modular design enhances readability and makes it easier to maintain and update the code, as changes to a specific functionality can be made without affecting the overall program flow.

Classes

The script is organized into two primary classes: “FileProcessor” for managing data operations (reading from and writing to the JSON file) and “IO” for handling user input and output. This object-oriented approach allows for encapsulation of related functionalities, promoting better organization and reusability of code.

Error Handling

The script employs structured error handling using try-except blocks to catch and manage exceptions that may arise during file operations and user input. For instance, it gracefully handles “FileNotFoundError” when attempting to read from a non-existent JSON file and “TypeError” when invalid data is written to the file. Custom error messages are provided through the “output_error_messages” method, informing users of specific issues while also displaying technical details for debugging, which enhances the robustness of the application.

Separation of Concerns

The design of the script exemplifies the separation of concerns by distinguishing between the data handling (via the “FileProcessor” class), user interface (via the “IO” class), and the main program logic (the loop that manages the menu and user interactions). This separation allows

each component to focus on its specific role, improving code clarity and making it easier to implement changes or troubleshoot issues. For example, if modifications are needed for how data is displayed or handled, those changes can be made in the “IO” class without affecting the file processing logic.

Running the Script

Using PyCharm IDE

The ‘Assignment06’ script was run using PyCharm IDE using the starter ‘Enrollments.json’ file. Figure 2 (left) demonstrates selecting menu option “1” and entering multiple new students. Example user input is shown in green text. Figure 2 (right) demonstrates the output when menu options “2” and “3” are chosen.

```
----- 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: Tony
Enter the student's last name: Hawk
Please enter the name of the course: Skateboarding 900

You have registered Tony Hawk for Skateboarding 900.

----- 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: Indiana
Enter the student's last name: Jones
Please enter the name of the course: Intro to Archaeology

You have registered Indiana Jones for Intro to Archaeology.
```

```
What would you like to do? 2

-------------------------------------------------------------
    Bob Smith is enrolled in Python 100
    Sue Jones is enrolled in Python 100
    Tony Hawk is enrolled in Skateboarding 900
    Indiana Jones is enrolled in Intro to Archaeology
-------------------------------------------------------------

----- 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

-------------------------------------------------------------
    Bob Smith is enrolled in Python 100
    Sue Jones is enrolled in Python 100
    Tony Hawk is enrolled in Skateboarding 900
    Indiana Jones is enrolled in Intro to Archaeology
-------------------------------------------------------------
```

Figure 2: ‘Assignment06’ Menu Selections “1” thru “3”.

Figure 3 shows the program output when menu option “4” is chosen and the program is exited.

```
----- 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

----------------------------------
*** Exiting Program. Thank you! ***
----------------------------------
```

Figure 3: Menu Option "4" Output

Figure 4 shows the contents of the 'Enrollments.json' file after running the 'Assignment06' script and providing user input to register two additional students.

Assignment06.py Enrollments.json

```
[
  {"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"},
  {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"},
  {"FirstName": "Tony", "LastName": "Hawk", "CourseName": "Skateboarding 900"},
  {"FirstName": "Indiana", "LastName": "Jones", "CourseName": "Intro to Archaeology"}
]
```

Figure 4: 'Enrollments.json' Data Contents after running Script

Figure 5 demonstrates error handling when incorrect menu option selections are provided. In the first scenario, a blank input was provided. In the second scenario, an alphabetic phrase was provided.

```
----- 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?

!!! Please choose a menu option (1, 2, 3, or 4). !!!

----- 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? Not an Option

!!! Please choose a menu option (1, 2, 3, or 4). !!!

----- 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
---------------------------------------
```

Figure 5: Menu Selection User Input Error Handling

Figure 6 demonstrates error handling when incorrect user input is provided for a new student data entry. In this example, a phone number was provided instead of a first name.

```
What would you like to do? 1
Enter the student's first name: 555-5555

That value is not the correct type of data!

----------- Technical Error Message -----------
The first name should only contain letters!
Inappropriate argument value (of correct type).
<class 'ValueError'>
```

Figure 6: Menu Option "1" User Input Error Handling

Figure 7 demonstrates error handling when the 'Enrollments.json' file is not found when initially read. To simulate this error, the file was temporarily renamed.

```
!!! JSON file must exist before running this script. !!!

---------- Technical Error Message ----------
[Errno 2] No such file or directory: 'Enrollments1.json'
File not found.
<class 'FileNotFoundError'>
Traceback (most recent call last):
  File "C:\Users\Greco\Desktop\Foundations of Programming\Code Repository\Module06\Assignment06.py", line 210, in <module>
    students = FileProcessor.read_data_from_file(file_name = FILE_NAME, student_data = students)
  File "C:\Users\Greco\Desktop\Foundations of Programming\Code Repository\Module06\Assignment06.py", line 67, in read_data_from_file
    if not file.close():
           ^^^^
UnboundLocalError: cannot access local variable 'file' where it is not associated with a value
```

Figure 7: 'Enrollments.json' Read Error Handling

Figure 8 demonstrates error handling when the 'Enrollments.json' file is unable to be written to. To simulate this error, the file was set to read-only mode.

```
What would you like to do? 3

There was a non-specific error!

---------- Technical Error Message ----------
[Errno 13] Permission denied: 'Enrollments.json'
Not enough permissions.
<class 'PermissionError'>
Traceback (most recent call last):
  File "C:\Users\Greco\Desktop\Foundations of Programming\Code Repository\Module06\Assignment06.py", line 231, in <module>
    FileProcessor.write_data_to_file(file_name = FILE_NAME, student_data = students)
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Greco\Desktop\Foundations of Programming\Code Repository\Module06\Assignment06.py", line 93, in write_data_to_file
    if not file.close():
           ^^^^
UnboundLocalError: cannot access local variable 'file' where it is not associated with a value
```

Figure 8: 'Enrollments.json' Write Error Handling

Using Windows Command Prompt

Figure 9 shows running the same script using Windows Command Prompt.

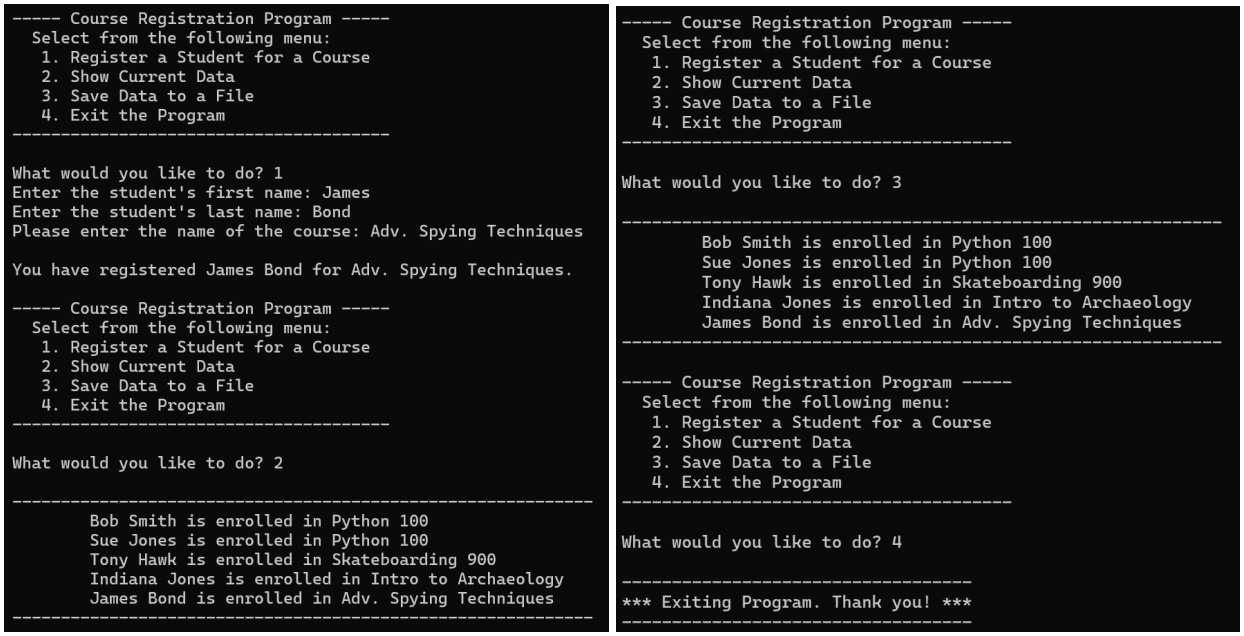

Figure 9: Using Windows Command Prompt to run 'Assignment06.py'

Figure 10 shows the contents of the JSON file after executing the script using Command Prompt.

Figure 10: JSON File Contents after running Script in Windows Command Prompt

Summary

This script demonstrates a basic course registration system using Python, featuring a menu-driven interface for users to register students, display current enrollments, and save data to a JSON file. It incorporates structured error handling to manage file operations and user input validation, ensuring data integrity and providing informative feedback for various error scenarios. By organizing the code into distinct classes for file processing and user interaction, the script showcases effective programming practices for modularity and maintainability.

Citations

OpenAI ChatGPT. (November 2024). https://chatgpt.com/: Aspects of this assignment were informed and created by queries I submitted to ChatGPT.