Jennifer Grace L. Duldulao
05/24/2025
Foundations Of Programming: Python
Assignment 06
https://github.com/hihellojini/IntroToProg-Python-Mod06

Functions

Introduction

In the last module, I learned about dictionary and JSON file and how these tools are very popular to programmers due to its ability to store data in key-value pairs and help data to be organized, flexible and readable. I also learned about error handling particularly the try-except block. In module 5, we also started sharing our work to the class using Github.

In this module, we will be making a program similar to module 5. But this time, we will be defining functions and calling those functions in the main body of the script. We will also be using classes to group together the functions.

Understanding Functions and Classes

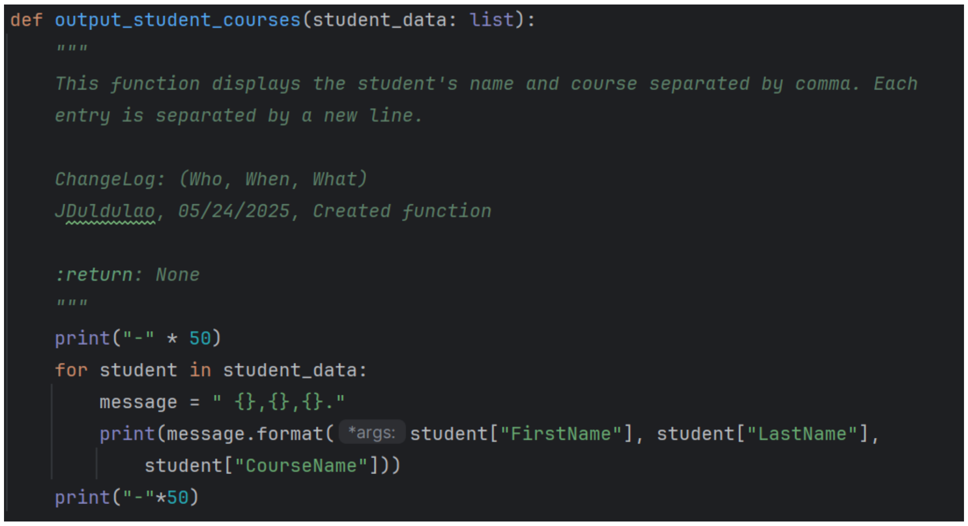
In Python, ***functions*** are block of statements that return a task. The purpose is to put together repeatedly used code and make into a function. The function can then be called in the code and can be used over and over again in any part of the code without re-writing the same code all over again.

Some functions that are already built-in Python. Examples of these are the ***print()*** and ***input()*** functions. However, programmers may also create their own functions by using the syntax ***def name_of_function (parameters)***.

Figure 1 is an example of user-defined function. I used the keyword ***def*** followed by the function name ***output_student_courses*** and the parameter ***student_data: list***. Inside the function is the descriptive documents strings that explains what the function does, the change log and what it returns if there is any. This is then followed by the body and the return expression (if any) using the keyword ***return***.

When the function is called in the script, the simple way of calling it is simply to write the function name.

A function can also be called inside a function. Going back to Figure 1, you will see the built-in function ***print()*** is called.



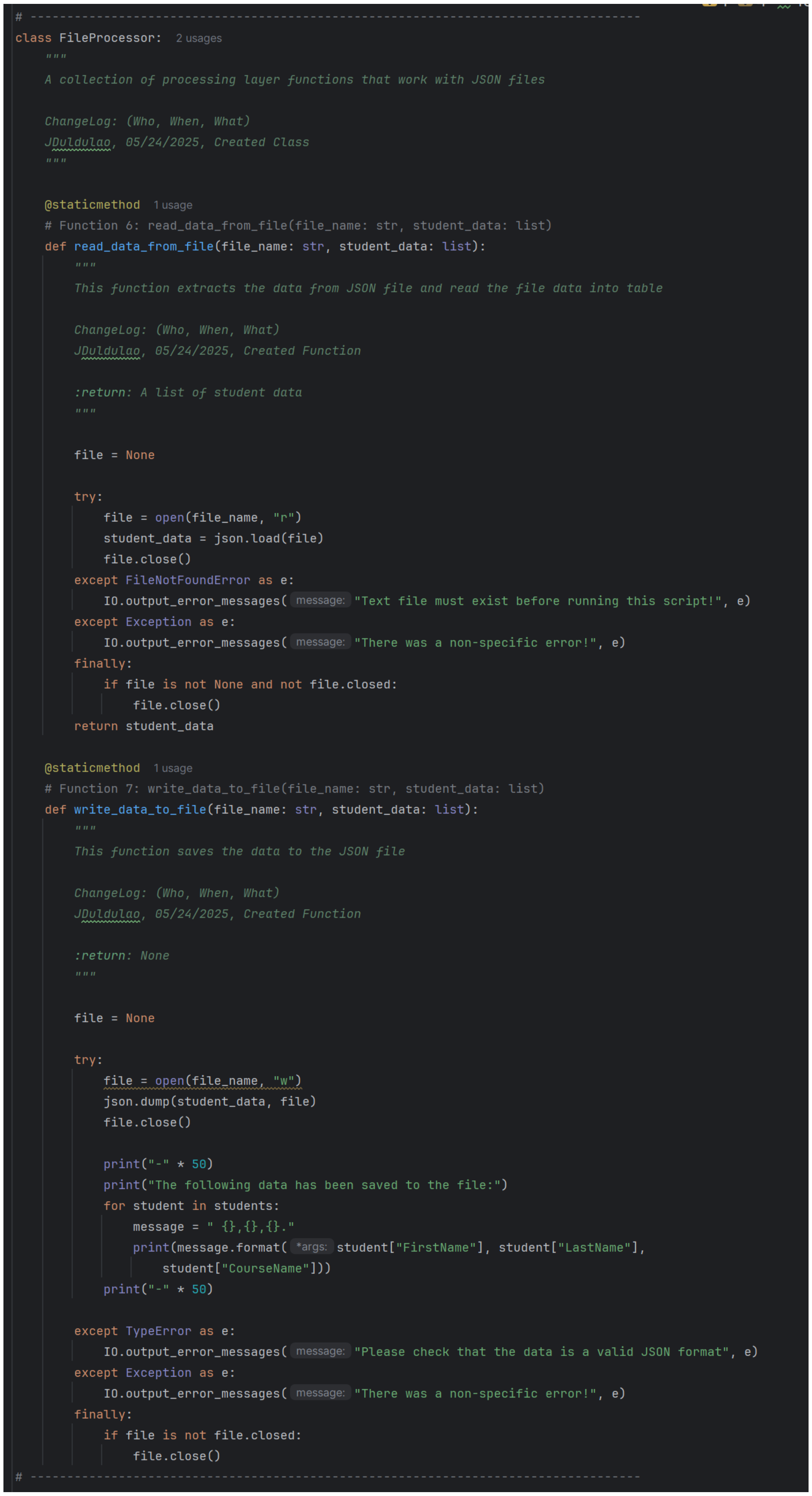
```python
def output_student_courses(student_data: list):
    """
    This function displays the student's name and course separated by comma. Each
    entry is separated by a new line.

    ChangeLog: (Who, When, What)
    JDuldulao, 05/24/2025, Created function

    :return: None
    """
    print("-" * 50)
    for student in student_data:
        message = " {},{},{}."
        print(message.format( *args: student["FirstName"], student["LastName"],
            student["CourseName"]))
    print("-"*50)
```

Figure 1. Example of a Function

A variable defined inside the function is called a local variable and is used exclusively inside the function block. If a variable is defined outside of the function, the variable is global and can be used in the entire script including in the function. One of the learnings I have while making the program is that, while you can use a global variable in the function, best practice is to use local variables and if data are pass to a function, use parameters and return values. This allows the function to have clear boundaries and if anything is changed or mistakenly changed, it does not affect the whole script or vice-versa. Hence, it makes testing and debugging easier.

Classes, on the other hand, are user defined template for creating objects that let users' group together data and functions for ease of management. Classes are created using the keyword ***class*** followed by the class name.

Figure 2 is an example of ***class FileProcessor***. In this class are functions ***read_data_from_file()*** and ***write_data_to_file()***. This class group together the functions that work with JSON files. In every instance that class FileProcessor is used, an object is created which will have it's own space in the computer's memory and independent from the another object that was also created using the same class FileProcessor.



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

    ChangeLog: (Who, When, What)
    JDuldulao, 05/24/2025, Created Class
    """

    @staticmethod  1 usage
    # Function 6: read_data_from_file(file_name: str, student_data: list)
    def read_data_from_file(file_name: str, student_data: list):
        """
        This function extracts the data from JSON file and read the file data into table

        ChangeLog: (Who, When, What)
        JDuldulao, 05/24/2025, Created Function

        :return: A list of student data
        """

        file = None

        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 this script!", e)
        except Exception as e:
            IO.output_error_messages( message: "There was a non-specific error!", e)
        finally:
            if file is not None and not file.closed:
                file.close()
        return student_data

    @staticmethod  1 usage
    # Function 7: write_data_to_file(file_name: str, student_data: list)
    def write_data_to_file(file_name: str, student_data: list):
        """
        This function saves the data to the JSON file

        ChangeLog: (Who, When, What)
        JDuldulao, 05/24/2025, Created Function

        :return: None
        """

        file = None

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

            print("-" * 50)
            print("The following data has been saved to the file:")
            for student in students:
                message = " {},{},{}."
                print(message.format( *args: student["FirstName"], student["LastName"],
                    student["CourseName"]))
            print("-" * 50)

        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 is not file.closed:
                file.close()
# -----------------------------------------------------------------------------------
```

Figure 2. Example of a Class

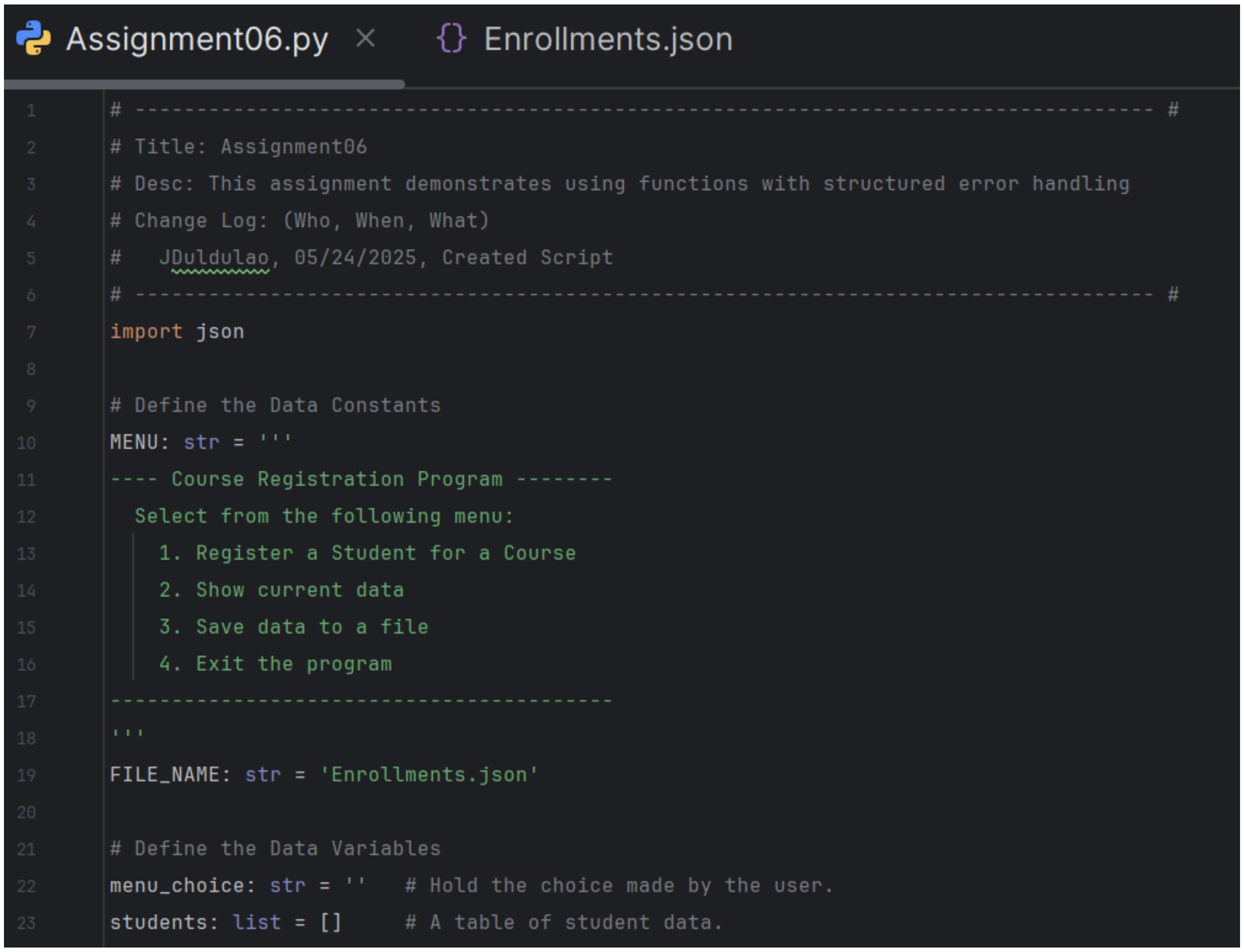
Importance of Separation of Concerns Pattern

Separation of Concerns is important in programming because it breaks down complex systems into smaller and more manageable parts. This helps programmers understand the program easily as the concerns are separated in modules or components so they can work on one component at a time. This also makes the program much easier to maintain and debug.

In this module, this is practiced by writing the script with comments that describe the purpose and structure of the code.

Assignment Discussion

In Assignment 6, I started the code with a header that describes about the program. It is then followed by the import line, and data constants and variables definition. Figure 3.

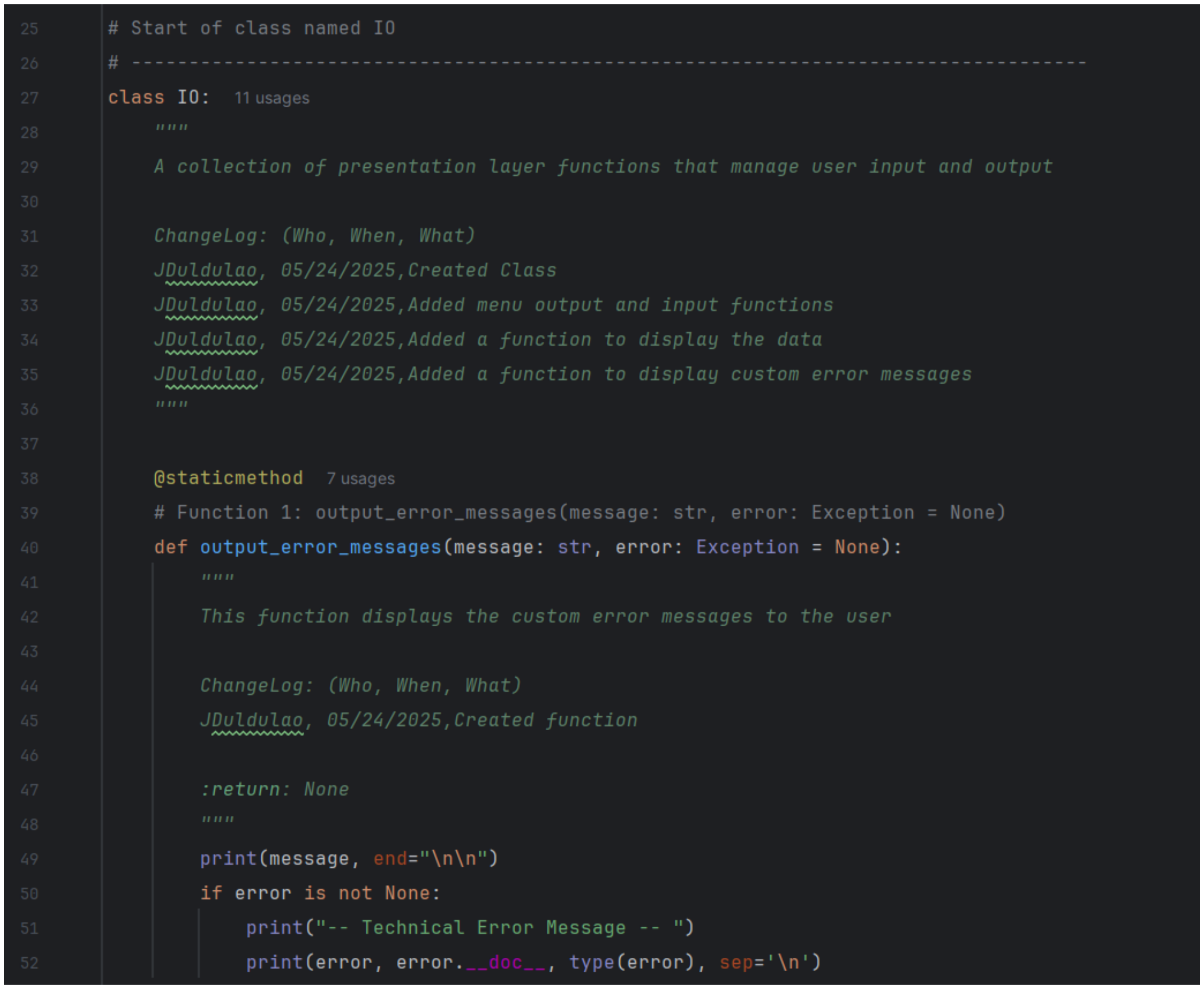


Assignment06.py × Enrollments.json

```
# ------------------------------------------------------------------------------------- #
# Title: Assignment06
# Desc: This assignment demonstrates using functions with structured error handling
# Change Log: (Who, When, What)
#   JDuldulao, 05/24/2025, Created Script
# ------------------------------------------------------------------------------------- #
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
-----------------------------------------
'''
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 3. Header, Import, and Data Definition

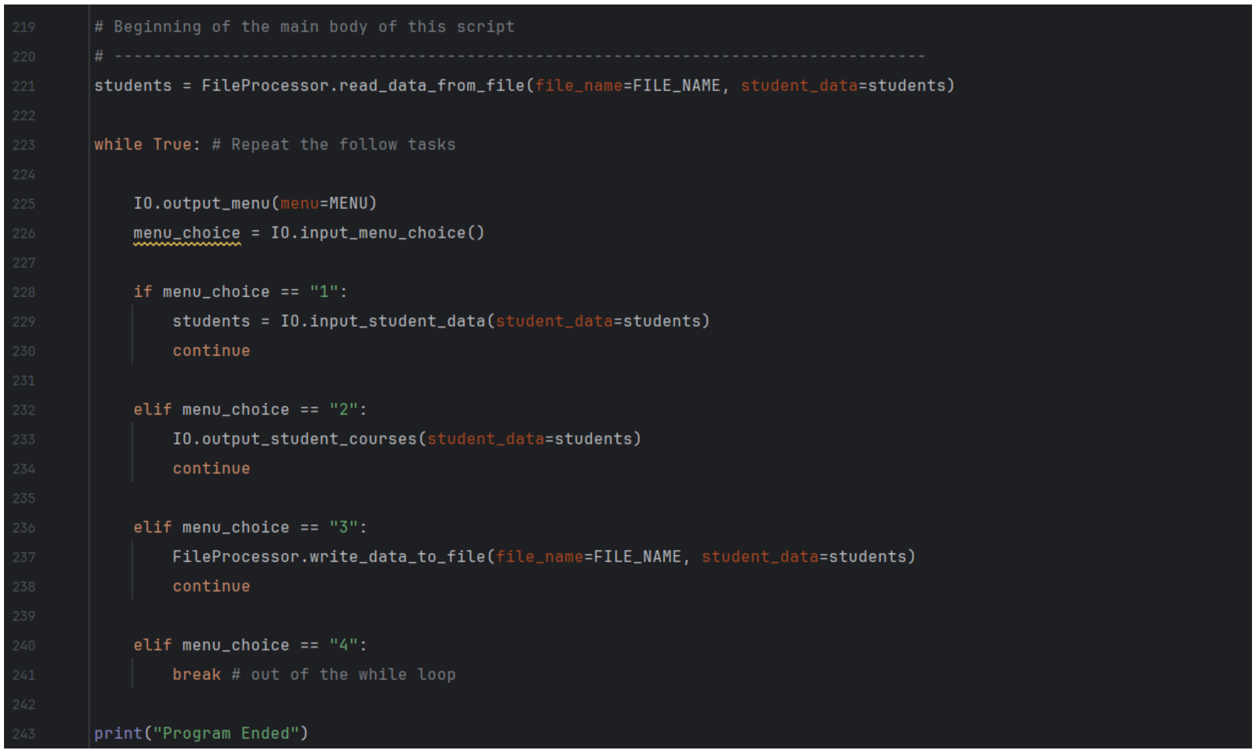
Next, I defined class IO and class FileProcessor. Each class contained different defined functions with parameters. Figure 4 below shows part of the program. Note that, each class and function have descriptive document strings to explain what the class or function is all about. This help me and other programmers understand what the code is all about and what it is use for.



```python
# Start of class named IO
# ----------------------------------------------------------------------------------
class IO:  11 usages
    """
    A collection of presentation layer functions that manage user input and output

    ChangeLog: (Who, When, What)
    JDuldulao, 05/24/2025,Created Class
    JDuldulao, 05/24/2025,Added menu output and input functions
    JDuldulao, 05/24/2025,Added a function to display the data
    JDuldulao, 05/24/2025,Added a function to display custom error messages
    """


    @staticmethod  7 usages
    # Function 1: output_error_messages(message: str, error: Exception = None)
    def output_error_messages(message: str, error: Exception = None):
        """
        This function displays the custom error messages to the user

        ChangeLog: (Who, When, What)
        JDuldulao, 05/24/2025,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')
```

Figure 4. Classes and Functions Definition

Figure 5 is the main body of the script. The script starts by reading the data from file Enrollments.json. If the file does not exist, an error handling script is called and then it presents a menu of choices for the user to choose. The program will execute the script depending on the user's choice. Note that instead of writing the code inside the while loop like what we did in Assignment 5, in here, you will see that the codes are written into functions and grouped in classes. And depending on the user's choice, the associated function is called using the parameters and the function returns a value.



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

while True: # Repeat the follow tasks

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

    if menu_choice == "1":
        students = IO.input_student_data(student_data=students)
        continue

    elif menu_choice == "2":
        IO.output_student_courses(student_data=students)
        continue

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

    elif menu_choice == "4":
        break # out of the while loop

print("Program Ended")
```

Figure 5. Main Body of Script

The program was tested to run in PyCharm, Command Prompt and IDLE and it works well in all of them.

Summary

In this module, I learned about how functions and classes work. I find them very useful especially as the program becomes more complex. They make the code more organize, easy to understand and easy to troubleshoot and debug. It is also very helpful to programmers especially if the code has to be re-used multiple times as the programmer does not need to re-write the code all the time. The function can be just called every time it is needed in code which also helps minimize errors.