Pattama Srianomai

Aug 20, 2025

Foundations of Programming: Python

Assignment 06
https://github.com/patanomai/IntroToProg-Python-Mod06

Steps to complete Assignment 06:
Data Processing Using Classes, Functions, and the Separation of Concerns Pattern

Introduction

In this paper, I will describe the steps I took to create a Python Script using classes, functions, and the separation of concerns programming pattern. Guided by the notes and videos provided in Module 06. I will walk through each step, beginning with opening and reviewing the starter file Assignment06-Starter.py.

Step-by-Step Process

Step 1. Opening and reviewing the starter file Assignment06-Starter.py

Opening a file in PyCharm

1. Use the Project Pane on the left side
2. Double-click the Assignment 06-Starter.py
3. It will appear in the editor window

Renaming a file in PyCharm

1. Right-click the Assignment 06-Starter.py on the Project Pane
2. Select Rename
3. Rename the file to Assignment06.py
4. Click Refactor

Step 2. Updating a script header

Update the script header with my name, current date, and change history. *My script header is shown in Figure 1 below.*

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment06_Starter
# Desc: This assignment demonstrates using functions
# with structured error handling
# Change Log: (Who, When, What)
#   RRoot,1/1/2030,Created Script
#   PSrianomai,8/17/2025,Created Script
# ------------------------------------------------------------------------------------------ #
```

Figure 1: Script Header

Step 3. Defining Data Constants and Global Variables

Assignment 06 - Task 4 outlined the required constants and variables for this exercise:

Constants:

- The constant **MENU: str** is set to the value:

  ---- 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

- The constant **FILE_NAME: str** is set to the value "Enrollments.json"
- The constant values do not change throughout the program.

Variables:

- **menu_choice: str** is set to empty string.
- **students: list** : list is set to and empty list

Figure 2 presents the constants and global variables used in my script.

```
import io as _io #needed to try closing in the finally block
import json #import code from Python's JSON module into my script


# 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.json"

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

Figure 2: Constants and Global Variables definition

Step 4. Processing and presenting the data

This assignment builds on the structure introduced in Assignment 05, but emphasizes encapsulating logic within custom classes and functions. Task 4 outlined requirements of classes, functions, Input/Output, Processing and Error Handling.

Classes:

- The program includes a class named FileProcessor.
- The program includes a class named IO.
- All classes include descriptive document strings.

Functions:

- All functions include descriptive document strings.
- All functions with except blocks include calls to the function handling error messages.
- All functions use the @staticmethod decorator.
- The program includes functions with the following names and parameters:

    - output_error_messages(message: str, error: Exception = None)
    - output_menu(menu: str)
    - input_menu_choice()
    - output_student_courses(student_data: list)
    - input_student_data(student_data: list)
    - read_data_from_file(file_name: str, student_data: list):
    - write_data_to_file(file_name: str, student_data: list):

Processing

- When the program starts, the contents of the "Enrollments.json" are automatically read into the **students** two-dimensional list of dictionary rows using the json.load() function. (**Tip:** Make sure to put some starting data into the file or you will get an error!)
- On menu choice 3, the program opens a file named "Enrollments.json" in write mode using the open() function. It writes the contents of the **students** variable to the file using the json.dump() function. Next the file is closed using the close() method. Finally, the program displays what was written to the file using the **students** variable.
- On menu choice 4, the program ends.

Error Handling

- The program provides structured error handling when the file is read into the list of dictionary rows.
- The program provides structured error handling when the user enters a first name.
- The program provides structured error handling when the user enters a last name.
- The program provides structured error handling when the dictionary rows are written to the file.

To meet these requirements, I separated the script into processing and presentation layers.

Figure 3 - Processing: FileProcessor class, which includes two functions

- *read_data_from_file () uses json.load() function to load data into a list of dictionary*
- *write_data_to_file () writes the student data to Enrollments.json using json.dump()*

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

    ChangeLog: (Who, When, What)
    PSrianomai,8.17.2025,Created Class
    """
```

```python
@staticmethod  1 usage
def read_data_from_file(file_name: str, student_data: list):
    """ This function reads student data from a file
    and returns it as a list of dictionaries.

    ChangeLog: (Who, When, What)
    PSrianomai,8/17.2025,Created function
    :return: list of student dictionaries
    """
    file = _io.TextIOWrapper # add file as a local variable
    try:  # use error handling
        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"
                                  " this 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
```

```python
@staticmethod  1 usage
def write_data_to_file(file_name: str, student_data: list):
    """ This function writes student data to a file

    ChangeLog: (Who, When, What)
    PSrianomai,8/17.2025,Created function
    :return: None
    """
    file = _io.TextIOWrapper # add file as a local variable
    try:
        file = open(file_name, "w")
        json.dump(student_data, file, indent = 2) #write file using json dump
        file.close()
    except TypeError as e:
        IO.output_error_messages( message: "Please check that the data is a 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()
```

Figure 3: the FileProcessor class with read_data_from_file () and write_data_to_file ()

I followed the approach from Mod06-Lab03, using @staticmethod decorator for utility methods and adding docstring for clarity. I also implemented error handling block to handle potential errors.

Input / Output:

- On menu choice 1, the program prompts the user to enter the student's first name and last name, followed by the course name, using the input() function and stores the inputs in the respective variables.
- Data collected for menu choice 1 is added to the **students** two-dimensional list of dictionaries rows.
- On menu choice 2, the program uses the print() function to show a string of comma-separated values for each row collected in the **students** variable.

- On menu choice 4, the program ends.

Figure 4 shows my script for IO class in the presentation layer, which incorporates all input and output functions:

- *output_error_messages (message: str, error: Exception = None)*
- *output_menu (menu: str)*
- *input_menu_choice ()*
- *output_student_courses (student_data: list)*
- *input_student_data (student_data: list)*

```
# Presentation-----------------------------------#
class IO:  10 usages
    """
    A collection of presentation layer functions that manage user input
    and output

    ChangeLog: (Who, When, What)
    PSrianomai,8.17.2025,Created Class
    PSrianomai,8.17.2025,Added menu output and input functions
    """
```

```
    @staticmethod  6 usages
    def output_error_messages(message:str, error:Exception=None):
        """ This function displays the a custom error messages to the user

        ChangeLog: (Who, When, What)
        PSrianomai,8/17.2025,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')

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

        ChangeLog: (Who, When, What)
        PSrianomai,8.17.2025,Created function
        :return: None
        """
        print(menu)
```

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

        ChangeLog: (Who, When, What)
        PSrianomai,8.17.2025,Created function
        :return: string with the users choice
        """
        return input("Enter your menu choice number: ")

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

        ChangeLog: (Who, When, What)
        PSrianomai,8.17.2025,Created function
        :return: None
        """
        # Process the data to create and display a custom message
        print("-" * 50)
        for student in student_data:
            print(f'{student["FirstName"]},{student["LastName"]},'
                  f'{student["CourseName"]}')
        print("-" * 50)
```

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

    ChangeLog: (Who, When, What)
    PSrianomai,8.17.2025,Created function
    :return: student data (A dictionary containing 'FirstName',
    'LastName', and 'CourseName')
    """
```

```
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: ")
    #stores data input to student variable
    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: "That value is not the correct type of data!", e)
except Exception as e:
    IO.output_error_messages( message: "There was a non-specific error!", e)
return student_data
```

Figure 4: IO class with all input and output functions

Figure 5 presents the main body of the script, which incorporates a for loop statement with call functions.

```
# Beginning of the main body of this script
students = FileProcessor.read_data_from_file(file_name = FILE_NAME,
                                             student_data = students)
# Repeat the follow tasks
while True:
    IO.output_menu(menu = MENU)

    menu_choice = IO.input_menu_choice()

    # On menu choice 1, the program prompts the user to enter the student's
    # first name and last name, followed by the course name, using the input()
    # function and stores the inputs in the respective variables.
    # Data collected for menu choice 1 is added to the students 2D list of
    # dictionaries rows.
    # using try-except to check when users input first names and last name
    if menu_choice == "1": # call input_student_data function
        students = IO.input_student_data(student_data=students)
        continue
    # Present the current data:
    # On menu choice 2, the presents a string by formatting the collected
    # data using the print() function.
    elif menu_choice == "2": # call output_student_courses function
        IO.output_student_courses(student_data=students)
        continue
```

```
    # On menu choice 3, the program opens a file named "Enrollments.json"
    # in write mode using the open() function. It writes the contents of
    # the students variable to the file using the json.dump() function.
    # Next, the file is closed using the close() method. Finally, the
    # program displays what was written to the file using the students variable
    elif menu_choice == "3":  # Call write_data_to_file function
        FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
        # the program displays what was written to the file using the
        # students variable
        print("The following data was saved to file!")
        for student in students:
            print(f'{student["FirstName"]},{student["LastName"]},'
                  f'{student["CourseName"]}')
        continue

    # On menu choice 4, the program ends
    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop
    else:
        print("Please only choose option 1, 2, 3 or 4")

print("Program Ended")
```

Figure 5: Incorporate a for loop to loop and call functions.

Step 5. Testing the Script

I tested the script in PyCharm and the Command Prompt. *The results are shown in Figure 6.* After running the script, I verified that the JSON file was saved successfully on my computer *(see Figure 7).*

PyCharm:

```
C:\Python\Python3.13\python.exe C:\Users\taeya\Documents\Python\PythonCourse\A06\As

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

Enter your menu choice number: 2
------------------------------------------------
Bob,Smith,Python 100
Sue,Jones,Python 100
Pat,Sri,Python 100
Am,Serm,Python 100
Taeya,Serm,Python 100
Sam,Smith,Python 100
Tata,Young,Python 100
------------------------------------------------
```

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

Enter your menu choice number: 1
Enter the student's first name: Serena
Enter the student's last name: Kim
Please enter the name of the course: Python 100
You have registered Serena Kim for Python 100.

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

```
Enter your menu choice number: 3
The following data was saved to file!
Bob,Smith,Python 100
Sue,Jones,Python 100
Pat,Sri,Python 100
Am,Serm,Python 100
Taeya,Serm,Python 100
Sam,Smith,Python 100
Tata,Young,Python 100
Serena,Kim,Python 100
```

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

Enter your menu choice number: 4
Program Ended

Process finished with exit code 0
```

Command Prompt:

```
C:\Users\taeya\Documents\Python\PythonCourse\A06>python 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.
----------------------------------------

Enter your menu choice number: 1
Enter the student's first name: Mary
Enter the student's last name: Kim
Please enter the name of the course: Python 100
You have registered Mary Kim for Python 100.

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

Enter your menu choice number: 2
```

```
-------------------------------------------------
Bob,Smith,Python 100
Sue,Jones,Python 100
Pat,Sri,Python 100
Am,Serm,Python 100
Taeya,Serm,Python 100
Sam,Smith,Python 100
Tata,Young,Python 100
Serena,Kim,Python 100
Mary,Kim,Python 100
-------------------------------------------------

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

Enter your menu choice number: 3
The following data was saved to file!
Bob,Smith,Python 100
Sue,Jones,Python 100
Pat,Sri,Python 100
Am,Serm,Python 100
Taeya,Serm,Python 100
Sam,Smith,Python 100
Tata,Young,Python 100
Serena,Kim,Python 100
Mary,Kim,Python 100
```

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

Enter your menu choice number: 4
Program Ended

C:\Users\taeya\Documents\Python\PythonCourse\A06>
```

Figure 6. Script results

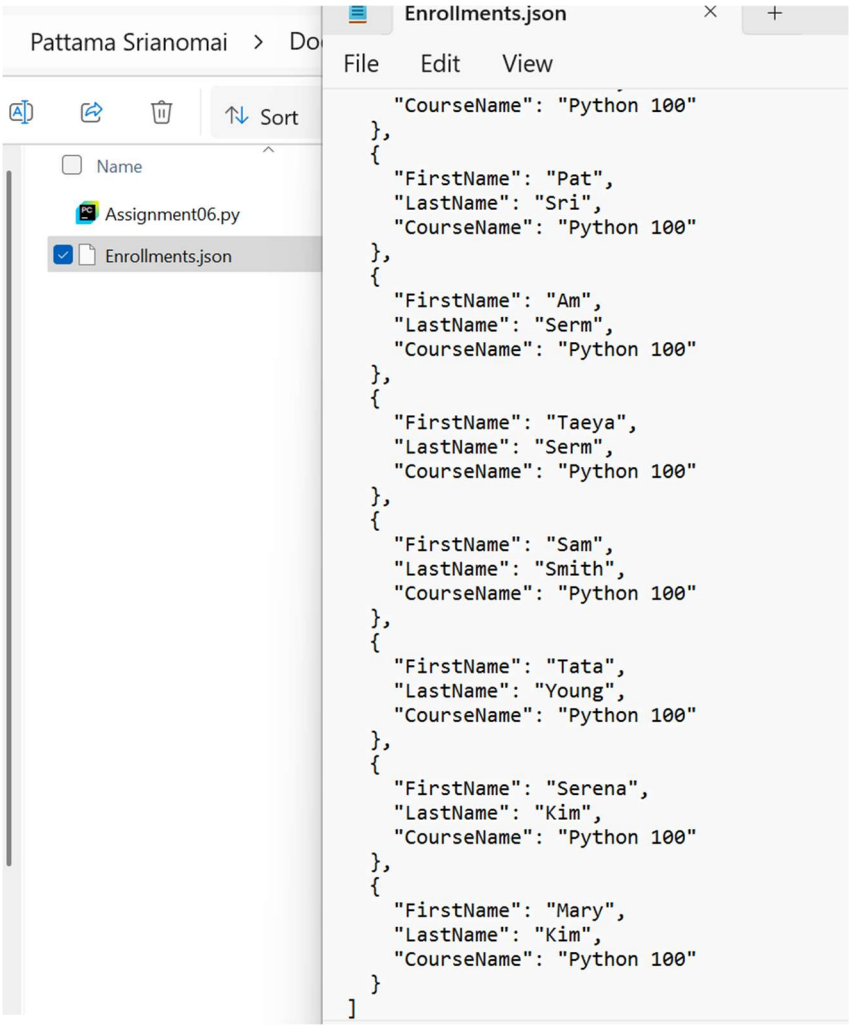

Figure 7. Saved JSON file

Summary

I developed a Python script using classes, functions, and the separation of concerns pattern to process and present student enrollment data. As the assignment progressed, I found the structure increasingly complex and leaned heavily on the lab materials for guidance. However, if I continue practicing, I believe my understanding will deepen over time.