Sashi Chintala

November 25, 2024

Course name: IT FDN 110 A Au 24: Foundations Of Programming: Python

Assignment 06

GitHub:

Advanced Collections and Error Handling

Introduction

In this assignment 06, I learned the core topics of blocks of programming which are functions and classes. Learned how to reuse the functions to perform a set of tasks, and the use of "pass" keyword which tells that code is not yet implemented. Understood the difference between global and local variables and their primary use in the program. Learned few other topics docstrings, and SoC (Separation of concerns patter) which is a design principle to make the code understandable to other developers or teammates in the team to collaborate.

Creating the program

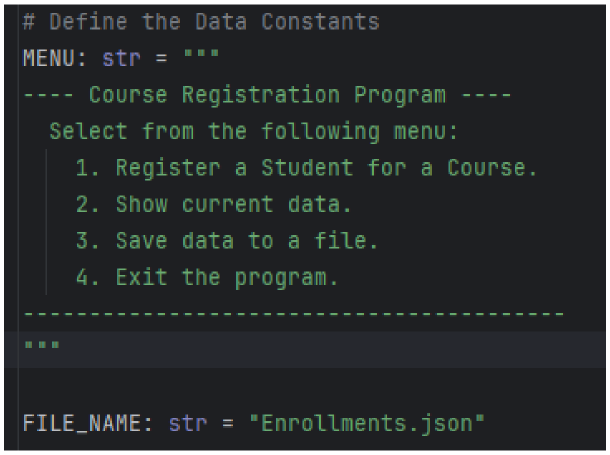
Thanks to Professor Root as he gave us the starter script from the previous module 05 assignment and I started building it. In our module03, 04, and 05 programs, we took student information and used it for student registration. Information regarding interactive console using the prompt and using the four menu choices and we will add more error checking, file handling, duplicate student check, course validation, and display a message at the end showing registrations.

Constants

As usual, we are using two constants for our program,

1. **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.
       ----------------------------------------------
   ```
2. **FILE_NAME: str** and is set to the value "Enrollments.json" to save information

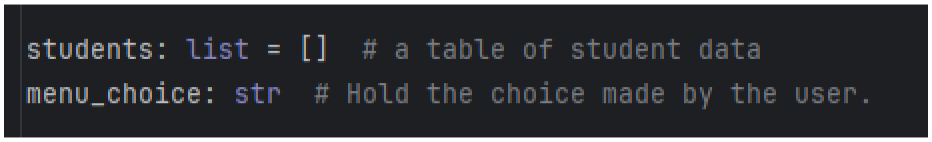
   Figure 1.1: *Constants*



```
# 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"
```

Variables

***Figure 1.2**: list of variables used*



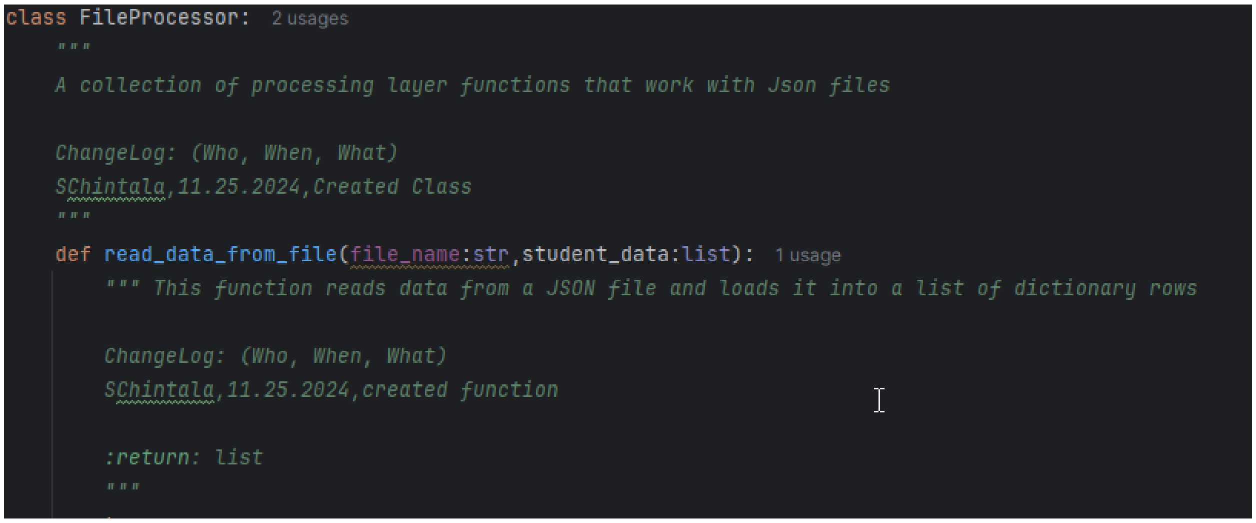
```
students: list = []  # a table of student data
menu_choice: str  # Hold the choice made by the user.
```

Students list to store student data, and menu_choice to store user's menu choice 1-4.

FileProcessor Class

In FileProcessor Class, added two functions, read_data_from_file: Reads data from a JSON file and loads it into a list, and write_data_to_file: Writes data from a list to a JSON file. Benefits of Using a FileProcessor Class are code reusability, error handling, and maintainability.

***Figure 1.3** showing the processing*

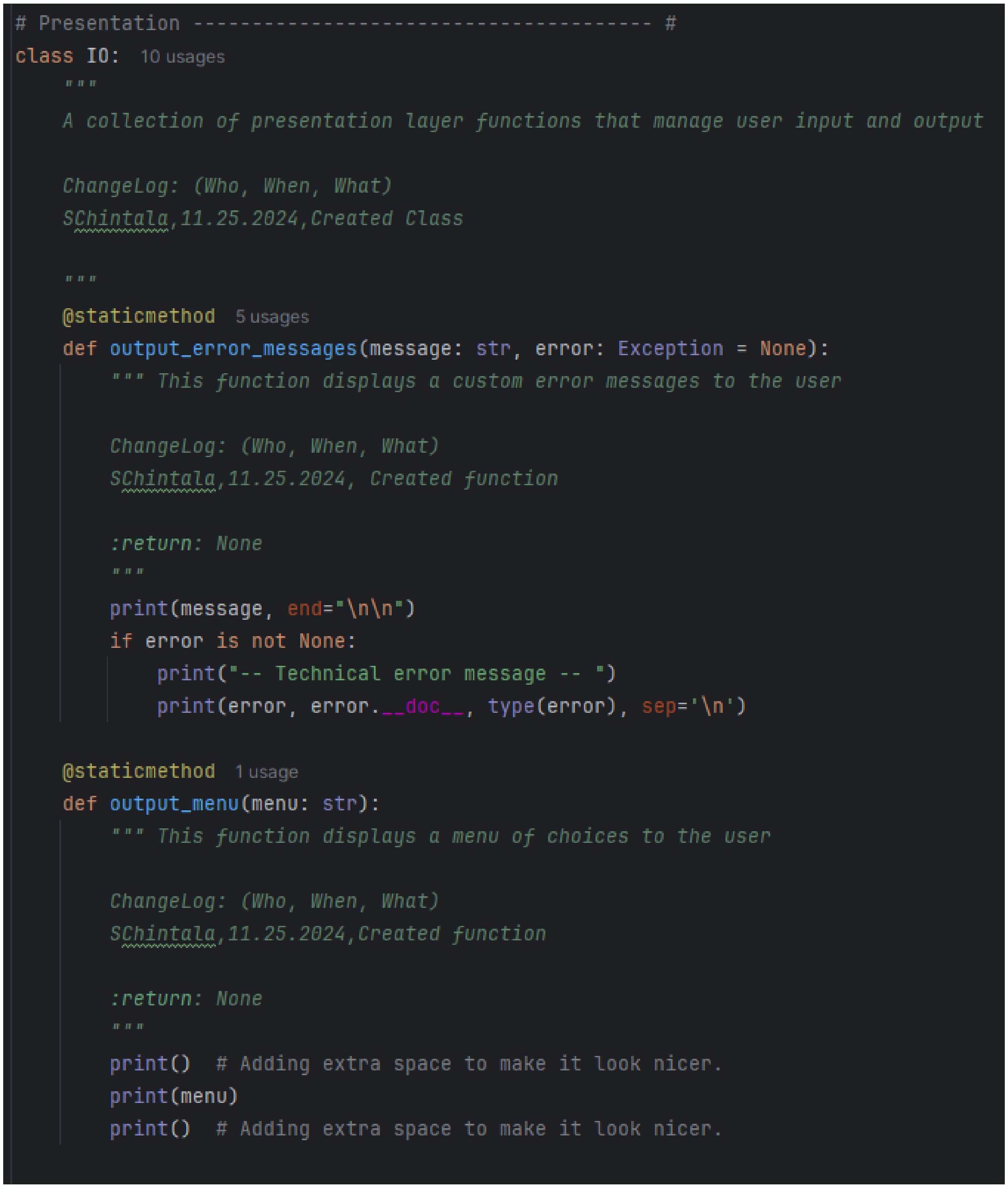


```
class FileProcessor:  2 usages
    """
    A collection of processing layer functions that work with Json files

    ChangeLog: (Who, When, What)
    SChintala,11.25.2024,Created Class
    """
    def read_data_from_file(file_name:str,student_data:list):  1 usage
        """ This function reads data from a JSON file and loads it into a list of dictionary rows

        ChangeLog: (Who, When, What)
        SChintala,11.25.2024,created function

        :return: list
        """
```

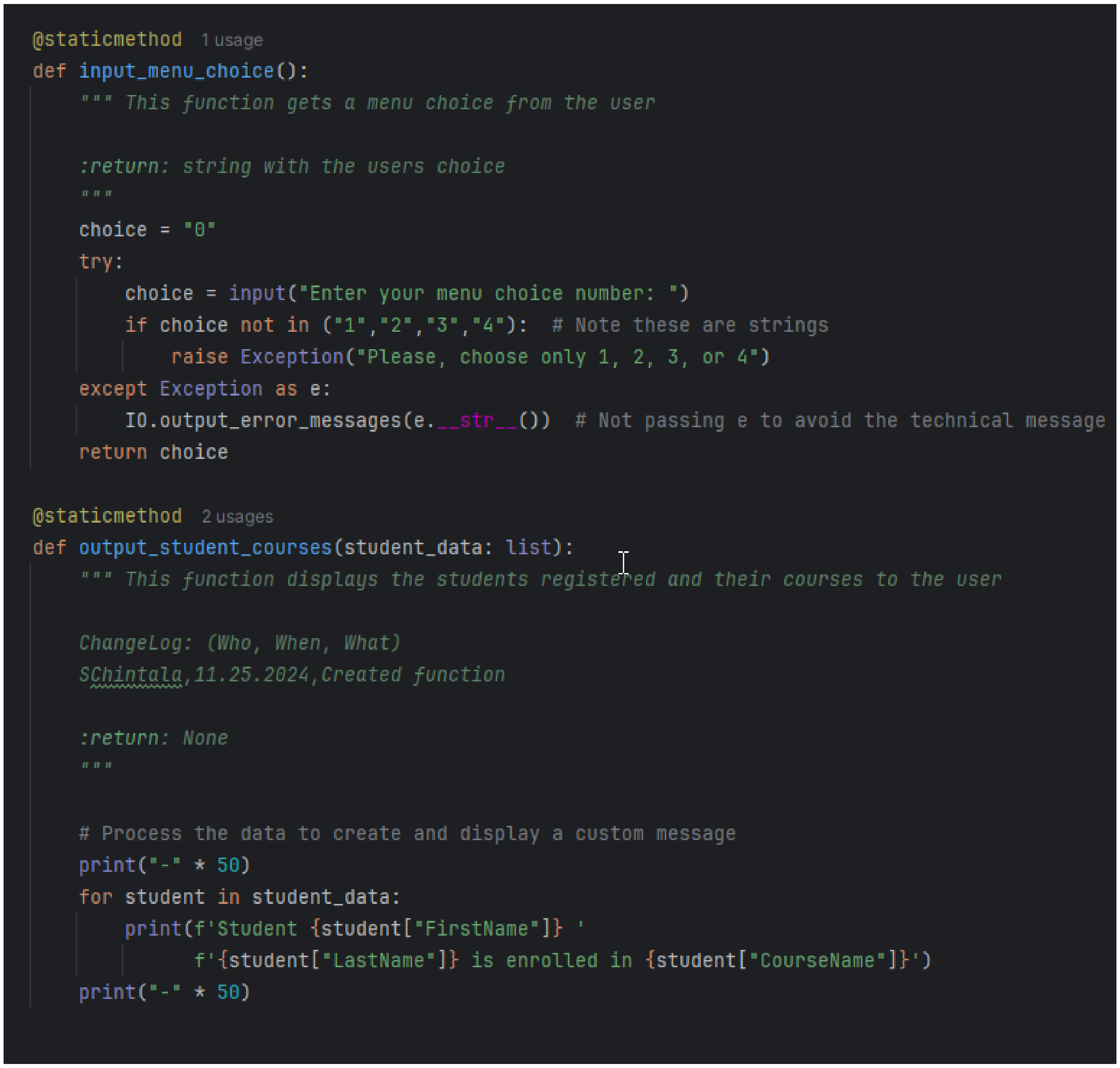
IO Class

The purpose of an IO class is to separate the concerns of input and output from the core logic of the application. This makes the code cleaner, more modular, and easier to test.

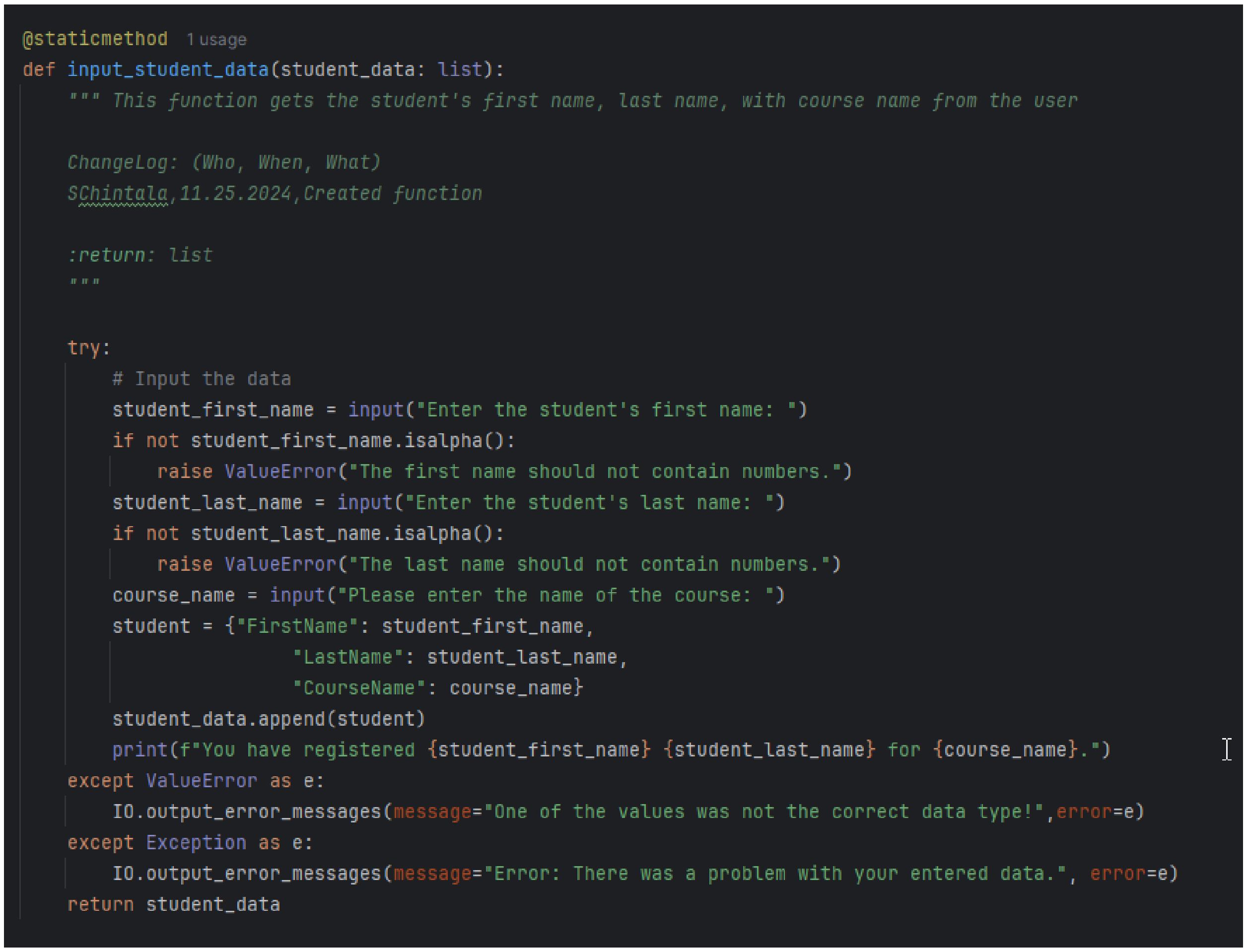
Figures 1.4-1.7 IO Class ***methods of user interaction***



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

    ChangeLog: (Who, When, What)
    SChintala,11.25.2024,Created Class

    """
    @staticmethod  5 usages
    def output_error_messages(message: str, error: Exception = None):
        """ This function displays a custom error messages to the user

        ChangeLog: (Who, When, What)
        SChintala,11.25.2024, 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 a menu of choices to the user

        ChangeLog: (Who, When, What)
        SChintala,11.25.2024,Created function

        :return: None
        """
        print()  # Adding extra space to make it look nicer.
        print(menu)
        print()  # Adding extra space to make it look nicer.
```



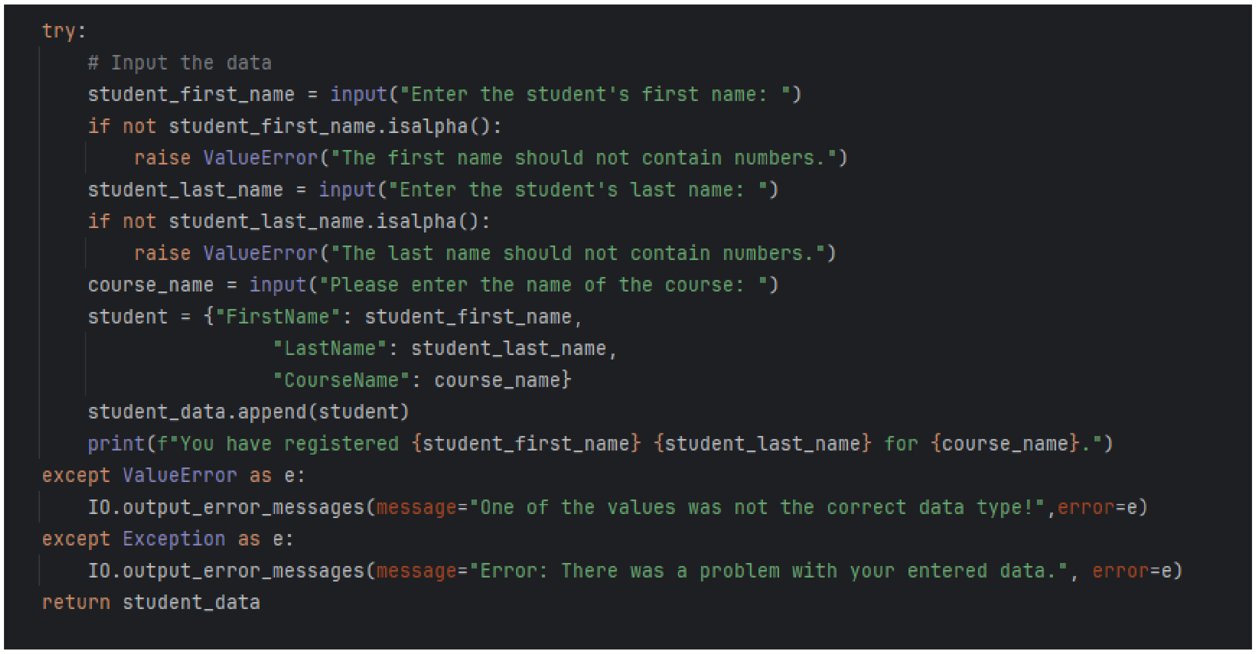
```
@staticmethod  1 usage
def input_menu_choice():
    """ This function gets a menu choice from the user

    :return: string with the users choice
    """
    choice = "0"
    try:
        choice = input("Enter your menu choice number: ")
        if choice not in ("1","2","3","4"):  # Note these are strings
            raise Exception("Please, choose only 1, 2, 3, or 4")
    except Exception as e:
        IO.output_error_messages(e.__str__())  # Not passing e to avoid the technical message
    return choice

@staticmethod  2 usages
def output_student_courses(student_data: list):
    """ This function displays the students registered and their courses to the user

    ChangeLog: (Who, When, What)
    SChintala,11.25.2024,Created function

    :return: None
    """

    # Process the data to create and display a custom message
    print("-" * 50)
    for student in student_data:
        print(f'Student {student["FirstName"]} '
              f'{student["LastName"]} is enrolled in {student["CourseName"]}')
    print("-" * 50)
```



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

    ChangeLog: (Who, When, What)
    SChintala,11.25.2024,Created function

    :return: list
    """

    try:
        # Input the data
        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: ")
        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="One of the values was not the correct data type!",error=e)
    except Exception as e:
        IO.output_error_messages(message="Error: There was a problem with your entered data.", error=e)
    return student_data
```

Error handling

In this module we learned how to handle the errors using the try-except blocks. Please see Figure 1.3, we will create a new file in the enrollments.json doesn't exist and to move further we continue the program by not stopping it. Also, we will validate the student inputs to be alphabetic and will display a message requesting user to enter only alphabets using the str. Isalpha().



```python
try:
    # Input the data
    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: ")
    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="One of the values was not the correct data type!",error=e)
except Exception as e:
    IO.output_error_messages(message="Error: There was a problem with your entered data.", error=e)
return student_data
```

Figure1.8 *error handling*

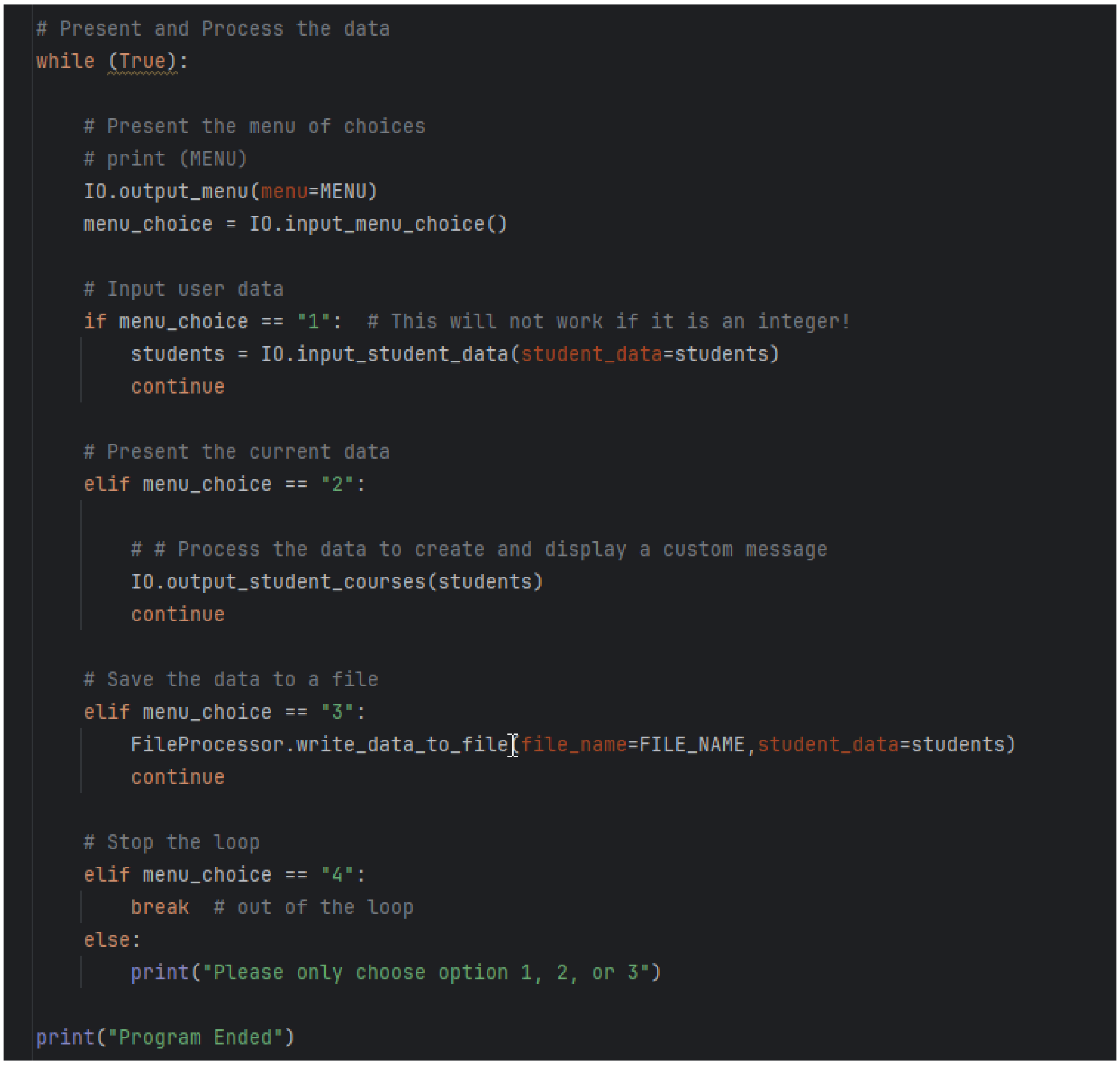
The program also shows how to collect input from students and double checks for errors before writing to the file. We will ensure only data is accepted for students by first name and last name as alphabetical and not numeric. Used a while true loop and it executes the program repeatedly till the condition is met with the menu choices. We use input prompts enabling users to enter the information from menu choice 1-4 till we use option 4 to break the program.

1. 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 variables (student's first name and last name, followed by the course name). The print function will display the student registration details.
2. On menu choice 2, the presents a coma-separated string by formatting the collected data using the print () function. We are using a loop for getting student data from students multidimensional list for first name, last name, and course name, and print the data.
3. On menu choice 3, program opens the file named "Enrollments.json" in write mode using the open () function. It writes the content of the json_data variable to the file using the write() function, then file is closed using the close() method. Then displays what was stored in the file. Using \n we will write the data in the new line instead of deleting the existing. Similar to the above step, we will use for loop for write the student details to the file for multiple students.

4. On the menu, 4, program prints the message that we are ending the program and stops the while loop and ends the program.

We are using elif or else if statements for menu choices, when choice 1 is not met, elif statement for choice 2 becomes true and prints the data, and same with choice 3, 4. When none of the conditions are met which are not 1-4 choice options, it executes the code in 'else' block and takes action to exit the program.

Figure 1.9: *shows input and output () functions, menu choices 1-4*



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

    # Present the menu of choices
    # print (MENU)
    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!
        students = IO.input_student_data(student_data=students)
        continue

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

        # # Process the data to create and display a custom message
        IO.output_student_courses(students)
        continue

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

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

print("Program Ended")
```

Testing the program

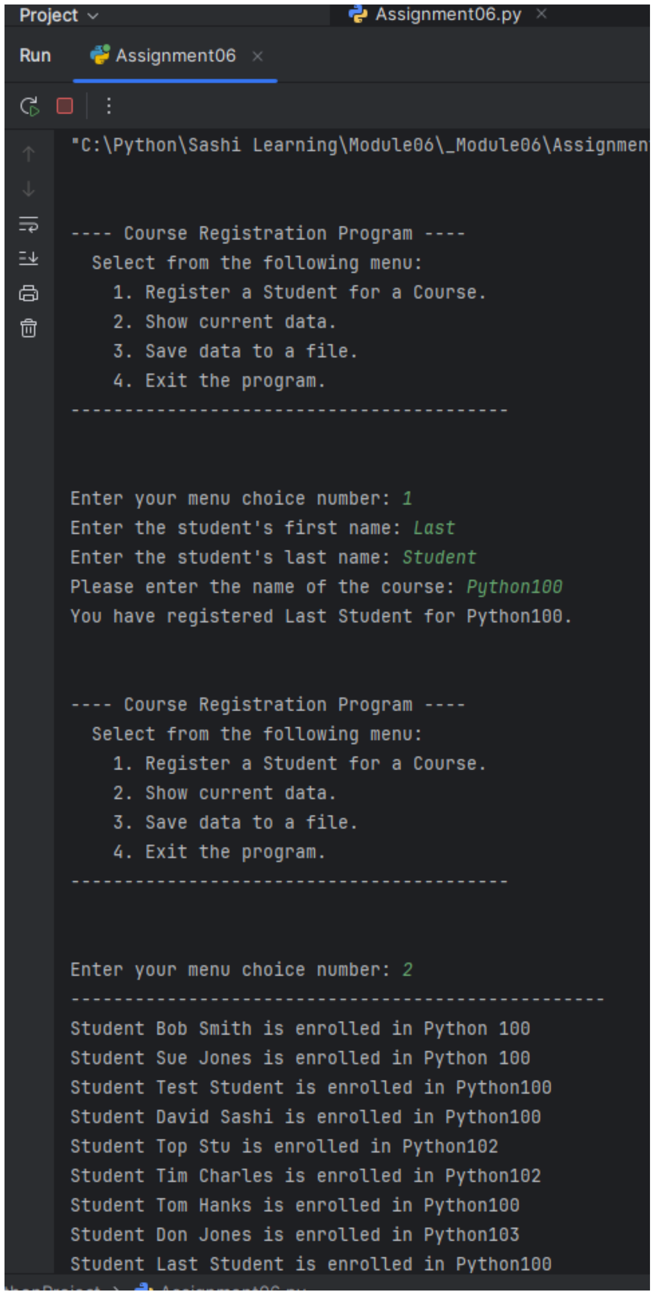
- Please see the figures 1.5 and 1.6, by selecting the menu choice 1, our program takes the user's input for a student's first, last name, and course name and updates to dictionaries instead of lists and ensure the keys or variables are consistent/
- By selecting the menu choice 2, our program displays the user's input for a student's first, last name, and course name.
- By selecting the menu choice 3, our program saves the user's input for a student's first, last name, and course name to the json file to reflect the keys
- By selecting the menu choice 4, our program exists.

- Our program allows us to register or display or save multiple student registrations with first name, last name, and course name as shown in the results screenshots.
- We can run in PyCharm and from the command prompt console

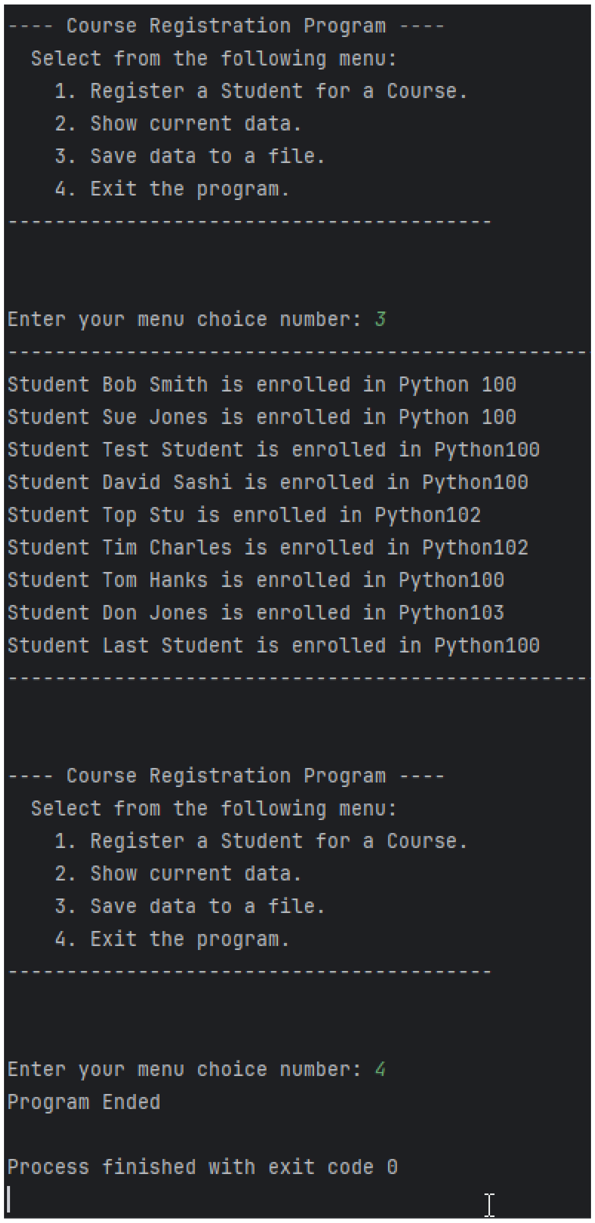
Result from PyCharm

Results from PyCharm IDE for menu choice 1-4 from the two screenshots, and json output.

***Figures 2.1 -2.2** – menu choice 1 -4, registers students, displays, writes, and exits*

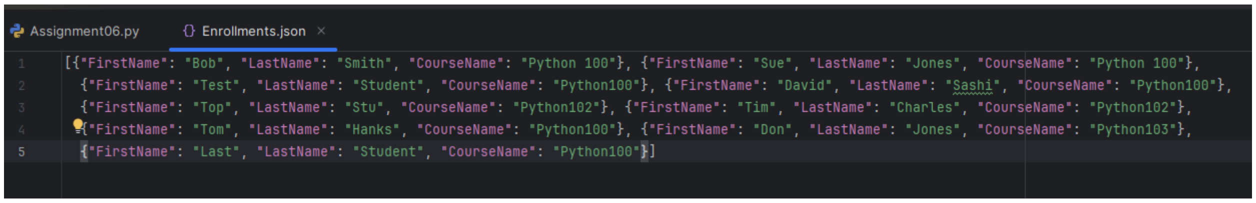


```
"C:\Python\Sashi Learning\Module06\_Module06\Assignmen


---- 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: Last
Enter the student's last name: Student
Please enter the name of the course: Python100
You have registered Last Student for Python100.


---- 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
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Test Student is enrolled in Python100
Student David Sashi is enrolled in Python100
Student Top Stu is enrolled in Python102
Student Tim Charles is enrolled in Python102
Student Tom Hanks is enrolled in Python100
Student Don Jones is enrolled in Python103
Student Last Student is enrolled in Python100
```



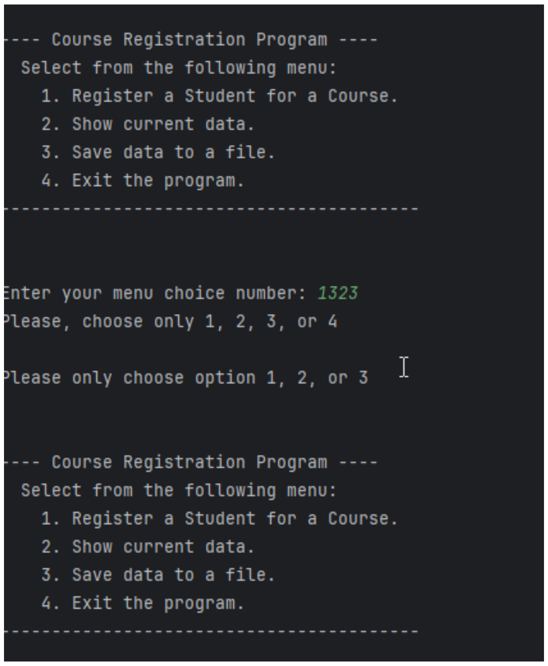
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


Enter your menu choice number: 3
-------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Test Student is enrolled in Python100
Student David Sashi is enrolled in Python100
Student Top Stu is enrolled in Python102
Student Tim Charles is enrolled in Python102
Student Tom Hanks is enrolled in Python100
Student Don Jones is enrolled in Python103
Student Last Student is enrolled in Python100
-------------------------------------------------


---- 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
```

Figure 2.5 shows the enrollments.json file data

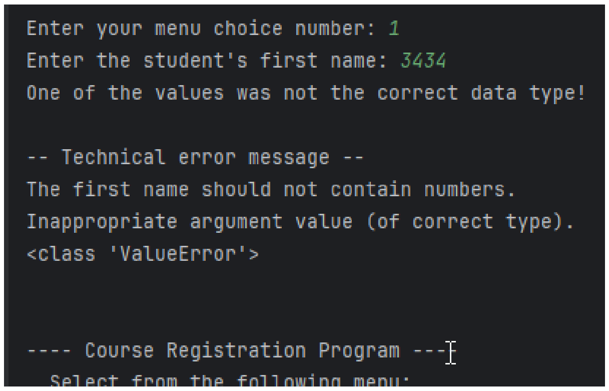


```
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"}, {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"},
  {"FirstName": "Test", "LastName": "Student", "CourseName": "Python100"}, {"FirstName": "David", "LastName": "Sashi", "CourseName": "Python100"},
  {"FirstName": "Top", "LastName": "Stu", "CourseName": "Python102"}, {"FirstName": "Tim", "LastName": "Charles", "CourseName": "Python102"},
  {"FirstName": "Tom", "LastName": "Hanks", "CourseName": "Python100"}, {"FirstName": "Don", "LastName": "Jones", "CourseName": "Python103"},
  {"FirstName": "Last", "LastName": "Student", "CourseName": "Python100"}]
```

Figiure 2.6 shows the message when user enters numeric, instead of stopping the program we will give an opportunity to the user to enter again. This is the new thing I learned from this module.



```
---- 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: 1323
Please, choose only 1, 2, 3, or 4

Please only choose option 1, 2, or 3


---- 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: 3434
One of the values was not the correct data type!

-- Technical error message --
The first name should not contain numbers.
Inappropriate argument value (of correct type).
<class 'ValueError'>


---- Course Registration Program ---
  Select from the following menu:
```

Figure 2.7 File not found error helps when a file doesn't exist, but it creates a new file to continue with the program.

Testing the program from Command prompt on Windows

I was able to test the same from command prompt.

Summary:

I enjoyed learning the differences between the class and functions. I am grateful for the notes and videos shared by Professor Root. Modules helped me to understand the use of separation of concerns and how to make the code readable for other members in the team or for future references. Still trying to do more examples from the labs.