Richard Khan
11/20/2023
Foundation of Programming: Python
Assignment06 (late submittal)

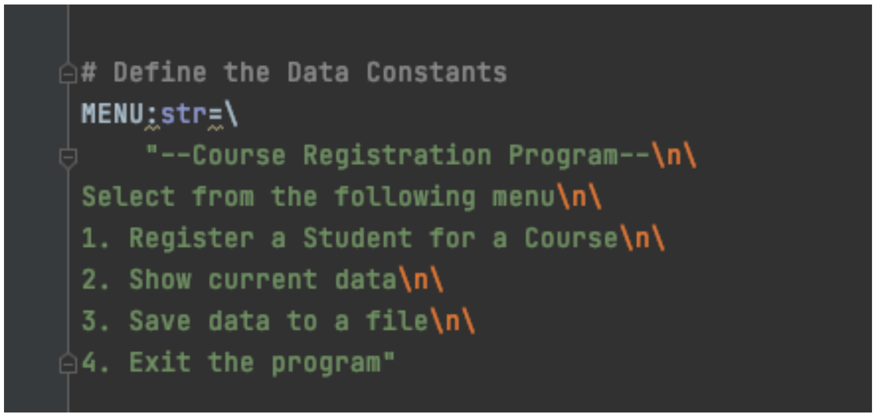
Creating Python Scripts

Introduction

Assignment 6 introduced new terminology and tools for python programming. This assignment and script is similar to previous assignment but now adds the use of functions, classes and separation of concerns pattern.

Assignment

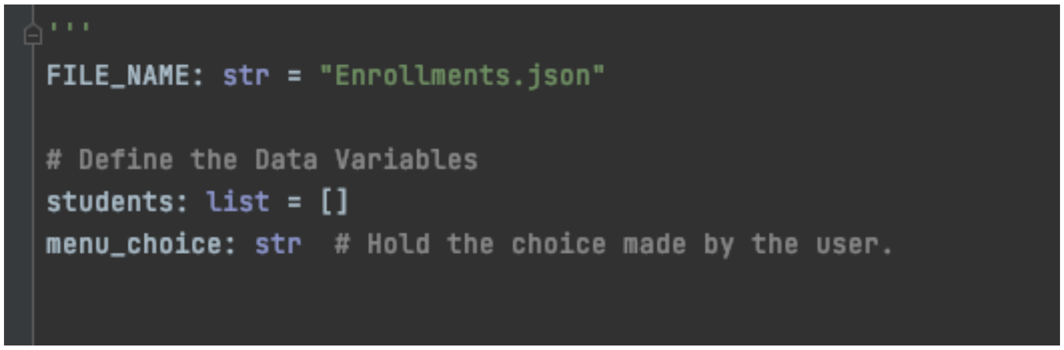
This week's assignment was similar to assignment05. Previously, I demonstrated using constants, variables and print statements to display messages while using while loop, and conditional logic. I followed the same criteria of adding the constants and variables. The constants were defined as **MENU:str**. The print (MENU) was moved to "present the menu of choice" in order to display the menu for each option using while loop.



```
# Define the Data Constants
MENU:str=\
    "--Course Registration Program--\n\
Select from the following menu\n\
1. Register a Student for a Course\n\
2. Show current data\n\
3. Save data to a file\n\
4. Exit the program"
```

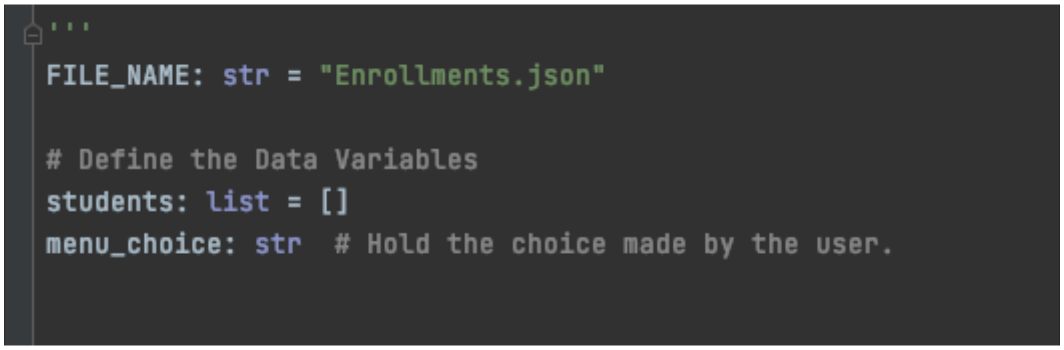
Figure 1: Defining Constants

After defining the constants (same as assignment05), I defined the variables as shown in



```
'''
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables
students: list = []
menu_choice: str  # Hold the choice made by the user.
```

Figure *2***.** However, the variables list has shorten and moved to a particular function of the script. This assignment introduced local and global variables, where global variables are declared outside the function and considered for the entire script. Local variables are for a section of that script.



```
'''
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables
students: list = []
menu_choice: str  # Hold the choice made by the user.
```

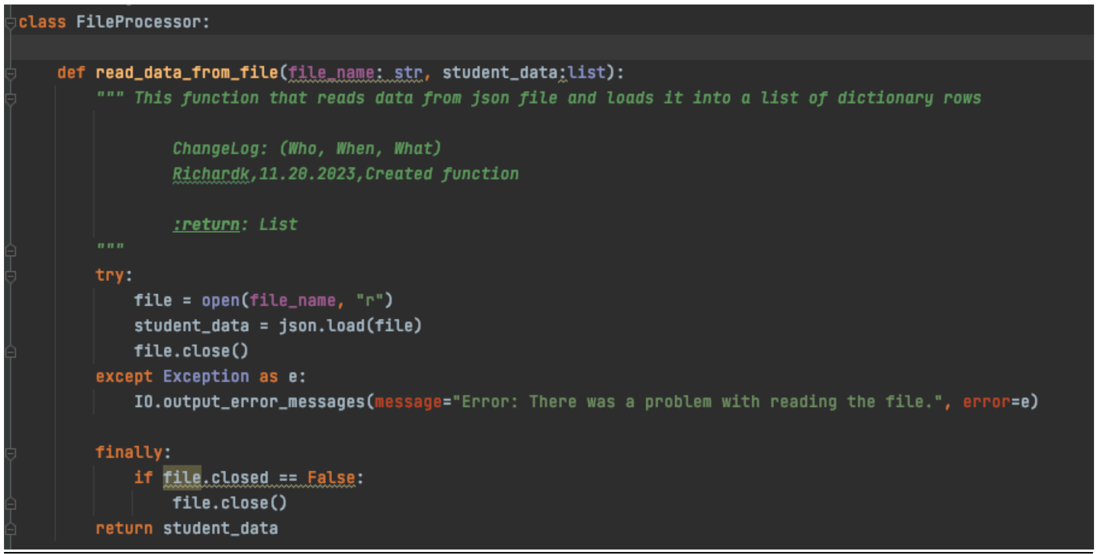
Figure 2: Defining Variables

Separation of Concerns was introduced to help our script be organized and be able to read clearer. This pattern promotes modularity, reduces code complexity and makes it easier to manage and extend software systems. There are several layers in organizing your code based on patterns and the first layer used in this script is the processing layer. This layer is used for implementing the core functionality and business rules of the application. It is the layer that process and transform data as required by the application.

Processing layer:

After declaring the constants and variables, the Class FileProcessor was created to organize the code that will be the foundation for this assignment. Under FileProcessor, code was written or borrowed from the class lab to read data from a file and write data to a file. This is the same from the last two assignment, reading data from the Json file and being able to write to the Json file.

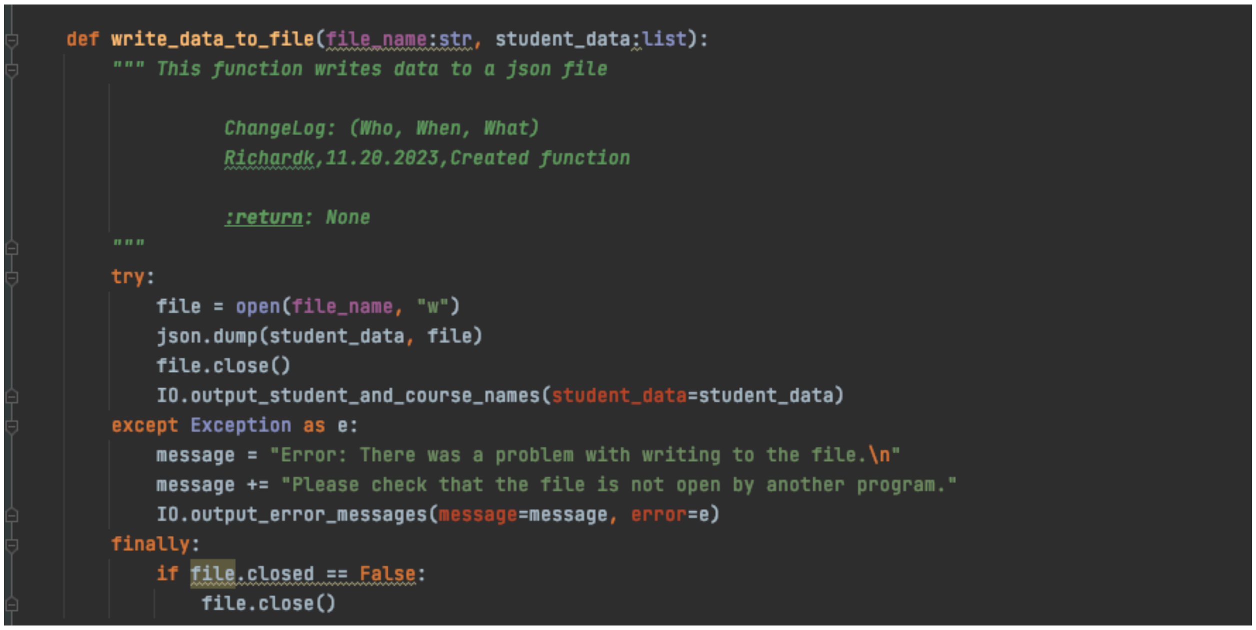
Figure 3 shows the code for reading data from a file:



```python
class FileProcessor:

    def read_data_from_file(file_name: str, student_data:list):
        """ This function that reads data from json file and loads it into a list of dictionary rows

                ChangeLog: (Who, When, What)
                Richardk,11.20.2023,Created function

                :return: List
        """
        try:
            file = open(file_name, "r")
            student_data = json.load(file)
            file.close()
        except Exception as e:
            IO.output_error_messages(message="Error: There was a problem with reading the file.", error=e)

        finally:
            if file.closed == False:
                file.close()
        return student_data
```

Figure 3: read data from a file (json)

Figure 4 shows the code for writing data to a file:

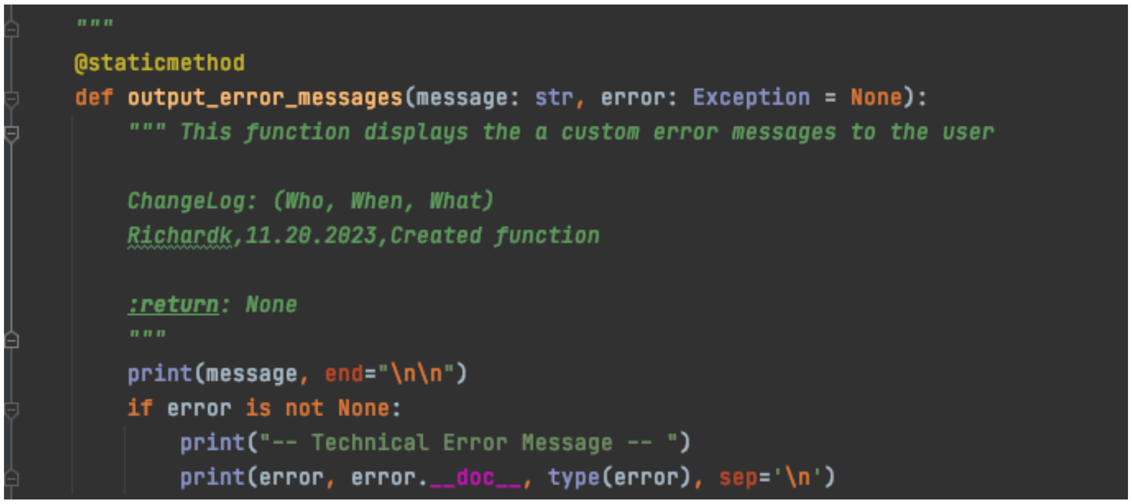


```python
    def write_data_to_file(file_name:str, student_data:list):
        """ This function writes data to a json file

                ChangeLog: (Who, When, What)
                Richardk,11.20.2023,Created function

                :return: None
        """
        try:
            file = open(file_name, "w")
            json.dump(student_data, file)
            file.close()
            IO.output_student_and_course_names(student_data=student_data)
        except Exception as e:
            message = "Error: There was a problem with writing to the file.\n"
            message += "Please check that the file is not open by another program."
            IO.output_error_messages(message=message, error=e)
        finally:
            if file.closed == False:
                file.close()
```

Figure 4: write data to a file (json)

<u>Presentation Layer:</u>
This layer in separation of concern, also known as the Input/output layer, focuses on presenting information to users and capturing user input.

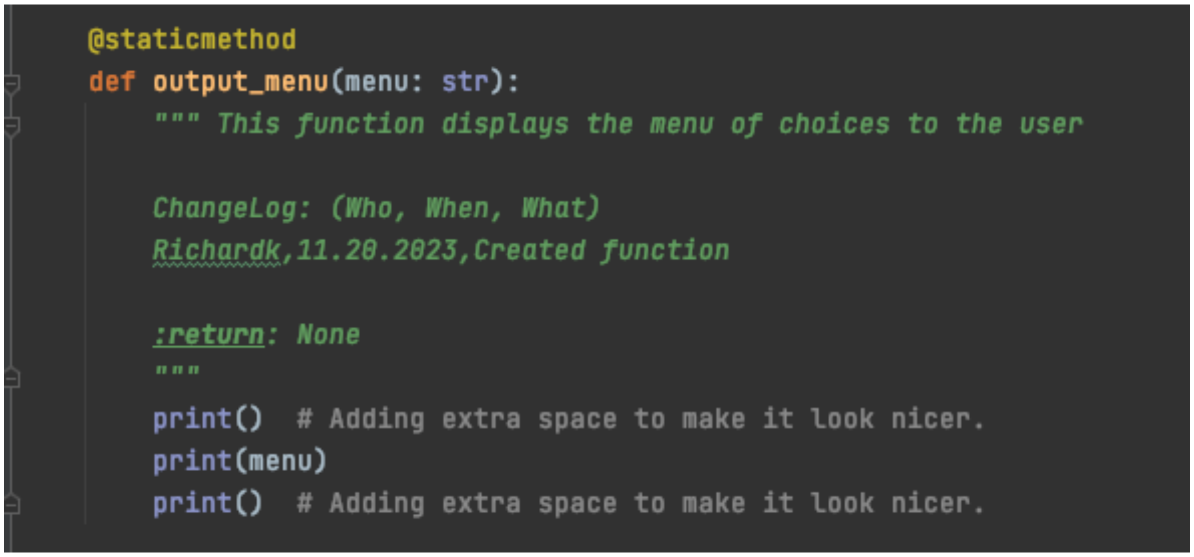
Figure 5 shows the first function, from the assignment instructions, output_error_message (message:str, error:Exception = None). This function displays an error message to the user and returns nothing.



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

        ChangeLog: (Who, When, What)
        Richardk,11.20.2023,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 5: Error message

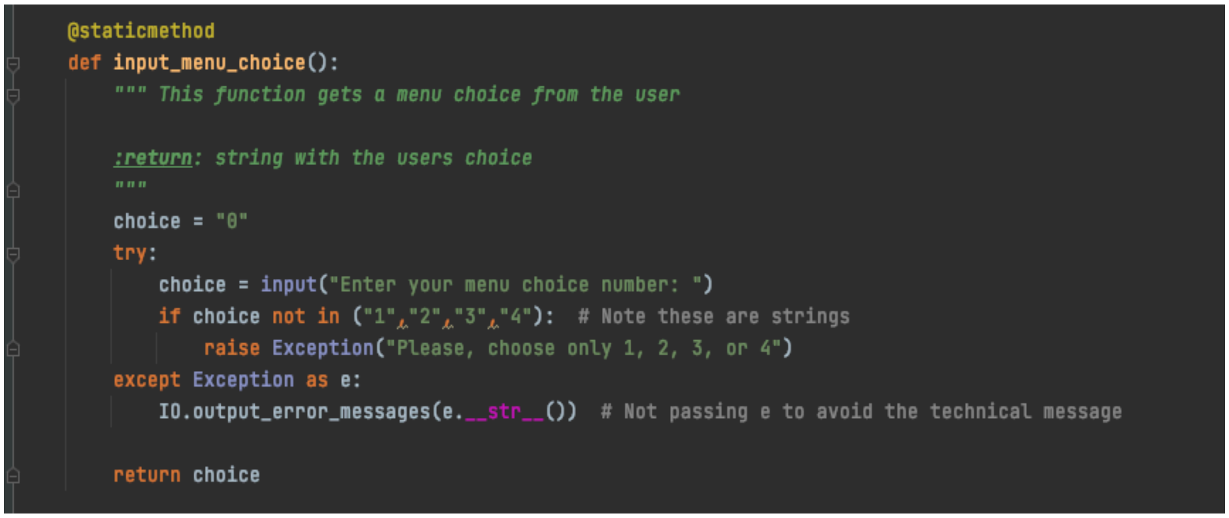
Figure 6 shows the second function, from the assignment instructions, output_menu (menu:str). This function displays the menu options for the user. The menu is shown in Figure *1*.



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

        ChangeLog: (Who, When, What)
        Richardk,11.20.2023,Created function

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

Figure 6: Menu Options

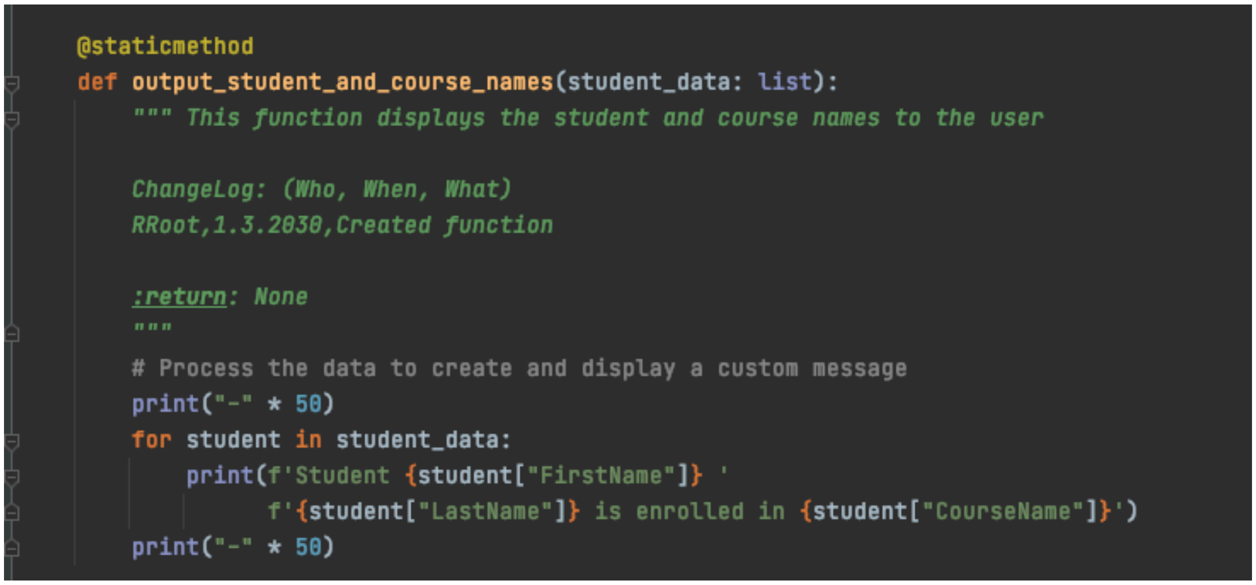
Figure *7* shows the third function, from the assignment instructions, input_menu_choice. This function gets the menu choice from the user. There is an error message and exception if one of the choices is not chosen and there has to be a return.



```python
@staticmethod
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
```

Figure 7: Input menu choice

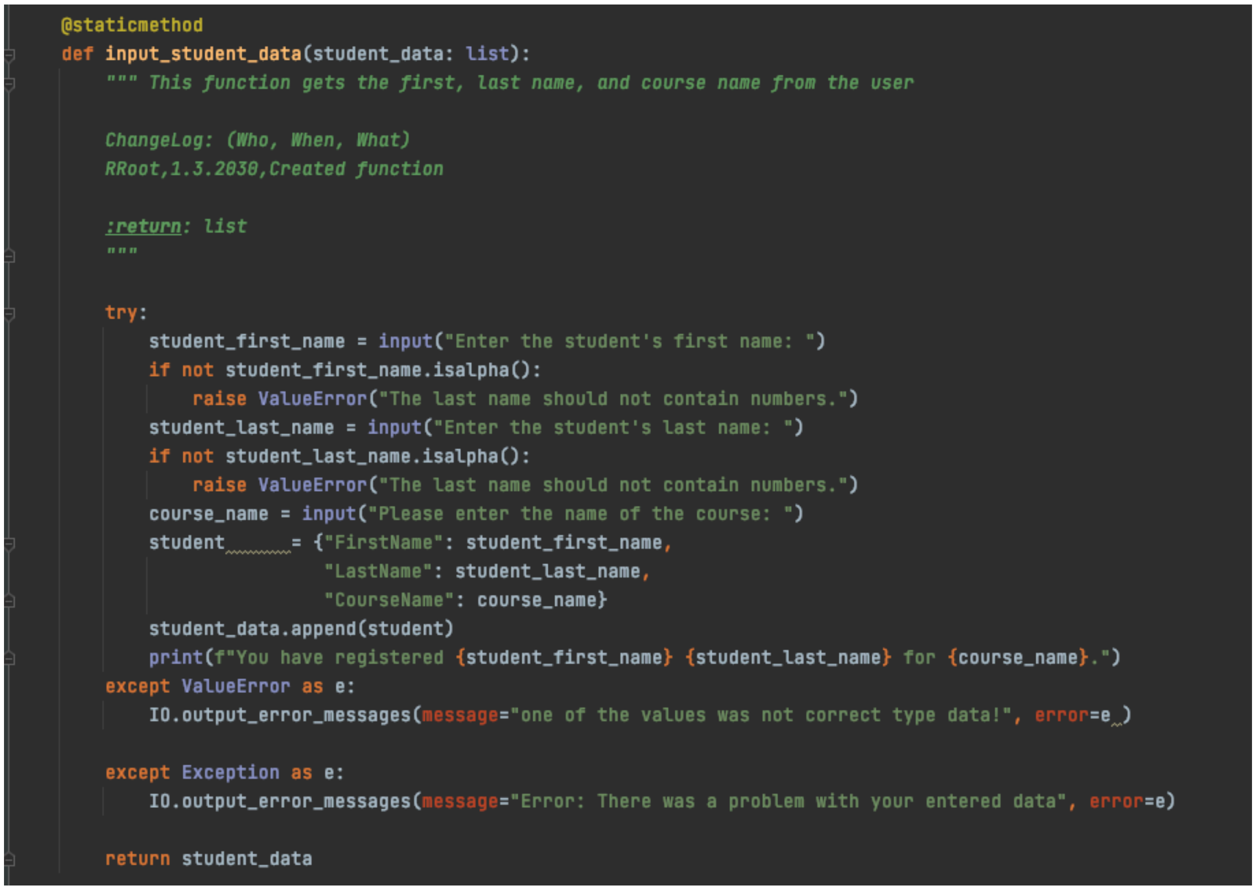
Figure 8 shows the fourth function, from the assignment instructions, output_student_and_course_names. This function displays the student and course names to the user after selecting the menu option from figure 7. Once selected, the data will show the student data (names and course name) as a list.



```python
@staticmethod
def output_student_and_course_names(student_data: list):
    """ This function displays the student and course names to the user

    ChangeLog: (Who, When, What)
    RRoot,1.3.2030,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)
```

Figure 8: Output student and course names

Figure 9 shows the fifth function, from the assignment instructions, input_student_data. This function continues after the input_menu function from figure 7. Once a menu choice is chosen, in this case menu 1, which is to register a student, you input the student first and last name and the course name they are registering to. The print menu prints out the data. The exception and error shows what will occur if incorrect data was incorporated.

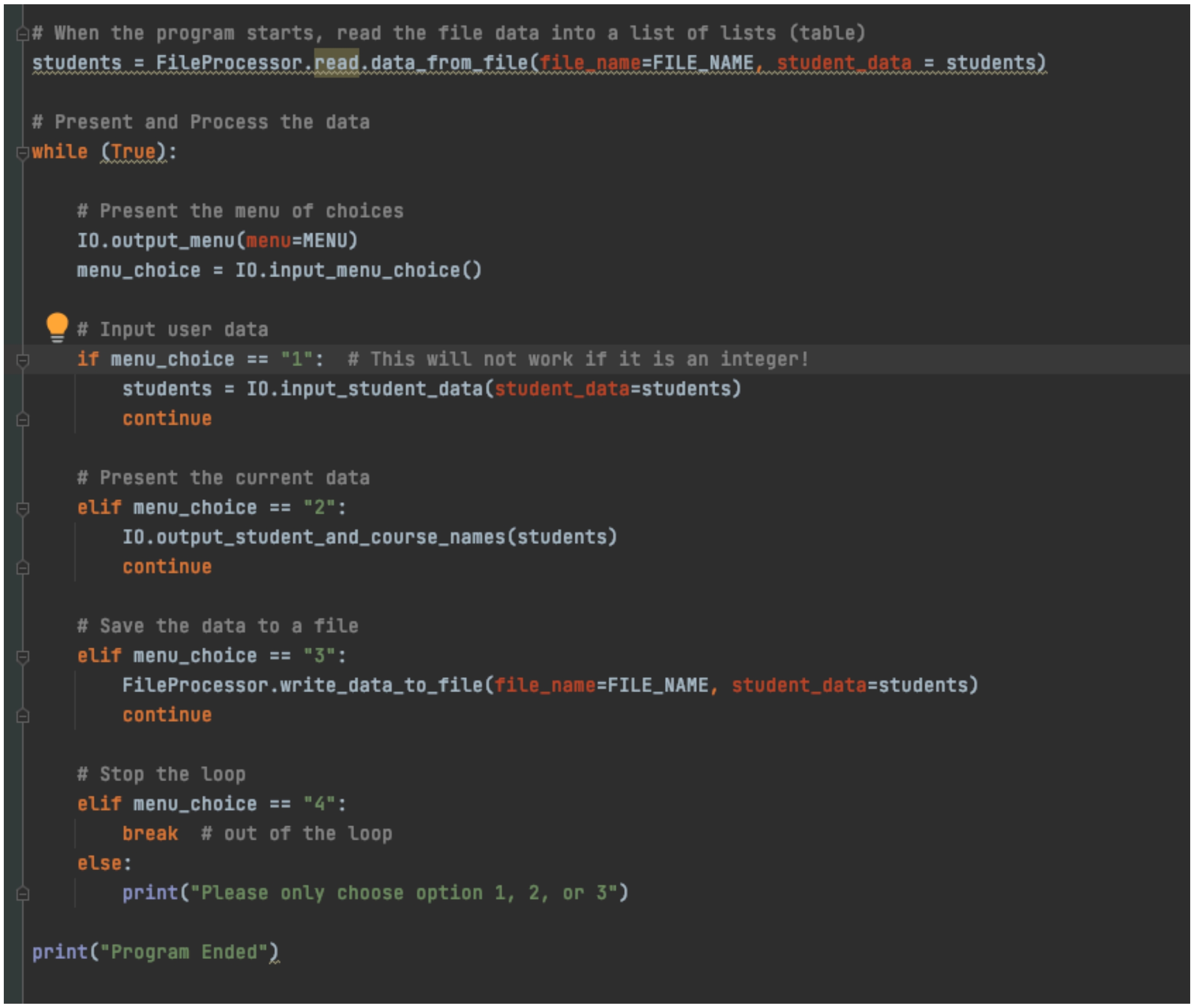


```python
@staticmethod
def input_student_data(student_data: list):
    """ This function gets the first, last name, and course name from the user

    ChangeLog: (Who, When, What)
    RRoot,1.3.2030,Created function

    :return: list
    """

    try:
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("The last 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 correct type data!", 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
```

Figure 9: Input student data

Once the presentation layer was complete with all of the instructions, the next stage of the separation of concerns was processing the data.

- First read data from the Json file
- Select option 1 from the menu choice to incorporate the student first and last name and course name.
- Option 2 is to present the data in a list format after completing option 1.
- Option 3 is to save the data to the Json file
- Option 4 is to exit the loop.

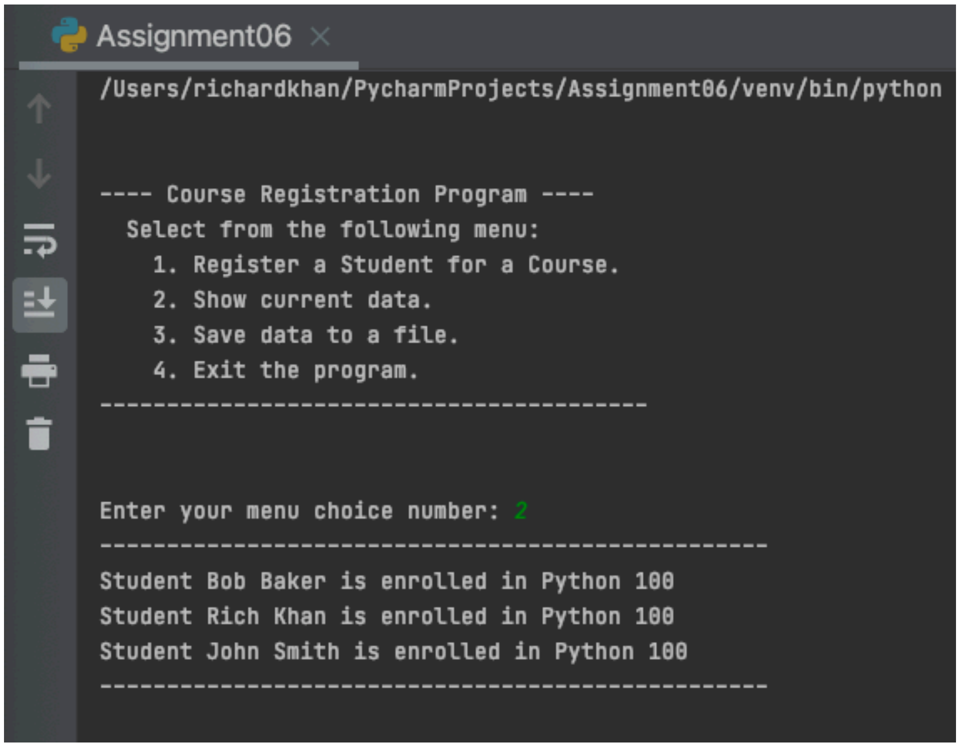


```python
# When the program starts, read the file data into a list of lists (table)
students = FileProcessor.read.data_from_file(file_name=FILE_NAME, student_data = students)

# Present and Process the data
while (True):

    # Present the menu of choices
    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":
        IO.output_student_and_course_names(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")
```

Figure 10: Start of the Main Body

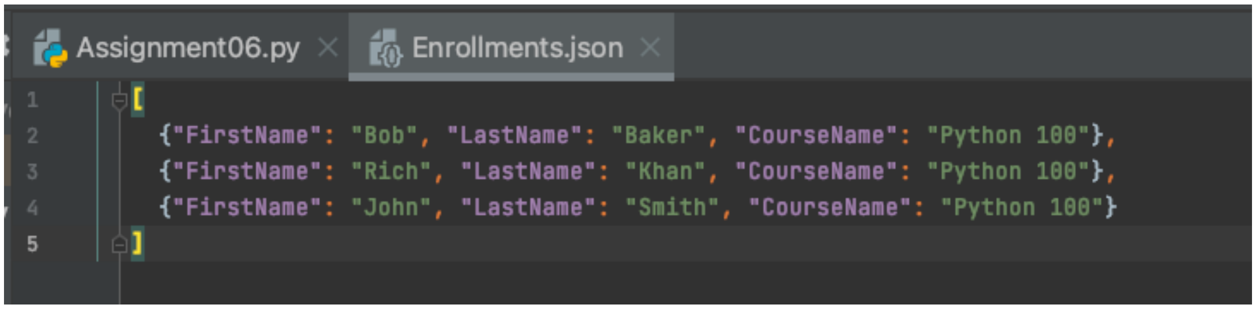
Executing the program

Read the data from the Json file:



Assignment06

```
/Users/richardkhan/PycharmProjects/Assignment06/venv/bin/python


---- 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 Baker is enrolled in Python 100
Student Rich Khan is enrolled in Python 100
Student John Smith is enrolled in Python 100
--------------------------------------------------
```

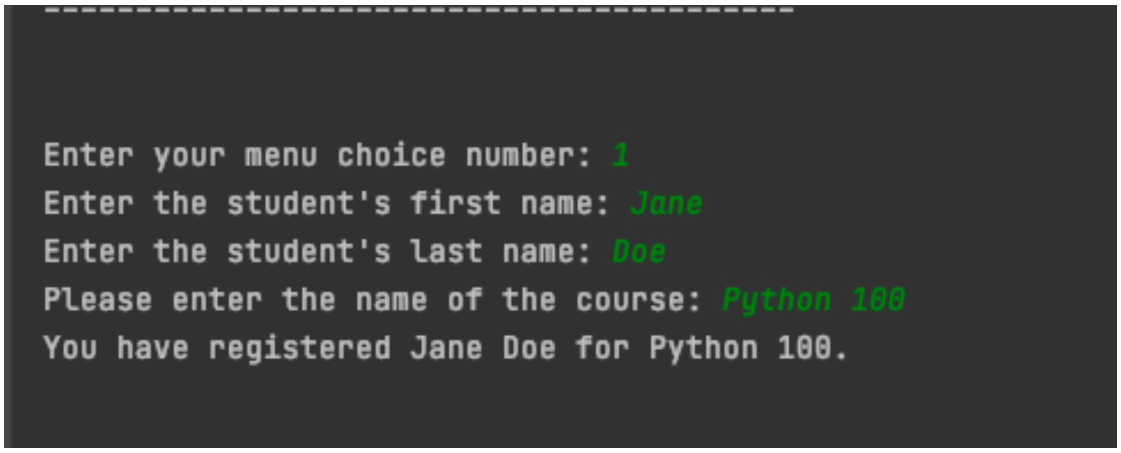


Assignment06.py   Enrollments.json

```
[
  {"FirstName": "Bob", "LastName": "Baker", "CourseName": "Python 100"},
  {"FirstName": "Rich", "LastName": "Khan", "CourseName": "Python 100"},
  {"FirstName": "John", "LastName": "Smith", "CourseName": "Python 100"}
]
```

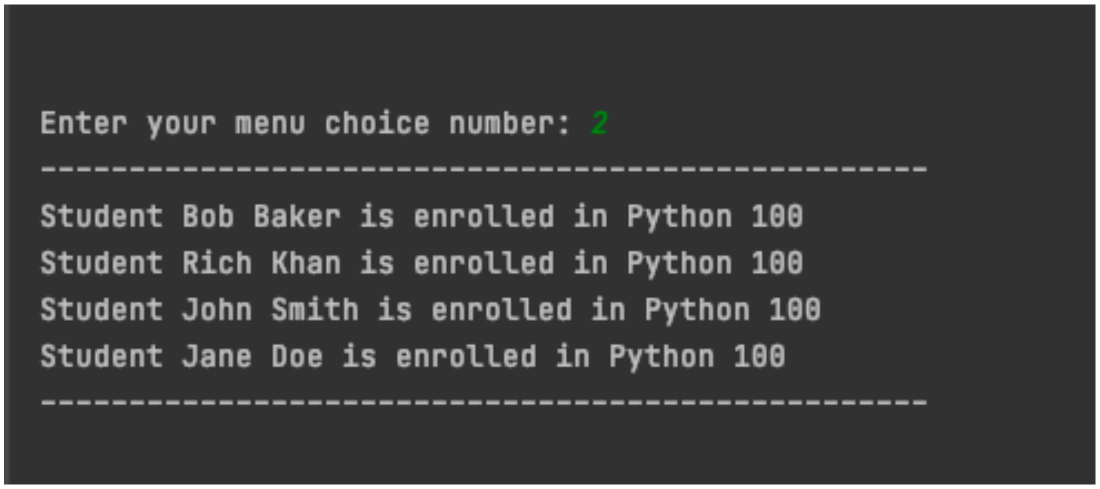
Enrollments.json

Select option 1 from the menu choice to incorporate the student first and last name and course name.



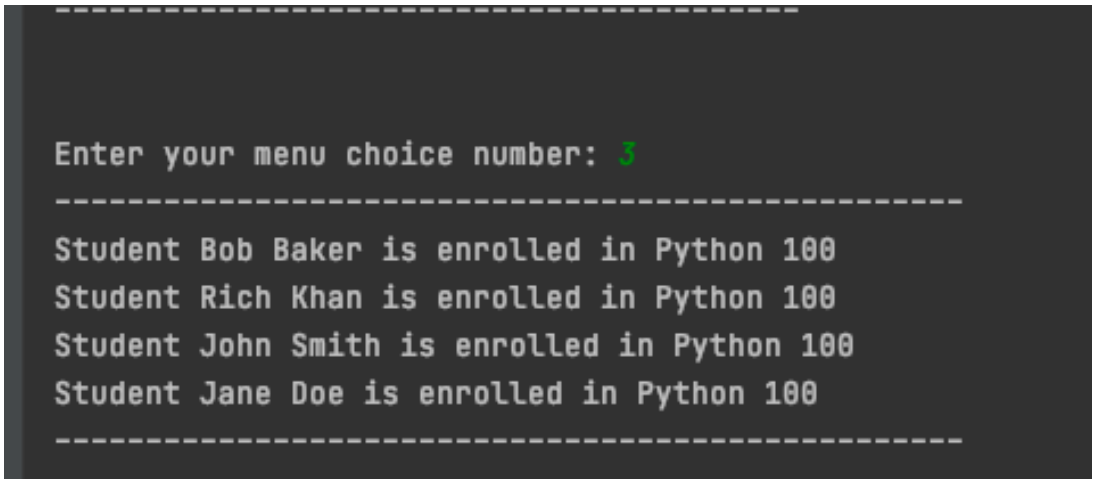
```
--------------------------------------------------


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

When the “enter your menu choice number” shows again, I inputted 2, in order to present the data.

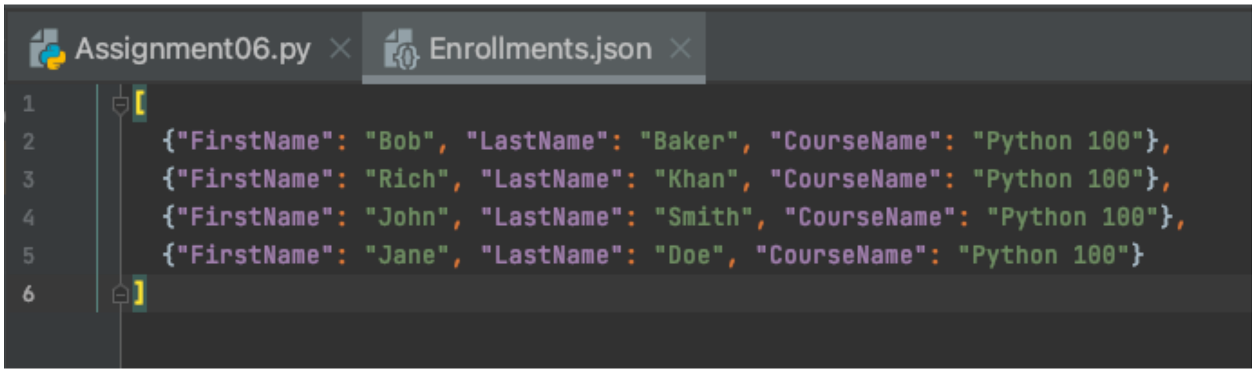


```
Enter your menu choice number: 2
--------------------------------------------------
Student Bob Baker is enrolled in Python 100
Student Rich Khan is enrolled in Python 100
Student John Smith is enrolled in Python 100
Student Jane Doe is enrolled in Python 100
--------------------------------------------------
```

When the "enter your menu choice number" shows again, I inputted 3, in order to save the data to enrollments.json.



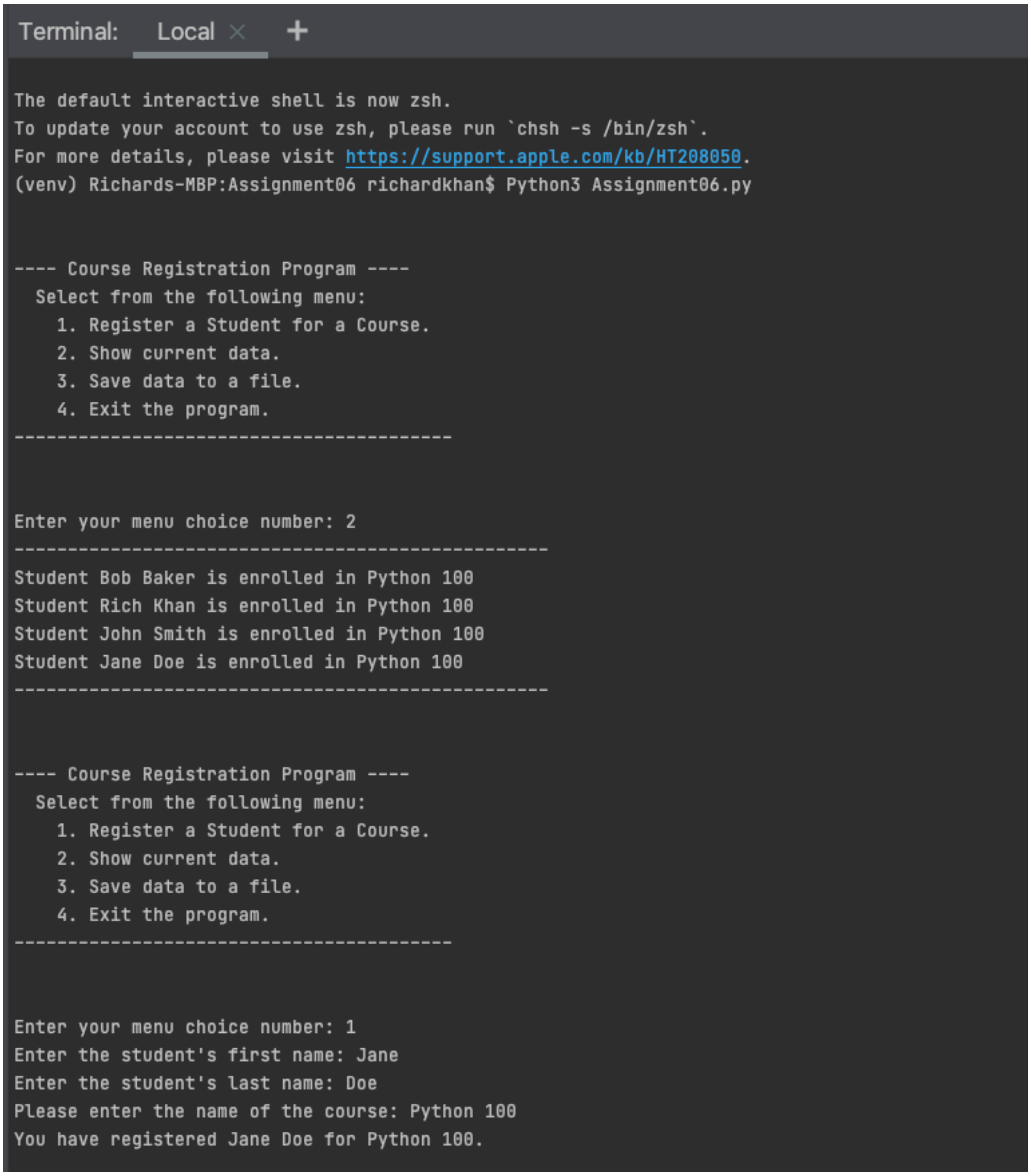
```
-----------------------------------------



Enter your menu choice number: 3
--------------------------------------------------
Student Bob Baker is enrolled in Python 100
Student Rich Khan is enrolled in Python 100
Student John Smith is enrolled in Python 100
Student Jane Doe is enrolled in Python 100
--------------------------------------------------
```



```
[
  {"FirstName": "Bob", "LastName": "Baker", "CourseName": "Python 100"},
  {"FirstName": "Rich", "LastName": "Khan", "CourseName": "Python 100"},
  {"FirstName": "John", "LastName": "Smith", "CourseName": "Python 100"},
  {"FirstName": "Jane", "LastName": "Doe", "CourseName": "Python 100"}
]
```

New Enrollments.json

Execution of script in Terminal



```
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(venv) Richards-MBP:Assignment06 richardkhan$ Python3 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: 2
--------------------------------------------------
Student Bob Baker is enrolled in Python 100
Student Rich Khan is enrolled in Python 100
Student John Smith is enrolled in Python 100
Student Jane Doe 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.
-----------------------------------------


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

Conclusion
Organizing one's code is important as codes and assignment becomes more complicated and larger. This assignment we learned three techniques for improving our scripts, Functions, Classes, and Separation of Concerns. Functions allows you to break down a larger program into smaller and more manageable pieces. Each function can focus on a specific task making the code easier to understand and read. Classes are very similar, but in this case, it groups functions together based on what is being accomplished. The last of the three, separation of concerns is separating the script

into layers: processing and presenting layer. This accomplished the same as functions and classes but what SOC does is group the input/output and the instruction of what the script is to accomplish. These three techniques are the way forward as scripts gets more complicated and we as students advance on to other python classes.