Name: Kornel Cieslik

Date: 11/18/2024

Class: IT FDN 110 B

Github Link: https://github.com/Kornel93/IntroToProg-Python

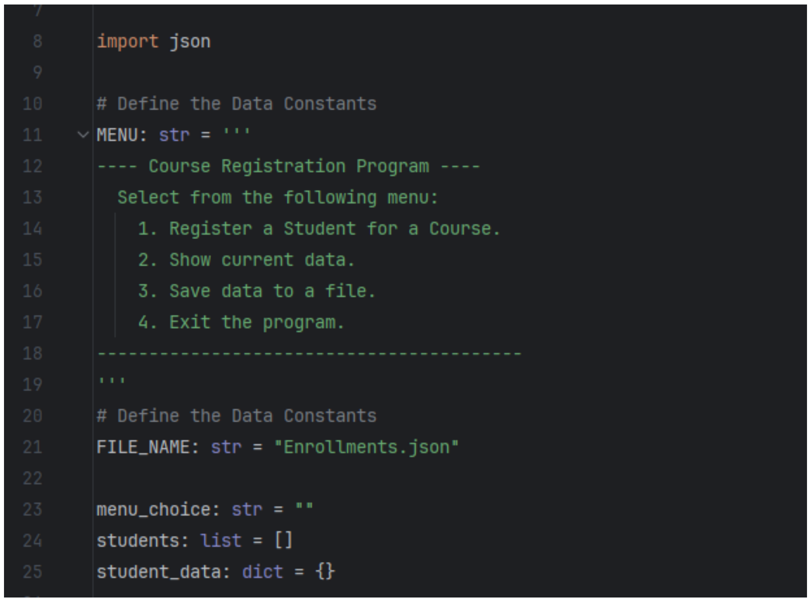
Assignment 06 – Classes and Functions

Introduction:

The subject matter for this assignment continues to build upon what our previous focus was upon. Loops, conditional logic, dictionaries, collections, error handling procedures while now introducing the concept of classes and defined functions. The goal of the assignment is to, once again, develop a menu for someone looking to register for a course, receive input values from the user but it will be saved in a JSON file.

Creation/Thought Process:

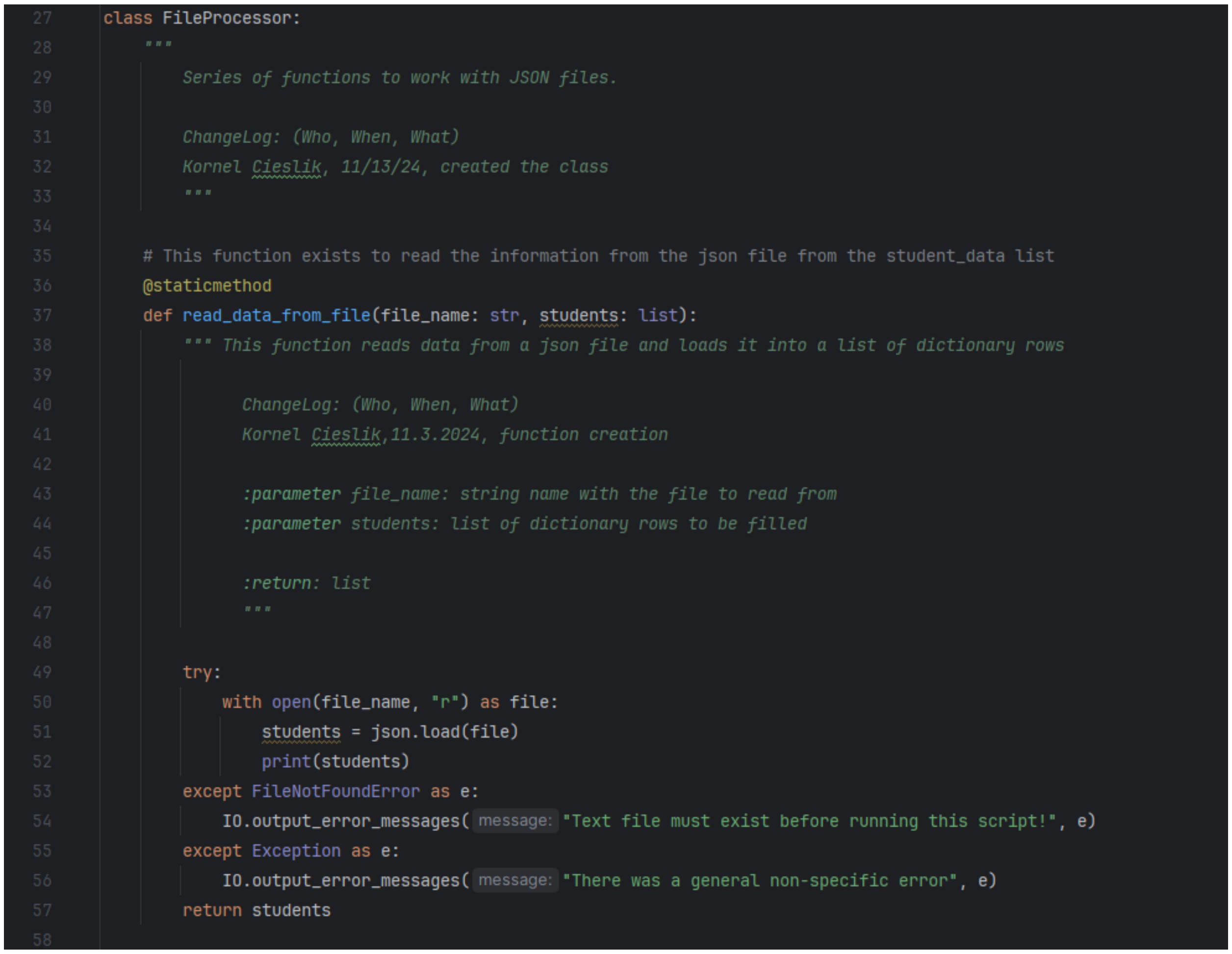
While the main goal of the project is the same; creating a student registration form, we are also focusing on the practice of utilizing a JSON file format. Best practice dictates that the beginning of the code provides a header of who, what, and where. **Fig 1.1** shows the declaration of variables, establishment of the enrollments file, and the menu options that will be offered to the user. Note the import of the json function at the top.



```
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.json"

menu_choice: str = ""
students: list = []
student_data: dict = {}
```

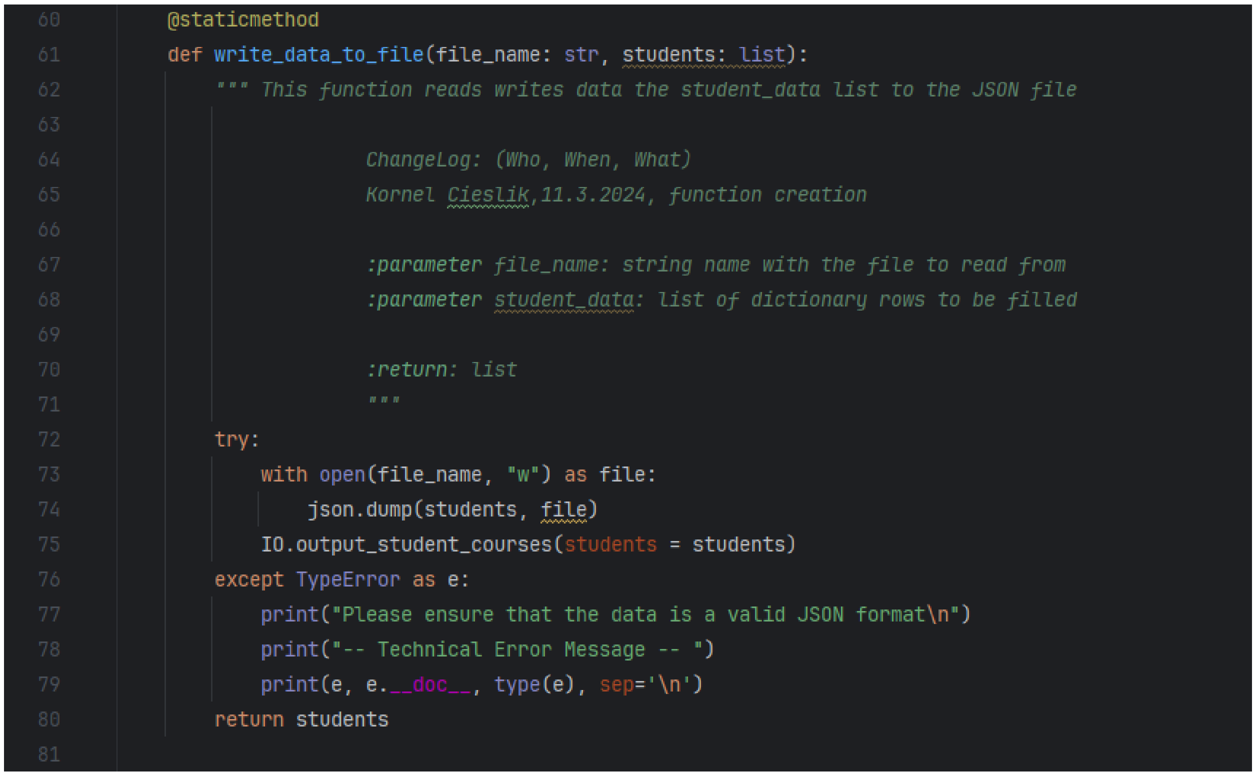
Fig 1.1 – Establishment of menu, variables, and enrollments file.

Prior to any menu choices being offered up to the user for their input, we first create classes to handle some of the functionality of the code. Classes are a method to create a user-defined data type that helps organize and clean up the code. The first class is the FileProcessor class which captures the reading and writing from the file. This can be seen in **Fig 1.2** & **Fig 1.3.** We define each function as a static method in order to ensure that it belongs to that particular class rather than an instance of that class. The read data from file function takes two arguments; the name of the file to read data from and a list where student related data will be stored. There is also a try and except block to capture any type of errors that may occur.



```python
class FileProcessor:
    """
        Series of functions to work with JSON files.

        ChangeLog: (Who, When, What)
        Kornel Cieslik, 11/13/24, created the class
        """

    # This function exists to read the information from the json file from the student_data list
    @staticmethod
    def read_data_from_file(file_name: str, students: list):
        """ This function reads data from a json file and loads it into a list of dictionary rows

            ChangeLog: (Who, When, What)
            Kornel Cieslik,11.3.2024, function creation

            :parameter file_name: string name with the file to read from
            :parameter students: list of dictionary rows to be filled

            :return: list
            """

        try:
            with open(file_name, "r") as file:
                students = json.load(file)
                print(students)
        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 general non-specific error", e)
        return students
```

Fig 1.2 – Creation of first class and associated functions.

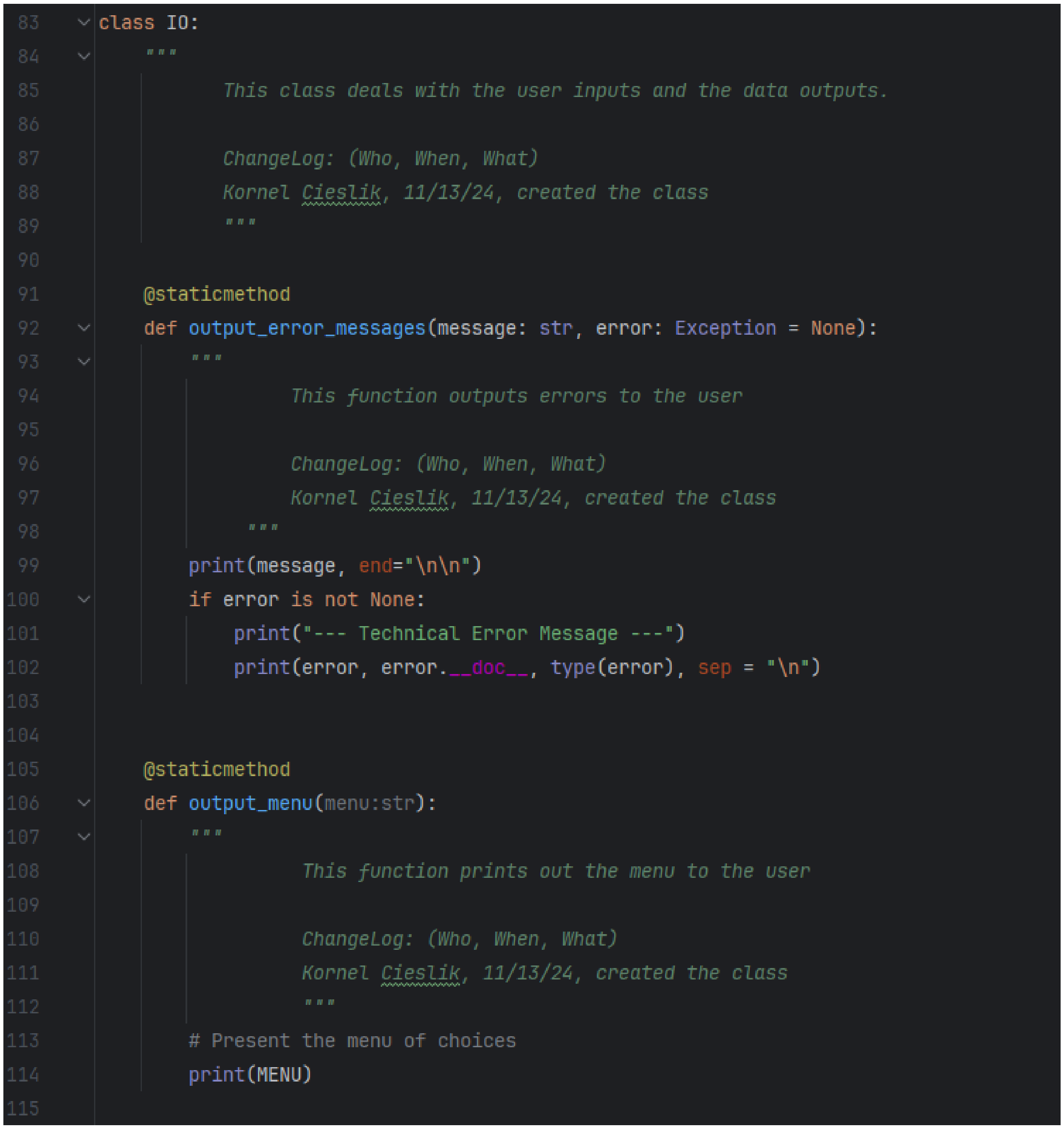


```
    @staticmethod
    def write_data_to_file(file_name: str, students: list):
        """ This function reads writes data the student_data list to the JSON file

                    ChangeLog: (Who, When, What)
                    Kornel Cieslik,11.3.2024, function creation

                    :parameter file_name: string name with the file to read from
                    :parameter student_data: list of dictionary rows to be filled

                    :return: list
                    """
        try:
            with open(file_name, "w") as file:
                json.dump(students, file)
            IO.output_student_courses(students = students)
        except TypeError as e:
            print("Please ensure that the data is a valid JSON format\n")
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep='\n')
        return students
```

Fig 1.3 – The function that writes to the json file.

The write data to file function opens the json file in write mode. Json.dump passes the students list into json format and writes it to our file. There is additional error handling to catch if a type-error exists and returns the input list at the bottom.

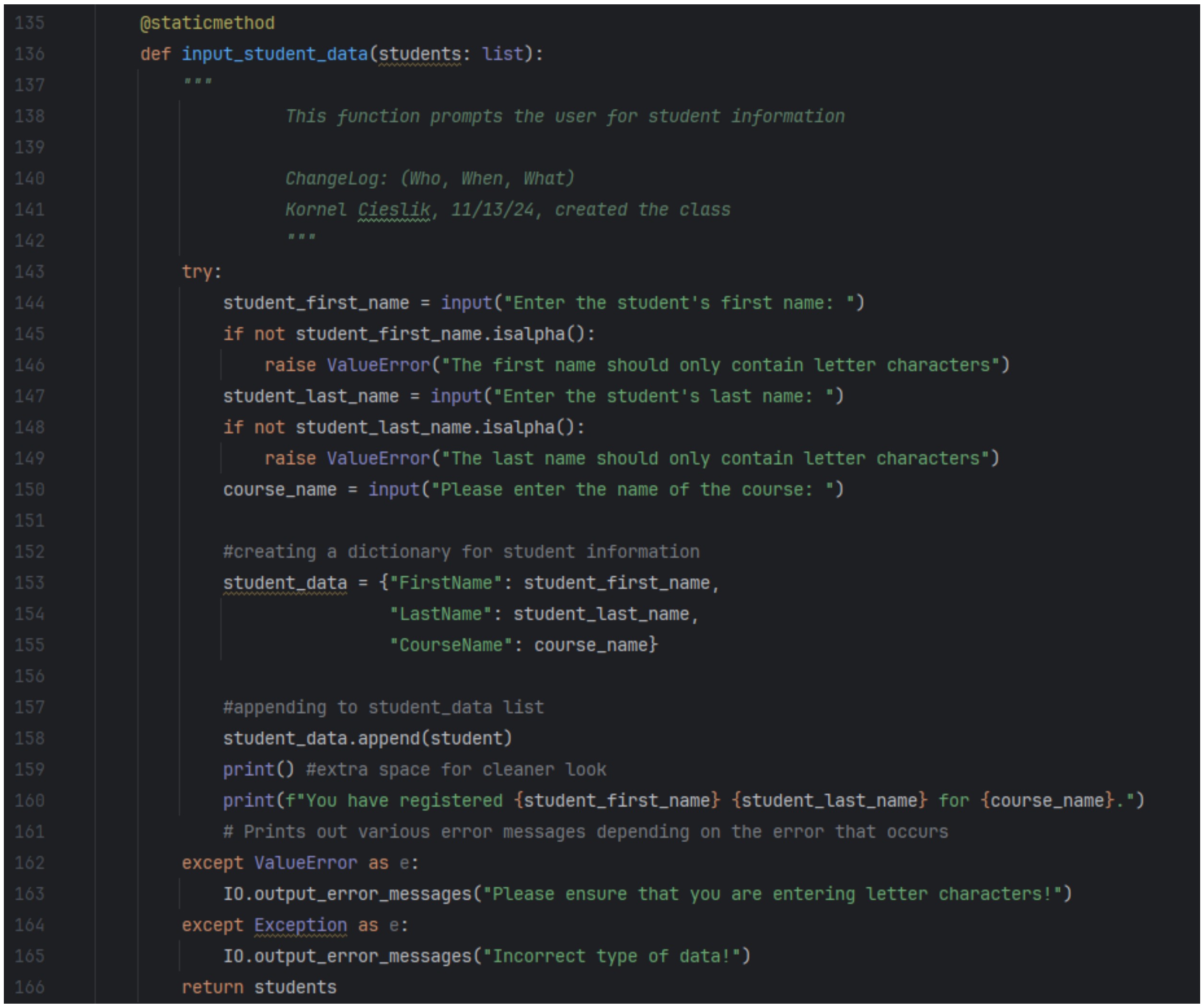
Once the FileProcessor class was developed. We needed to create an additional class to work with all the user inputs and subsequent outputs. The IO class is what contains the functions that perform that work statement. This can be seen in **Fig 1.4.** Two functions are also present in the aforementioned figure. These functions read any output error messages and provides a snippet of what the error may be and the other function prints the menu out that the user will see later when the program runs.



```python
class IO:
    """
        This class deals with the user inputs and the data outputs.

        ChangeLog: (Who, When, What)
        Kornel Cieslik, 11/13/24, created the class
        """

    @staticmethod
    def output_error_messages(message: str, error: Exception = None):
        """
                This function outputs errors to the user

                ChangeLog: (Who, When, What)
                Kornel Cieslik, 11/13/24, created the class
          """
        print(message, end="\n\n")
        if error is not None:
            print("--- Technical Error Message ---")
            print(error, error.__doc__, type(error), sep = "\n")


    @staticmethod
    def output_menu(menu:str):
        """
                 This function prints out the menu to the user

                 ChangeLog: (Who, When, What)
                 Kornel Cieslik, 11/13/24, created the class
                 """
        # Present the menu of choices
        print(MENU)

```

Fig 1.4 – Creation of IO class and two functions.

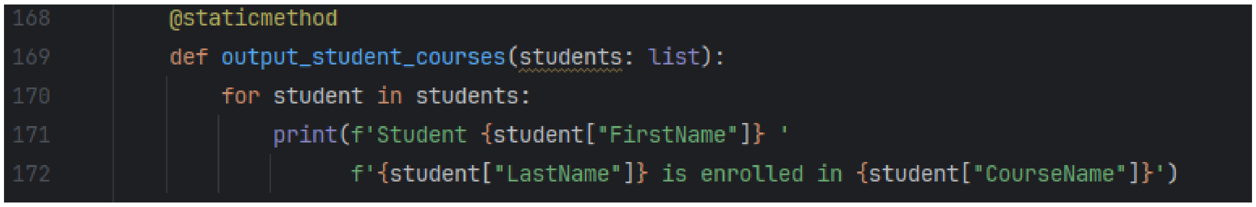
The remaining functions that can be seen in **Fig 1.5** provide options for the user to input a choice for the numbers on the menu which also raises an exception if any values other than the displayed values are selected. The input student data function is where the user will be prompted to input student information. A ValueError is raised if alphanumeric keys are not selected for the first and last name which prompts the user to input information in once more. A dictionary is created within the function to store this new information which is then appended to a list of student information.



```python
    @staticmethod
    def input_student_data(students: list):
        """
                This function prompts the user for student information

                ChangeLog: (Who, When, What)
                Kornel Cieslik, 11/13/24, created the class
                """
        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 letter characters")
            student_last_name = input("Enter the student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError("The last name should only contain letter characters")
            course_name = input("Please enter the name of the course: ")

            #creating a dictionary for student information
            student_data = {"FirstName": student_first_name,
                            "LastName": student_last_name,
                            "CourseName": course_name}

            #appending to student_data list
            student_data.append(student)
            print() #extra space for cleaner look
            print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
            # Prints out various error messages depending on the error that occurs
        except ValueError as e:
            IO.output_error_messages("Please ensure that you are entering letter characters!")
        except Exception as e:
            IO.output_error_messages("Incorrect type of data!")
        return students
```

Fig 1.5 - Creation of the input student data function.

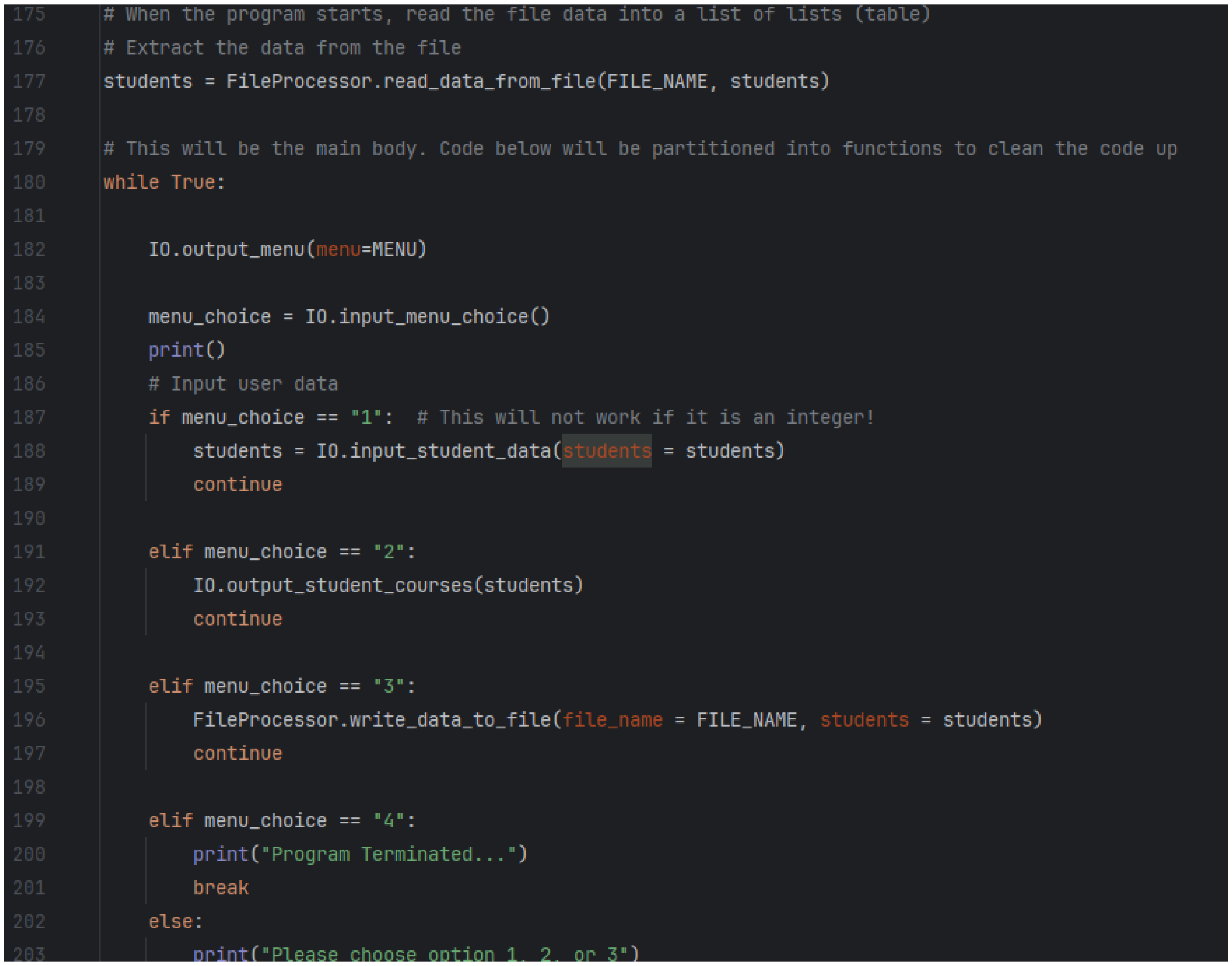
The last function in the class outputs the student courses by iterating through the students list



```python
    @staticmethod
    def output_student_courses(students: list):
        for student in students:
            print(f'Student {student["FirstName"]} '
                  f'{student["LastName"]} is enrolled in {student["CourseName"]}')
```

Fig 1.5 – Creation of output student courses.

Now with the classes and functions completely defined, we can move to the main body where everything will be called as can be seen in **Fig 1.6**. We are extracting the information from the students list in our file name and performing a while loop that provides the menu and performs certain functions dependent on the user's choice.

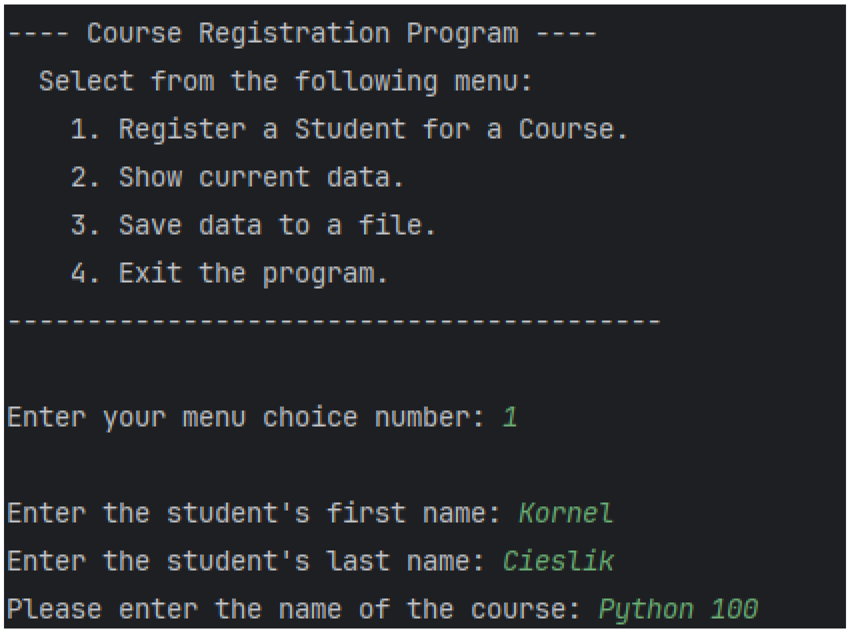


```
# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
students = FileProcessor.read_data_from_file(FILE_NAME, students)

# This will be the main body. Code below will be partitioned into functions to clean the code up
while True:

    IO.output_menu(menu=MENU)

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

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

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

    elif menu_choice == "4":
        print("Program Terminated...")
        break
    else:
        print("Please choose option 1, 2, or 3")
```

Fig 1.6 – Code main body that calls the class functions.

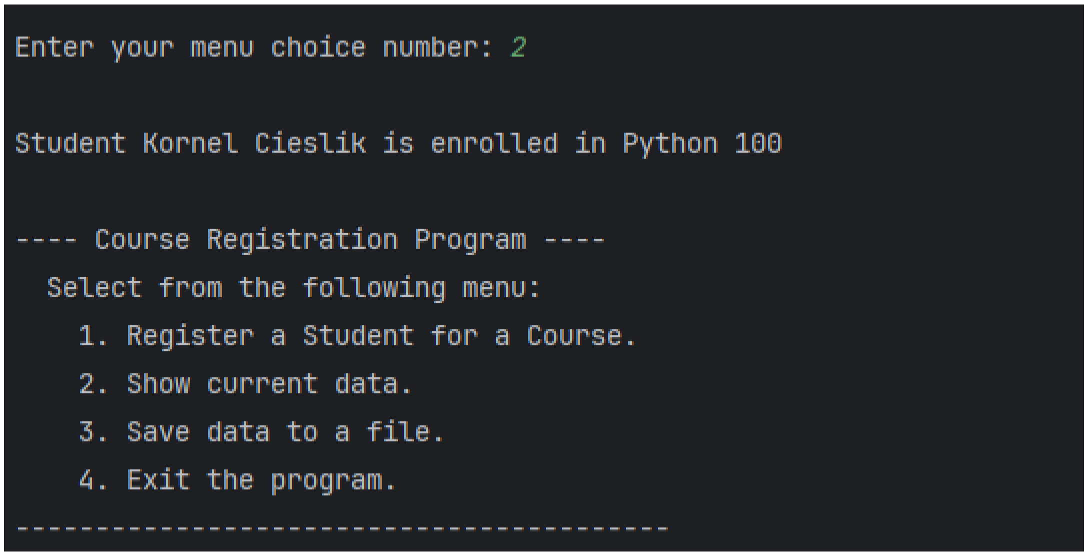
Possible Errors:

I have noticed however that when the code runs, it only contains one student and does not add more to the list. I am not quite sure what causes this and will be bringing it forward for discussion to the class.



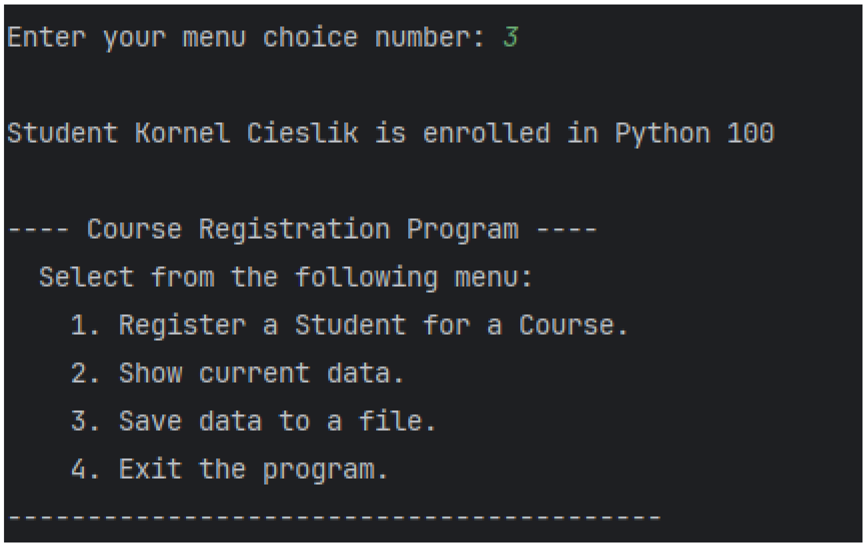
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


Enter your menu choice number: 1

Enter the student's first name: Kornel
Enter the student's last name: Cieslik
Please enter the name of the course: Python 100
```

Fig 1.7 – Menu choice 1 results



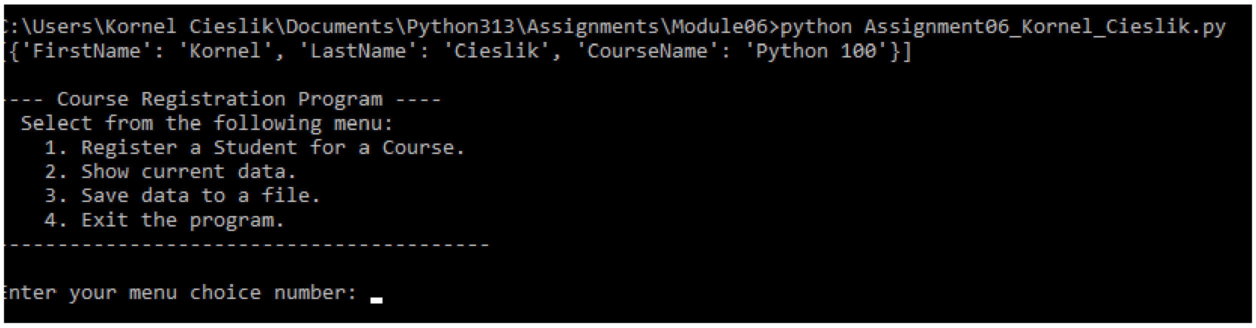
```
Enter your menu choice number: 2

Student Kornel Cieslik is enrolled in 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.
-----------------------------------------
```

Fig 1.8 – Menu choice 2 results



```
Enter your menu choice number: 3

Student Kornel Cieslik is enrolled in 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.
-----------------------------------------
```

Fig 1.9 – Menu choice 3 results



```
:\Users\Kornel Cieslik\Documents\Python313\Assignments\Module06>python Assignment06_Kornel_Cieslik.py
[{'FirstName': 'Kornel', 'LastName': 'Cieslik', 'CourseName': '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.
-----------------------------------------

nter your menu choice number:
```

Fig 2.0 – Code snippet from command terminal