Name: Student-PyQ
Date: 03/05/25
Course: IT FDN 110 A Wi 25: Foundations of Programming Python
Assignment: 06

GitHub URL:
https://github.com/Student-PyQ/IntroToProg-Python-Mod06

Organizing Code

Introduction

In this assignment, it is all about management of code over time. As code gets more complex and grows with more lines of code, maintaining it over time can get more complex. Using code organizing techniques like functions, classes, and separation of concerns, can help make the code easier to maintain and read. I approached completing this assignment by dividing it into 3 key areas:

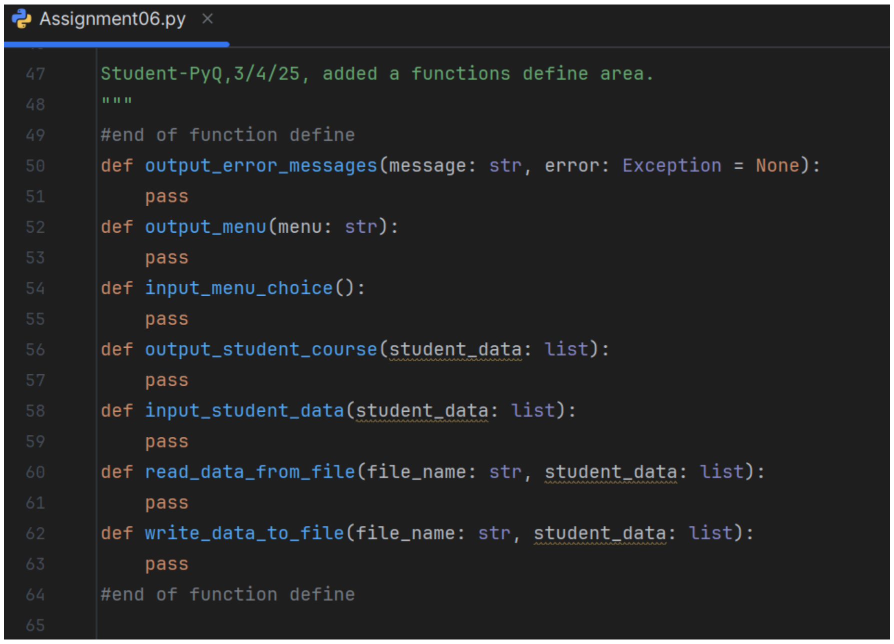
- Step 1 - Functions
- Step 2 - Classes
- Step 3 - Separations of Concerns

Step 1 - Functions

For the first step, I started with the example starter assignment code file. Then I updated the script head with an outline of changes to make in the code file. I realize the first thing I should tackle is creating functions. Functions are reusable blocks of code that provide modularity and reusability within a program. Since I knew that I had to create seven new functions, I started with a functions declaration section with empty functions as place holders using the pass syntax. And copied blocks of code associated with the function purpose. For example, defining the *output menu function* was the easiest one to associate the *print(menu)* statement with (**See Figures 1.1**). I continued on with the same approach with the other functions to define.

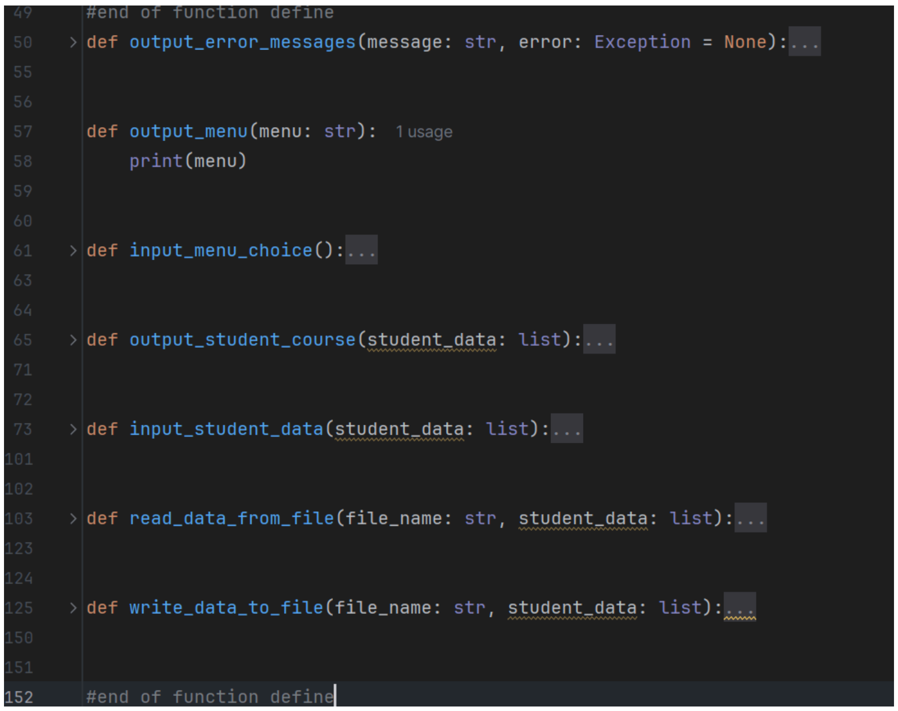
After defining each function, I also added the reference call to the new function within the main body of the program where the block of code was removed. For example, menu choice 1 section was replaced with a call to the *input student data* function with a student list as an argument (**See Figure 1.2**). For this key area of the assignment I learned about the differences between global and local variables. Local variables are contained inside the functions and are encapsulated/protected from other parts of the program outside of the function. Whereas, global variables defined within a function are still accessible from outside the function in other parts of the program. In this assignment, I found it helpful to explicitly define some function variables as global when the variable was not used as a function parameter.

Figures 1.1 - Empty Functions with Pass Syntax



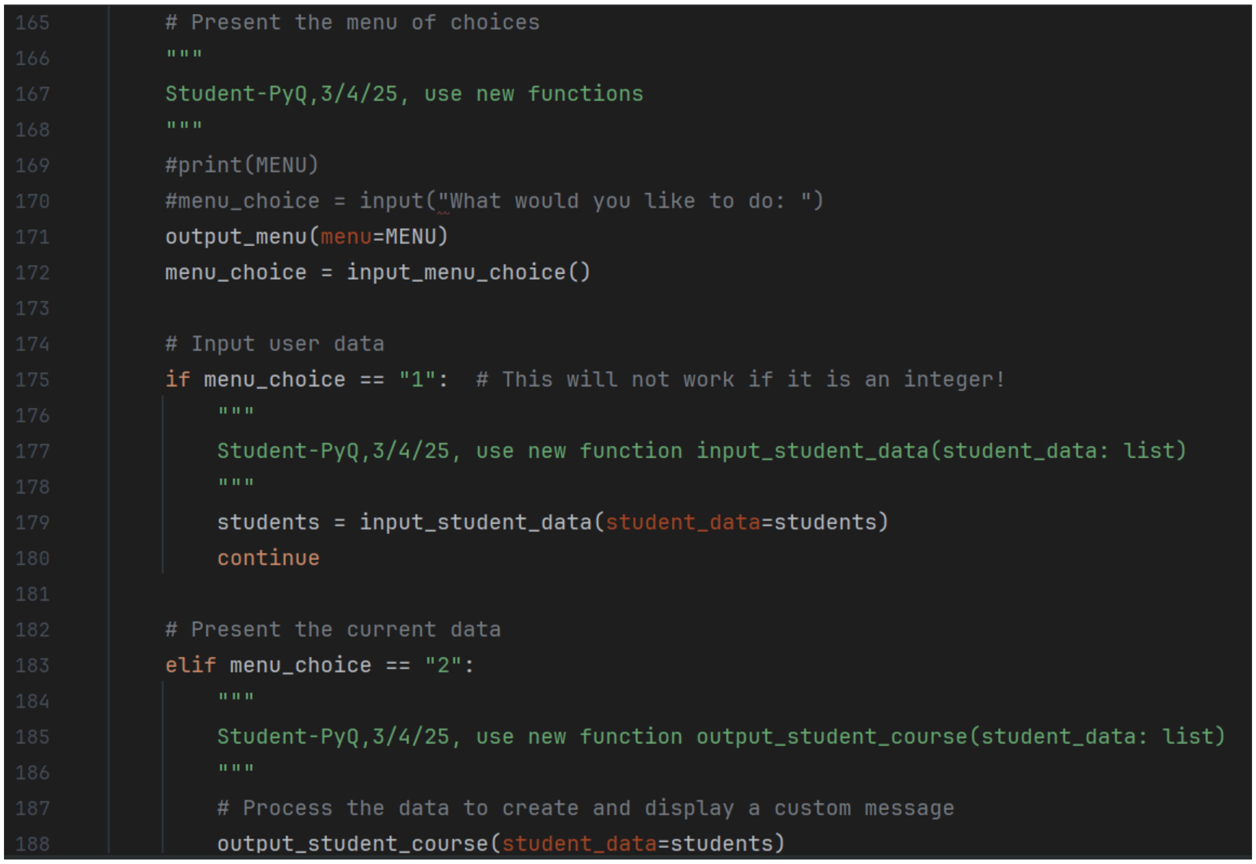
Assignment06.py

```
Student-PyQ,3/4/25, added a functions define area.
"""
#end of function define
def output_error_messages(message: str, error: Exception = None):
    pass
def output_menu(menu: str):
    pass
def input_menu_choice():
    pass
def output_student_course(student_data: list):
    pass
def input_student_data(student_data: list):
    pass
def read_data_from_file(file_name: str, student_data: list):
    pass
def write_data_to_file(file_name: str, student_data: list):
    pass
#end of function define
```



```
#end of function define
def output_error_messages(message: str, error: Exception = None):...


def output_menu(menu: str):  1 usage
    print(menu)


def input_menu_choice():...


def output_student_course(student_data: list):...


def input_student_data(student_data: list):...


def read_data_from_file(file_name: str, student_data: list):...


def write_data_to_file(file_name: str, student_data: list):...


#end of function define
```

Figure 1.2 - Functions with Arguments



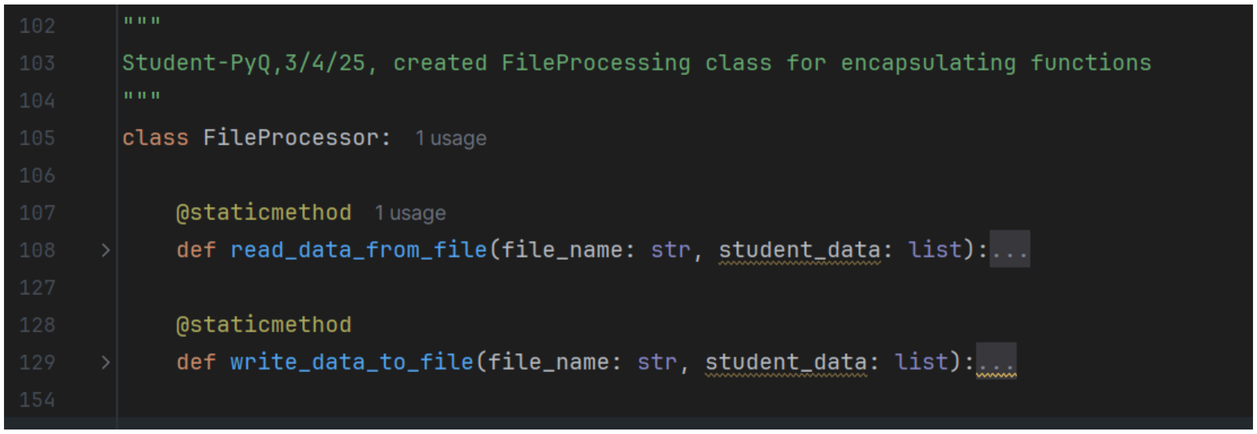
```
        # Present the menu of choices
        """
        Student-PyQ,3/4/25, use new functions
        """
        #print(MENU)
        #menu_choice = input("What would you like to do: ")
        output_menu(menu=MENU)
        menu_choice = input_menu_choice()

        # Input user data
        if menu_choice == "1":  # This will not work if it is an integer!
            """
            Student-PyQ,3/4/25, use new function input_student_data(student_data: list)
            """
            students = input_student_data(student_data=students)
            continue

        # Present the current data
        elif menu_choice == "2":
            """
            Student-PyQ,3/4/25, use new function output_student_course(student_data: list)
            """
            # Process the data to create and display a custom message
            output_student_course(student_data=students)
```

Step 2 - Classes

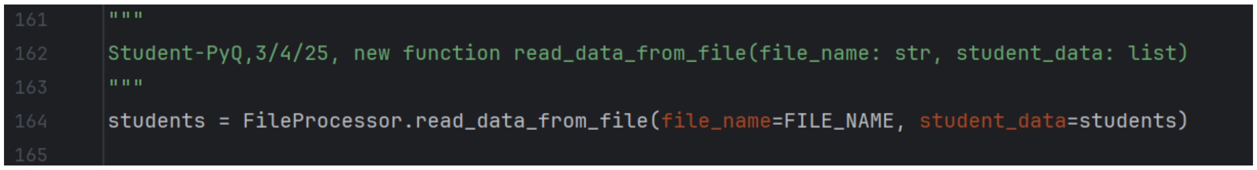
For the second step, I learned about the benefits of classes in organizing code. Classes can provide modularity in code structure by grouping like functions in the same class. Classes are also easier to read, code is logically organized, and can provide scalability over time by adding more functions. The assignment requirement was to create an IO and FileProcessor class. I started with the FileProcess class because there are only functions associated with file processing. Module 6 Lab 3 was also helpful in completing this step. I found that creating classes within a program is relatively simple to do with minimal code (**See Figures 2.1**).

Next, I focused on grouping all the input and output functions in the IO class. I also used the @staticmehtod decorator for the functions so that the functions could be called directly without having to declare a class object first (**See Figures 2.2)**. Descriptive document strings are useful for documenting what a class or function does and what parameters are used for. I used descriptive document strings for the IO, FileProcessor classes, and their associated functions (**See Figures 2.3**).

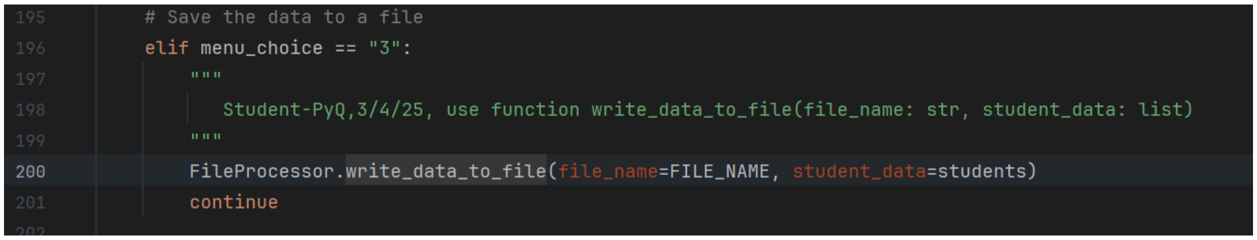
Figures 2.1 - FileProcessor Class



```
"""
Student-PyQ,3/4/25, created FileProcessing class for encapsulating functions
"""
class FileProcessor:  1 usage

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

    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):...

```

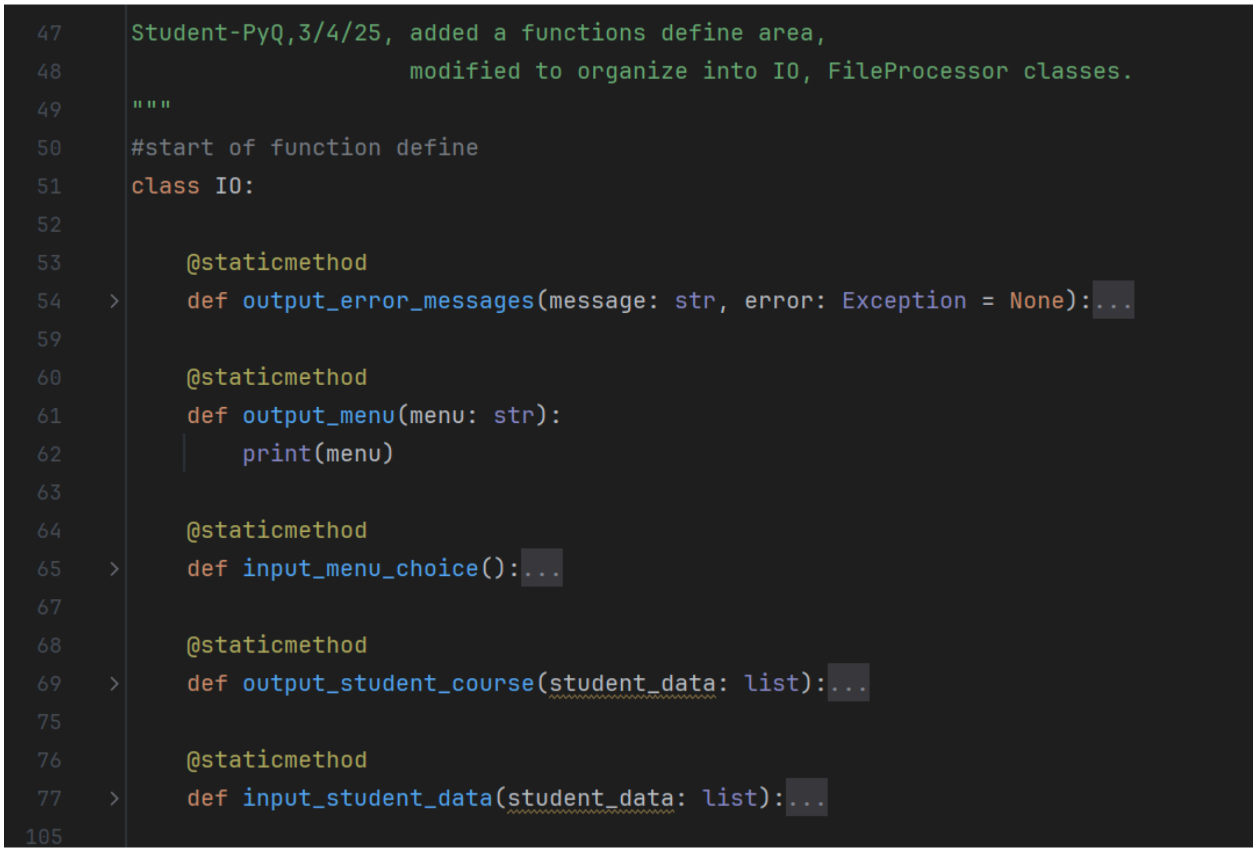


```
"""
Student-PyQ,3/4/25, new function read_data_from_file(file_name: str, student_data: list)
"""
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

```

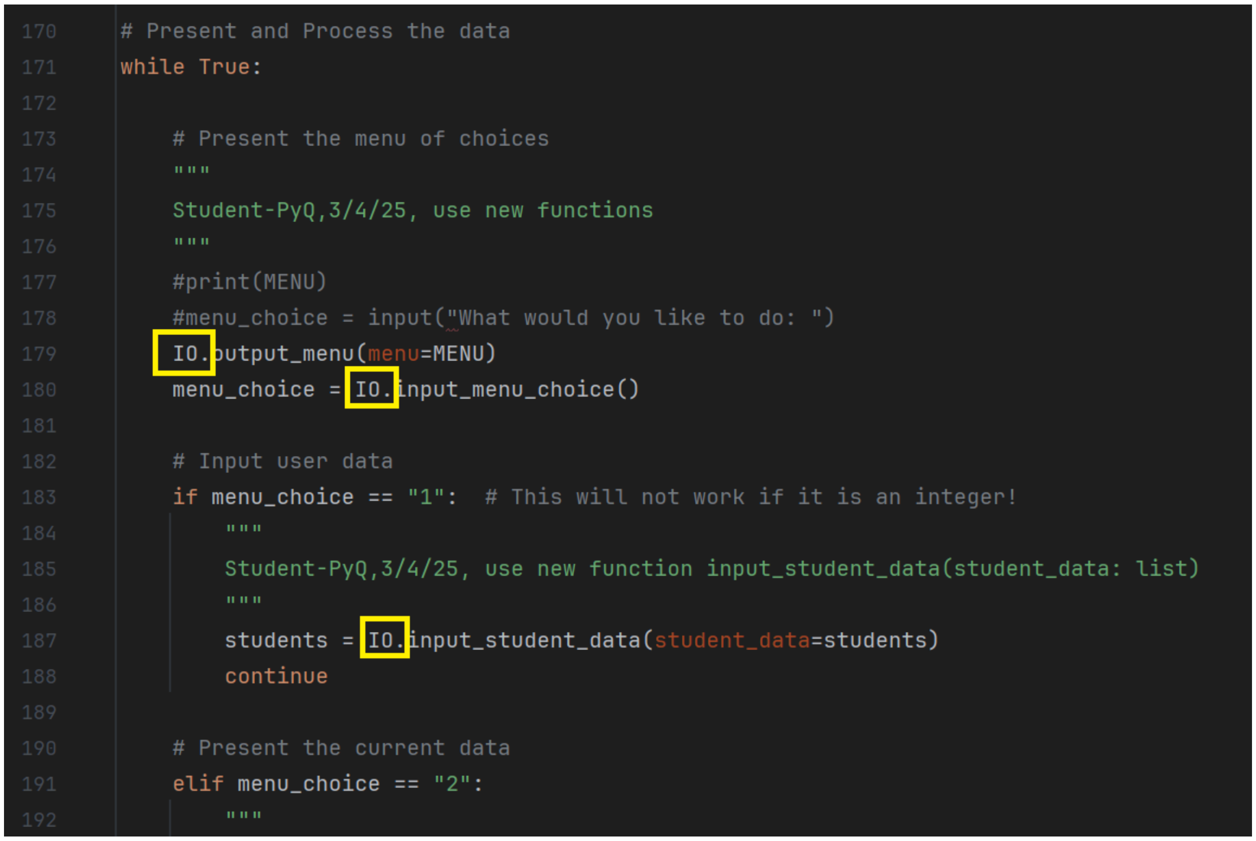


```
    # Save the data to a file
    elif menu_choice == "3":
        """
            Student-PyQ,3/4/25, use function write_data_to_file(file_name: str, student_data: list)
        """
        FileProcessor.write_data_to_file(file_name=FILE_NAME, student_data=students)
        continue
```

Figures 2.2 - IO Class

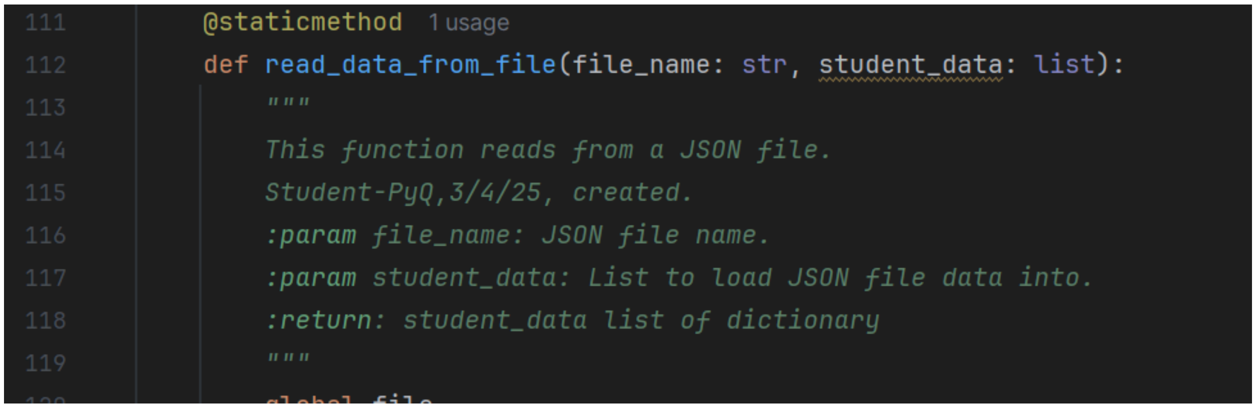


```
Student-PyQ,3/4/25, added a functions define area,
                    modified to organize into IO, FileProcessor classes.
"""
#start of function define
class IO:

    @staticmethod
    def output_error_messages(message: str, error: Exception = None):...

    @staticmethod
    def output_menu(menu: str):
        print(menu)

    @staticmethod
    def input_menu_choice():...

    @staticmethod
    def output_student_course(student_data: list):...

    @staticmethod
    def input_student_data(student_data: list):...
```

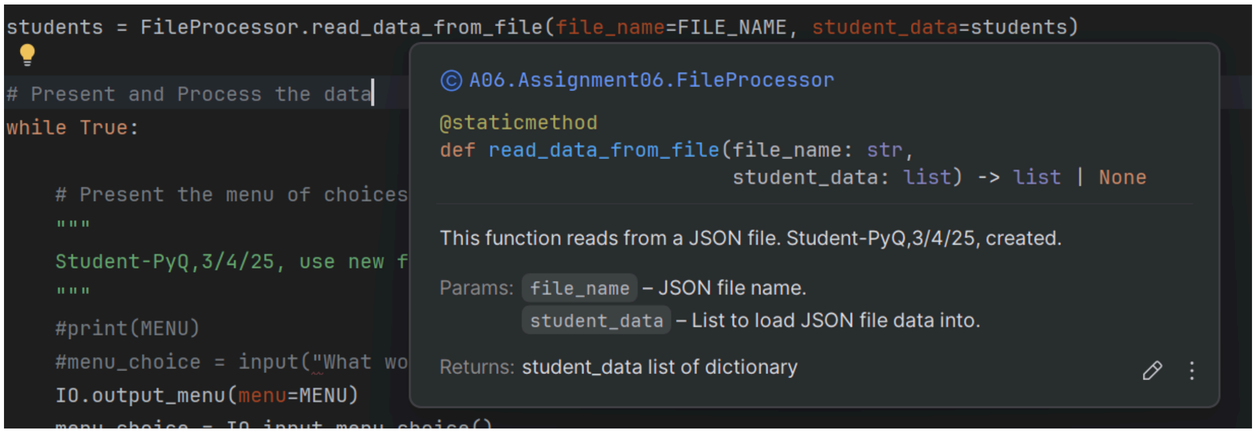


```
# Present and Process the data
while True:

    # Present the menu of choices
    """
    Student-PyQ,3/4/25, use new functions
    """
    #print(MENU)
    #menu_choice = input("What would you like to do: ")
    IO.output_menu(menu=MENU)
    menu_choice = IO.input_menu_choice()

    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        """
        Student-PyQ,3/4/25, use new function input_student_data(student_data: list)
        """
        students = IO.input_student_data(student_data=students)
        continue

    # Present the current data
    elif menu_choice == "2":
        """
```

Figures 2.3 - Descriptive Document Strings



```
    @staticmethod  1 usage
    def read_data_from_file(file_name: str, student_data: list):
        """
        This function reads from a JSON file.
        Student-PyQ,3/4/25, created.
        :param file_name: JSON file name.
        :param student_data: List to load JSON file data into.
        :return: student_data list of dictionary
        """
```



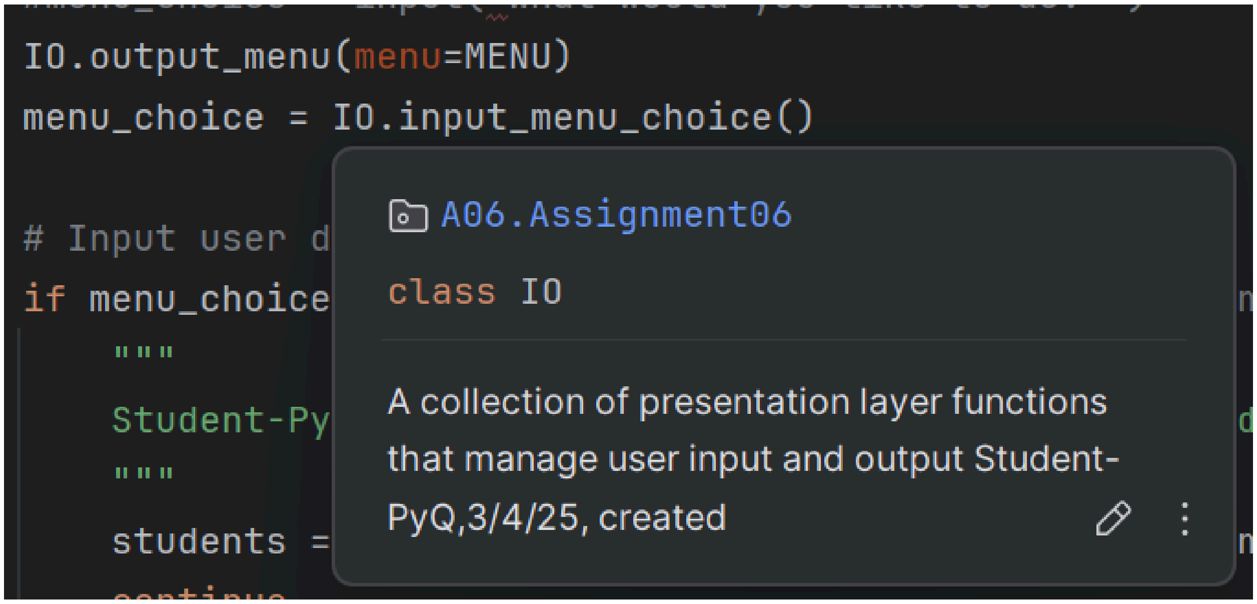
```
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_data=students)

# Present and Process the data
while True:

    # Present the menu of choices
    """
    Student-PyQ,3/4/25, use new f
    """
    #print(MENU)
    #menu_choice = input("What wo
    IO.output_menu(menu=MENU)
```

```
A06.Assignment06.FileProcessor
@staticmethod
def read_data_from_file(file_name: str,
                        student_data: list) -> list | None
```

This function reads from a JSON file. Student-PyQ,3/4/25, created.

Params: `file_name` – JSON file name.
`student_data` – List to load JSON file data into.

Returns: student_data list of dictionary



```
IO.output_menu(menu=MENU)
menu_choice = IO.input_menu_choice()

# Input user d
if menu_choice
    """
    Student-Py
    """
    students =
```

```
A06.Assignment06
class IO
```

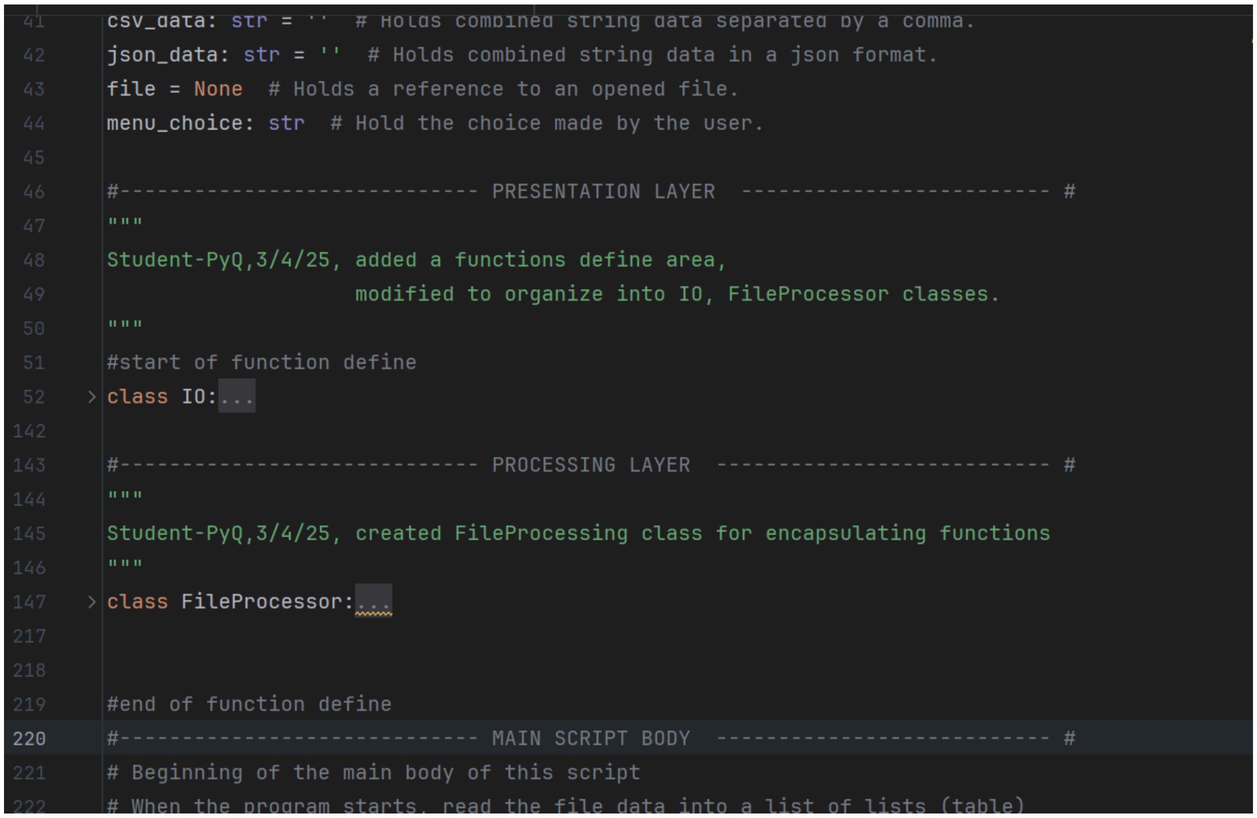
A collection of presentation layer functions that manage user input and output Student-PyQ,3/4/25, created

Step 3 - Separation of Concerns

The last step covers common ways to organize code at a very high level, called separation of concerns. It is essentially a design principle of identifying high level patterns concerning logical areas of separation in code functionality (i.e., data storage tasks, presentation tasks, or business logic/processing tasks). The separation of concerns principle does not change the way the program runs but it does make the code more manageable over time as more logic or complexity is added. For this assignment, I added a data layer, presentation layer, and processing layer (**See Figure 3.1**).

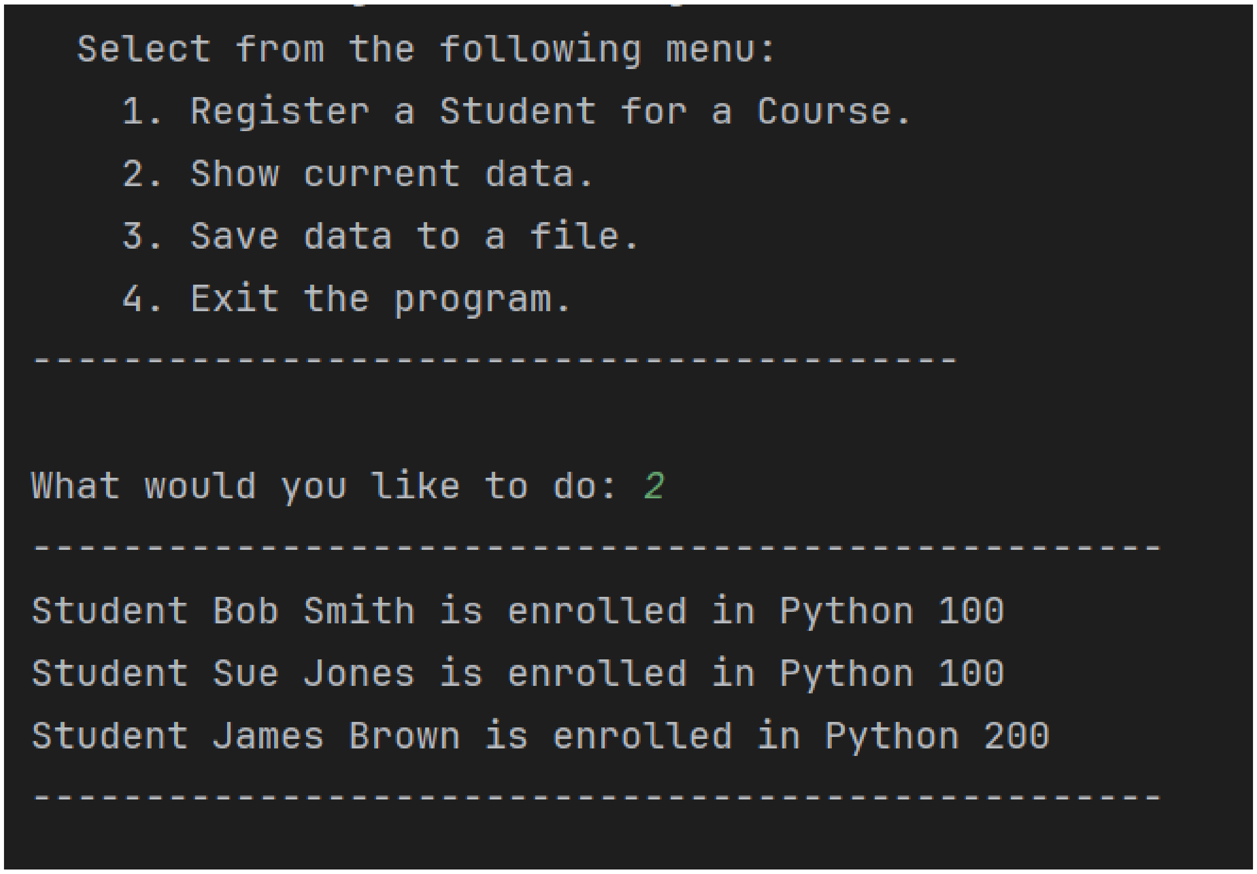
The last task for this step is to test that the code runs successfully in both PyCharm IDE and console terminal (**See Figures 3.2**).

Figures 3.1 - Presentation, Processing, Data Layers

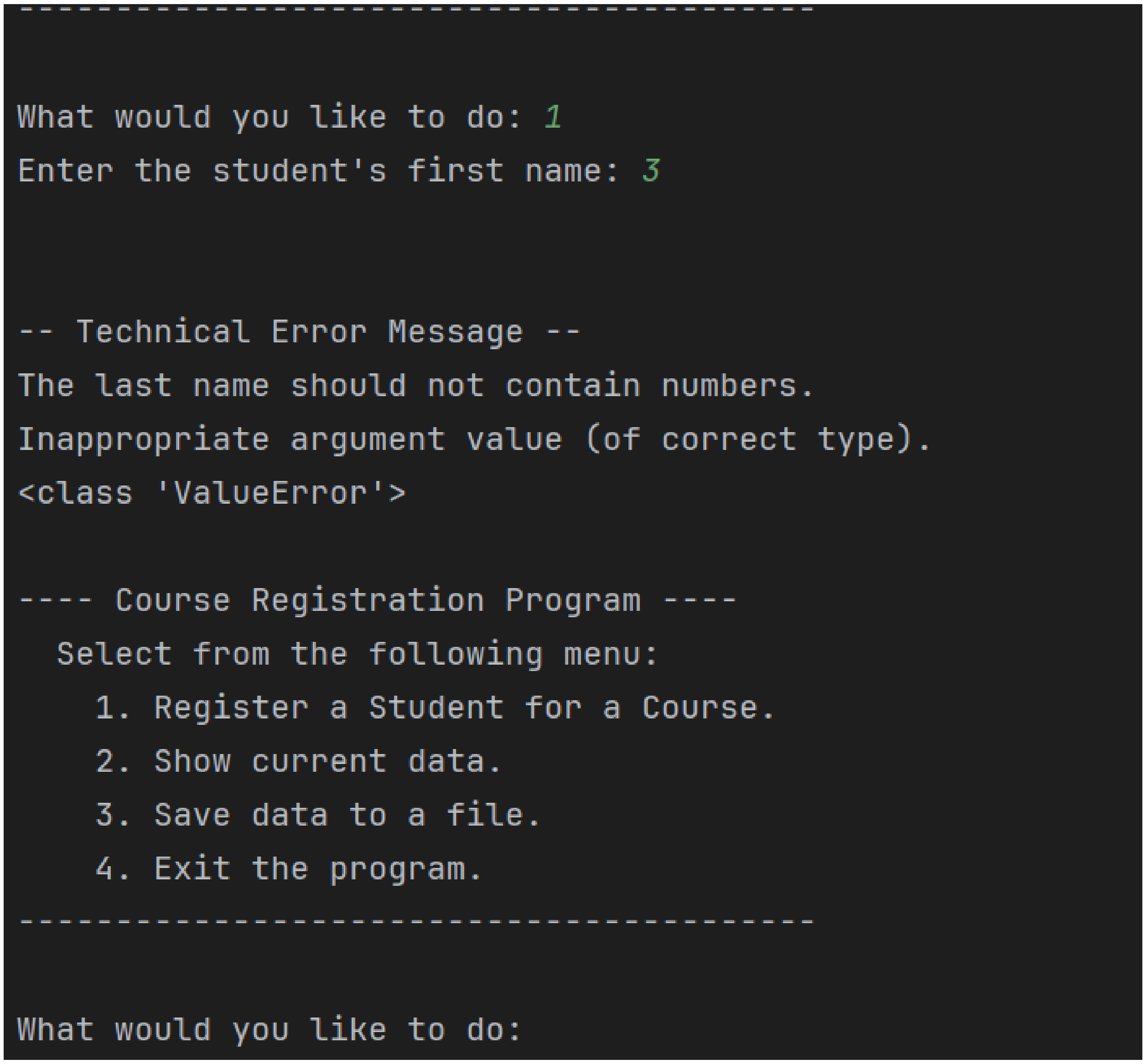


```
csv_data: str = ''  # Holds combined string data separated by a comma.
json_data: str = ''  # Holds combined string data in a json format.
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.

#---------------------------- PRESENTATION LAYER  ------------------------- #
"""
Student-PyQ,3/4/25, added a functions define area,
                    modified to organize into IO, FileProcessor classes.
"""
#start of function define
class IO:...

#---------------------------- PROCESSING LAYER  --------------------------- #
"""
Student-PyQ,3/4/25, created FileProcessing class for encapsulating functions
"""
class FileProcessor:...


#end of function define
#---------------------------- MAIN SCRIPT BODY  --------------------------- #
# Beginning of the main body of this script
# When the program starts, read the file data into a list of lists (table)
```

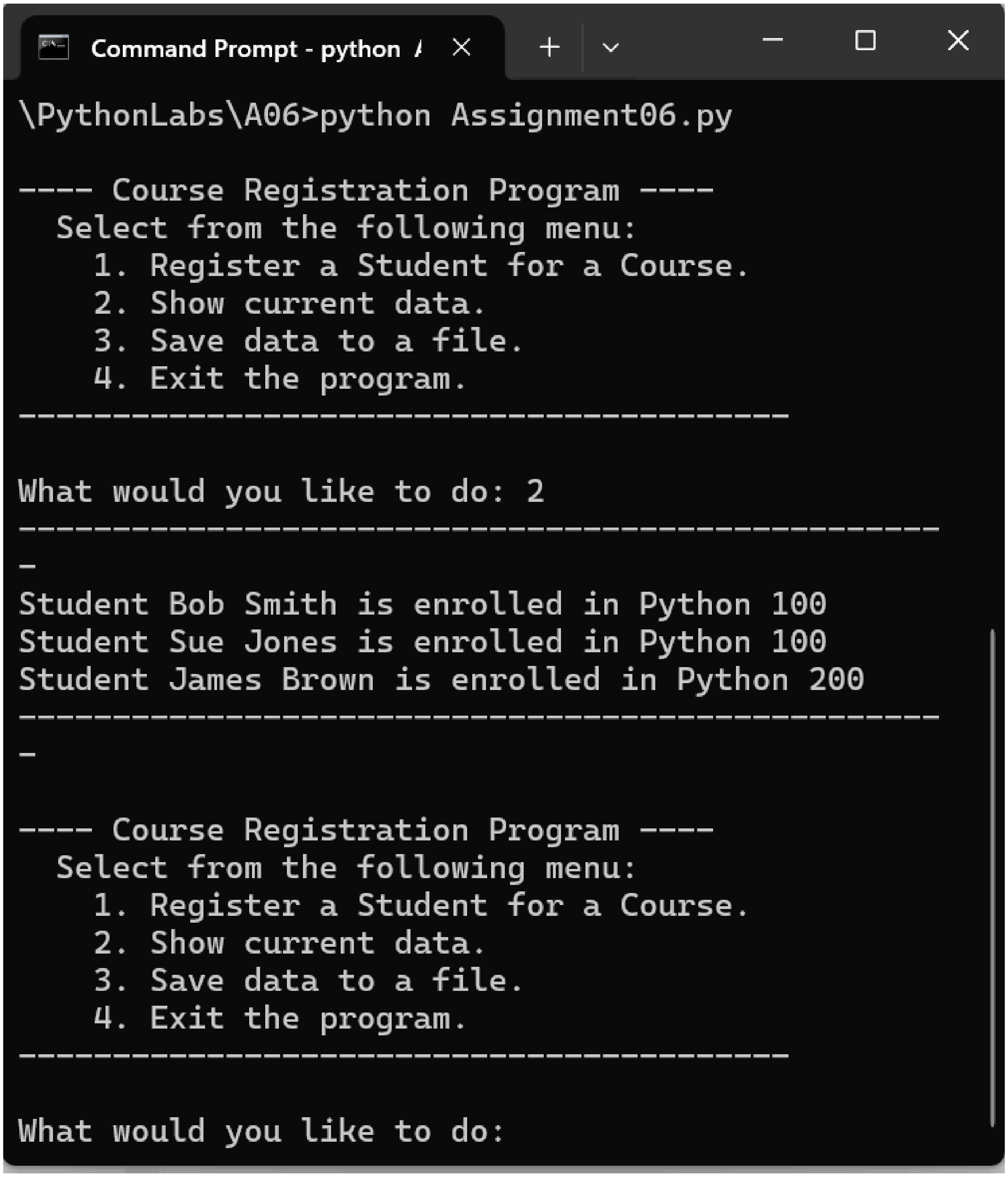
Figures 3.2 - PyCharm and Console Outputs



```
  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 Python 100
Student Sue Jones is enrolled in Python 100
Student James Brown is enrolled in Python 200
--------------------------------------------------
```



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

What would you like to do: 1
Enter the student's first name: 3


-- Technical Error Message --
The last name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>

---- 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:
```



```
\PythonLabs\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.
-----------------------------------------

What would you like to do: 2
-------------------------------------------------
-
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student James Brown is enrolled in Python 200
-------------------------------------------------
-

---- 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:
```

Summary

In summary, this assignment theme is all about maintainability and having well organized code through the use of functions, classes, and the separation of concerns design principle. Module 6 is also a foundation to learning object oriented programming.

References

Root, R. (2025). Module 06 - Functions. In IT FDN 110 A Winter 2025, *Introduction to Programming with Python*, (pp. 1-19) University of Washington

Arya, A. (Nov 5, 2023). Mod06 - SeparationsOfConcern [Mod06-Lab03 - Review issues]. YouTube. https://www.youtube.com/watch?v=fapZdUP-vdw