Harpreet Bassi

August 11, 2024

Foundations of Programming, Python

Assignment 07

GitHub Link- https://github.com/harpreetkbassi/IntroToProg-Python-Mod07

Classes and Objects

Intro

In assignment 07, I learned about the additional programming tools and techniques using PyCharm. In this document, I will explain the steps I took to create an interactive program similar to assignment 07, where I used my knowledge of how to create and use classes to manage data. And finally learned how to implement this data in a JavaScript Object Notation (JSON) file and GitHub.

Creating the Code

I started off by opening PyCharm IDE and Assignment07.py, edited the script header, and then began writing my code. As seen in Figure 1, first thing I started with was include the line that imports the "JSON" module, which is used for working with JSON data in Python. I then added the "MENU" line which is a multi-string that contains the menu options displayed to the user. I also included the "FILE_NAME" (Enrollments.json) is the name of the file where the data will be stored. I then included the variables which in this case I added "students" as a list that will store the data for the students, and the "menu_choice" and set it to a string value that holds the user's menu choice.

As seen in Figure 2-3, I then added the "FileProcessor" class provided in the code Is a utility class designed to handle the reading and writing of the student data to and from the JSON file. The purpose of adding the "FileProcessor" class centralizes all the file-related operations, specifically reading from and writing to a JSON file. I then added the "read_data_from_file" which was added for the method of reading the student data from a specified JSON file and loads it into the "student_data" list which is a list of dictionaries that represented a student. I then added the parameters which was "file_name" which is the name of the file to read from, and the "student_data" which is the list where the read data will be stored. I then added the error handling part in the code as shown in Figure 4, which included the "Try Block" that attempts to open the file and load the JSON data into the list, then added the "Except Block" which is if there is an error that occurs, it looks at the exception and uses the 'IO' class to show a user-friendly error message. I then added the "Finally Block" which makes sure that the file is closed even if there was an error that occurs during the reading. Lastly, then added the "Return Value" which is the method that returns the 'student_data' list filled with data from the file.

As shown in Figure 5, I then added the “write_data_to_file” which was added for the current ‘student_data’ list to a specified JSON file, it creates the list of dictionaries into JSON format and saves it to the file. I then added the parameters which was “file_name” which is the name of the file to write to, and the “student_data” which is the list of dictionaries that represent each student that will be written to the file. I then added the error handling which was the same as “read_data_to_file”, and then added the additional action which was after it successfully writing to the file, it calls the ‘IO.output_student_and_course_name’ to display the updated student data to the user.

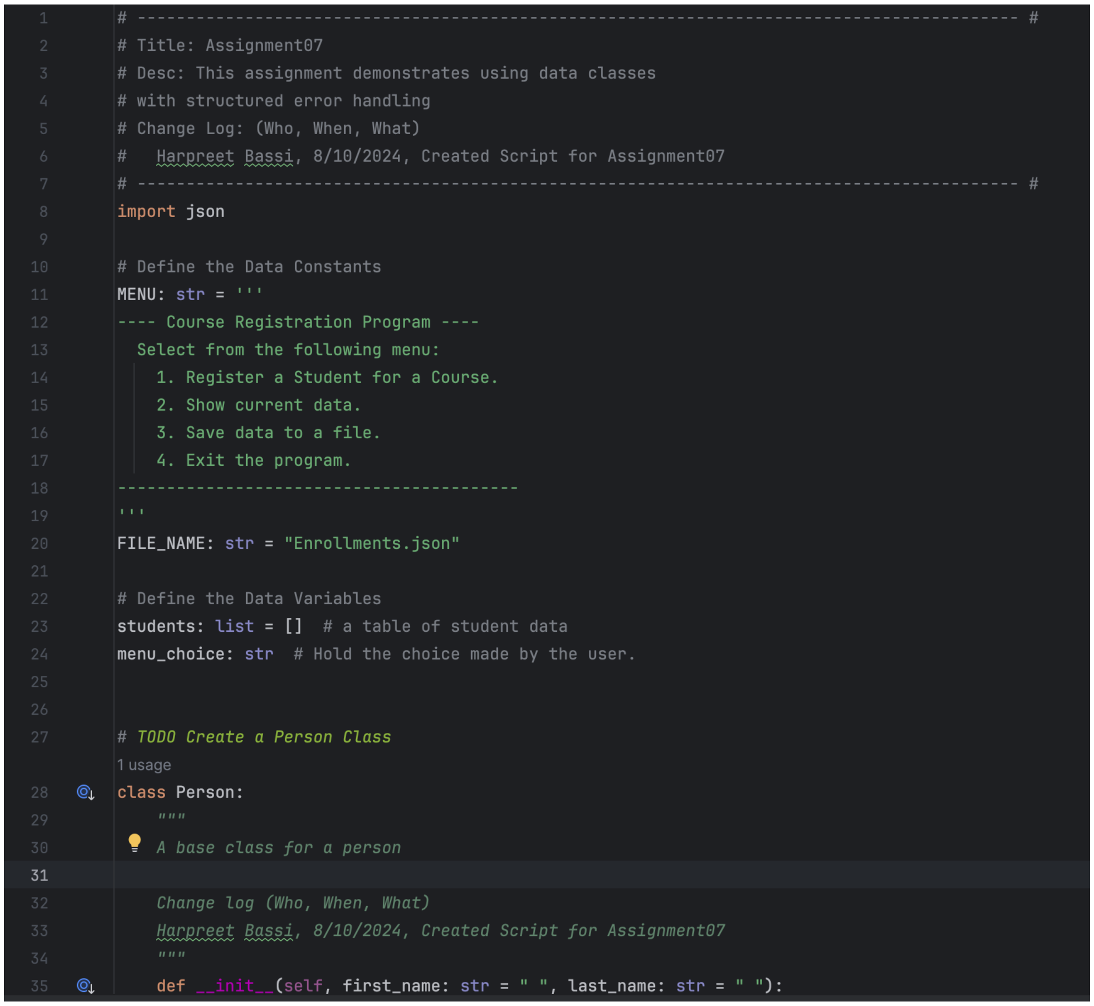
As seen in Figure 5-6, I then added the IO class which centralizes all the input and output operations, making it easier to manage and modify how the program interacts with the users. I then added the “output_error_messages” which was added for the method displays error messages to the user in a user-friendly way. I then added the ‘message’ which is a custom error message that you would want to display to the user and then the ‘error’ as an optional ‘Exception’ object that contain the technical error details. I used this method when an error occurs, providing a clear message to the user about what would have gone wrong, and optionally including the technical details for troubleshooting. I then added the “output_menu” which was added for this method that displays a menu to the user and the ‘menu’ parameter that is a string that contains the text of the menu to be displayed. I added this method when you need to show the user their options, such as registering a student, displaying the data, or saving data to a file.

Afterwards, I added the “input_menu_choice”. As shown in Figure 7, I added the input menu choice for it to collect the user’s menu choice and ensured it’s validated. In this code, the user would enter their choice, the method would check if their input was valid and if the choice is not valid, it then raises an exception and displays an error message using the ‘output_error_messages’. It then returns the users choice as a string. This method overall is called after displaying the menu to get the user’s input on what action they would want to take.

I then added the “output_student_and_course_name” which was added to display the names of the students and the courses they are enrolled in. I included the parameter of ‘student_data’ which is a list of dictionaries where it contains the first name, last name, and course name of a student. This method is used to display the current list of registered students and their respective courses to the user.

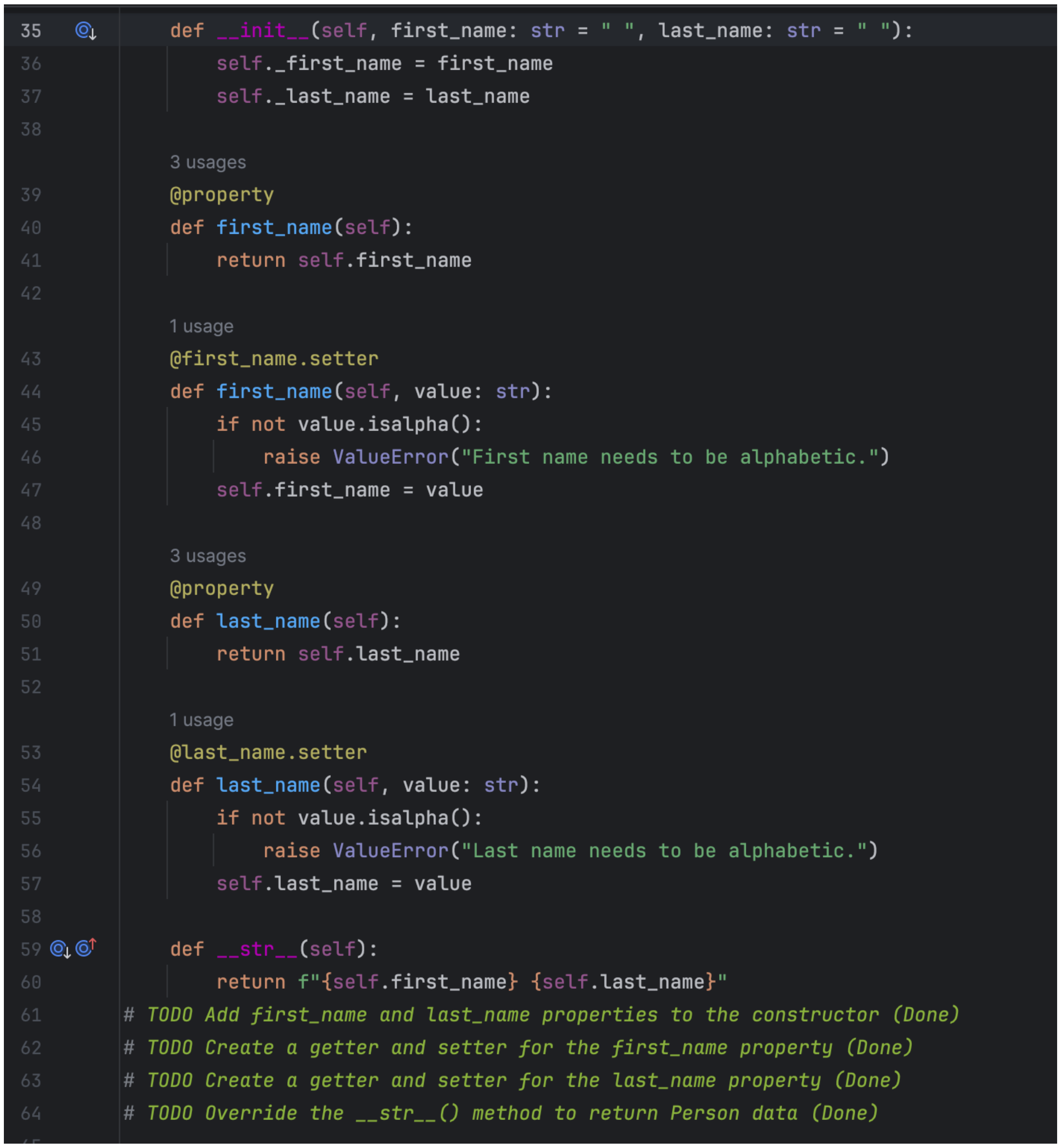
I then added the “input_student_data”. As shown in Figure 8-9, I added this part since it collects data for a new student (first name, last name, course name) from the user, and adds it to the ‘student_data’ list. This part of the data makes the user enter the student’s first name, last name, and course name, it also validated the first and last names to make sure they are alphabetic, added the student data to the list if all inputs are valid and displays a confirmation message once the student is registered. I then added the error handling part of the code which wrote out that if the first or last name is non-alphabetical characters, it raises a ‘ValueError’. And if any other exception occurs, it then catches it and displays the custom error message using the ‘output_error_messages’.

Finally, I then added the code that is the main control flow for a simple course registration program. It uses a loop to interact with the user, allowing them to register students for courses, views the current registrations, saves the data, and exits the program. As seen in Figure 10, I had the program start and included a script that printed “opening the program…”, added another script that loads the data from the JSON file into the ‘students’ list. I then added the main loop include the “while true” infinite loop, displayed the menu options to the user, get the user choice to get the user’s choice from the menu, and then processed the user choice. I then included the different options the user chooses such as if they choose option 1, 2, 3, or 4. I then ended the program that if the user chooses option "4", to exit the loop and end the program using the "break" method. I also added the invalid choice handing that states if the user inputted something other than 1,2,3,4, then print the code "Please only choose option 1, 2, or 3". Finally, then added the print() statement "Program Ended".



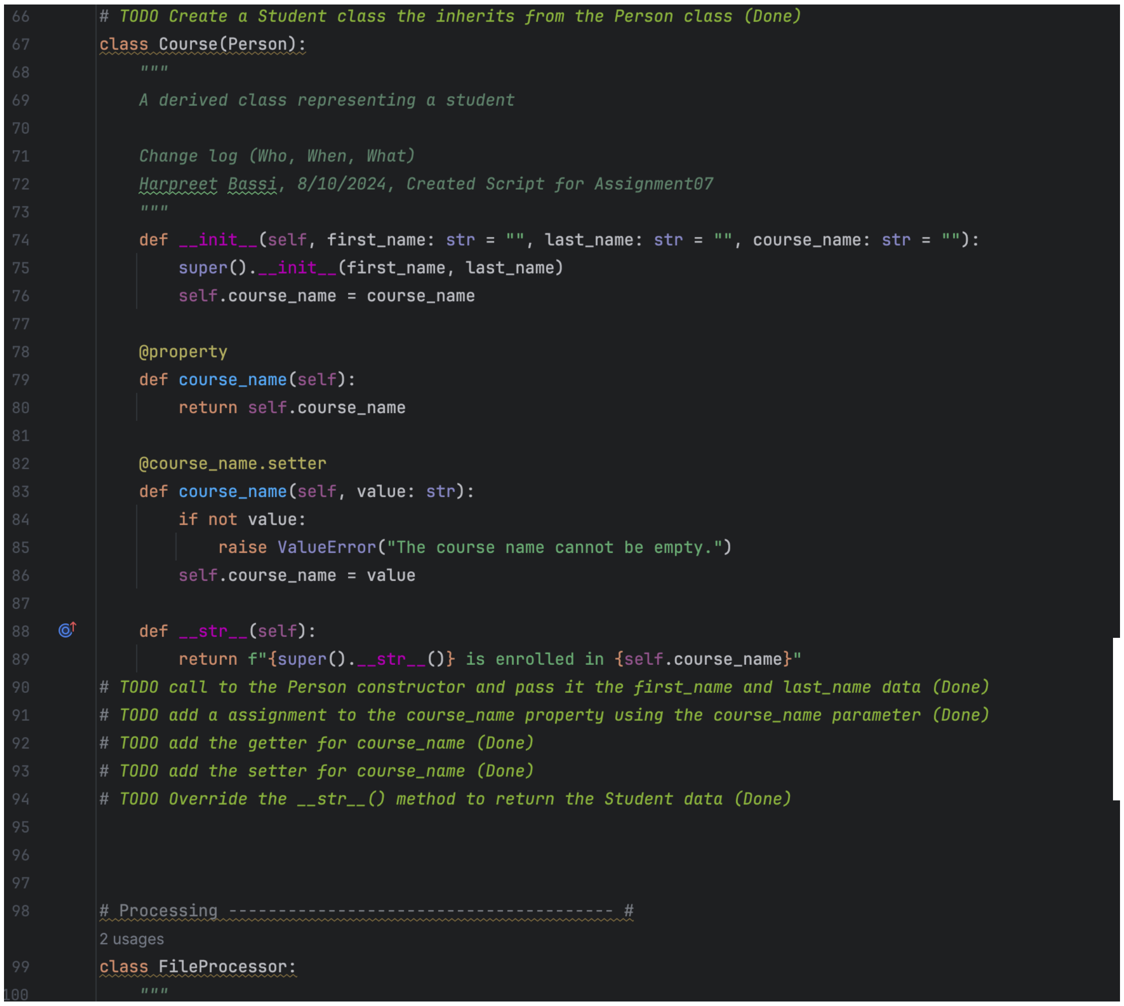
```python
# ---------------------------------------------------------------------------------------- #
# Title: Assignment07
# Desc: This assignment demonstrates using data classes
# with structured error handling
# Change Log: (Who, When, What)
#   Harpreet Bassi, 8/10/2024, Created Script for Assignment07
# ---------------------------------------------------------------------------------------- #
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
students: list = []  # a table of student data
menu_choice: str  # Hold the choice made by the user.


# TODO Create a Person Class
1 usage
class Person:
    """
    A base class for a person

    Change log (Who, When, What)
    Harpreet Bassi, 8/10/2024, Created Script for Assignment07
    """
    def __init__(self, first_name: str = " ", last_name: str = " "):
```

Figure 1: Screenshot of my Final Script in the PyCharm development program



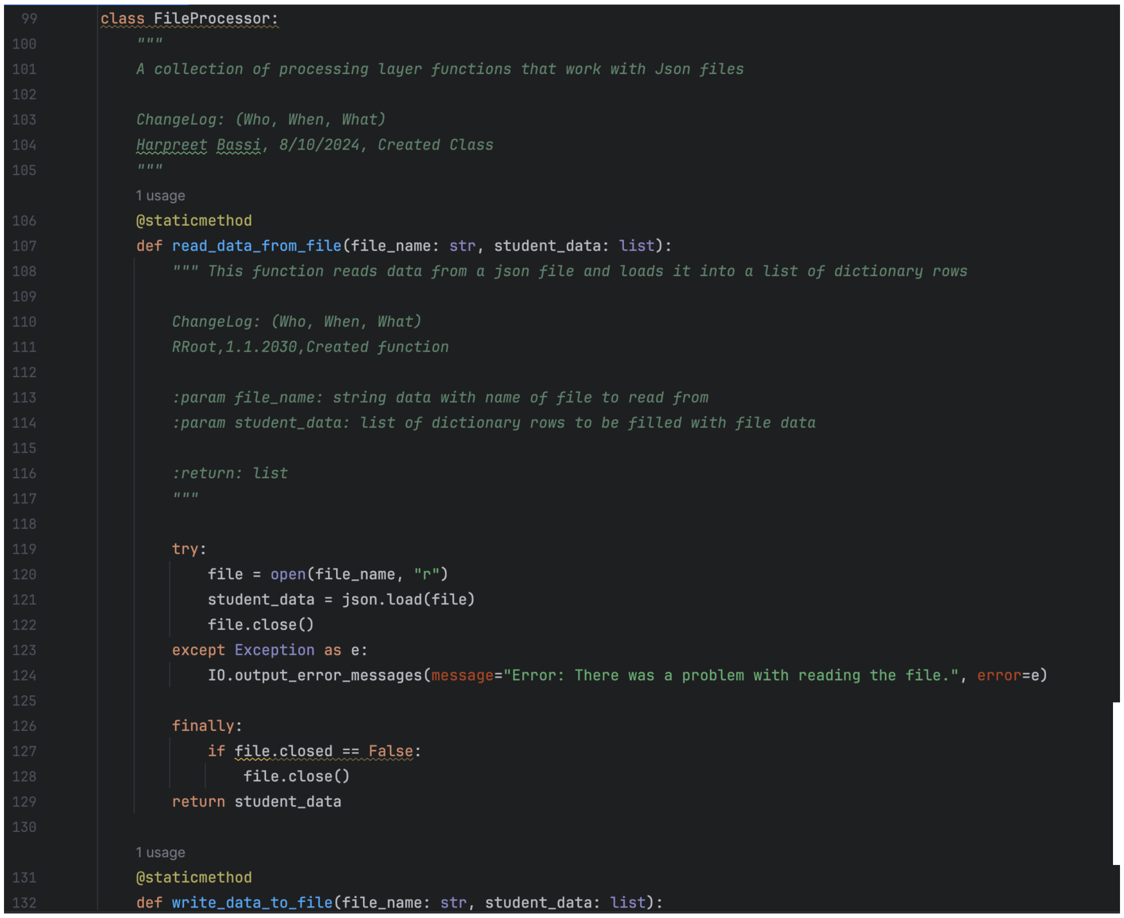
```
    def __init__(self, first_name: str = " ", last_name: str = " "):
        self._first_name = first_name
        self._last_name = last_name

    3 usages
    @property
    def first_name(self):
        return self.first_name

    1 usage
    @first_name.setter
    def first_name(self, value: str):
        if not value.isalpha():
            raise ValueError("First name needs to be alphabetic.")
        self.first_name = value

    3 usages
    @property
    def last_name(self):
        return self.last_name

    1 usage
    @last_name.setter
    def last_name(self, value: str):
        if not value.isalpha():
            raise ValueError("Last name needs to be alphabetic.")
        self.last_name = value

    def __str__(self):
        return f"{self.first_name} {self.last_name}"
# TODO Add first_name and last_name properties to the constructor (Done)
# TODO Create a getter and setter for the first_name property (Done)
# TODO Create a getter and setter for the last_name property (Done)
# TODO Override the __str__() method to return Person data (Done)
```

Figure 2: Screenshot of my Final Script in the PyCharm development program



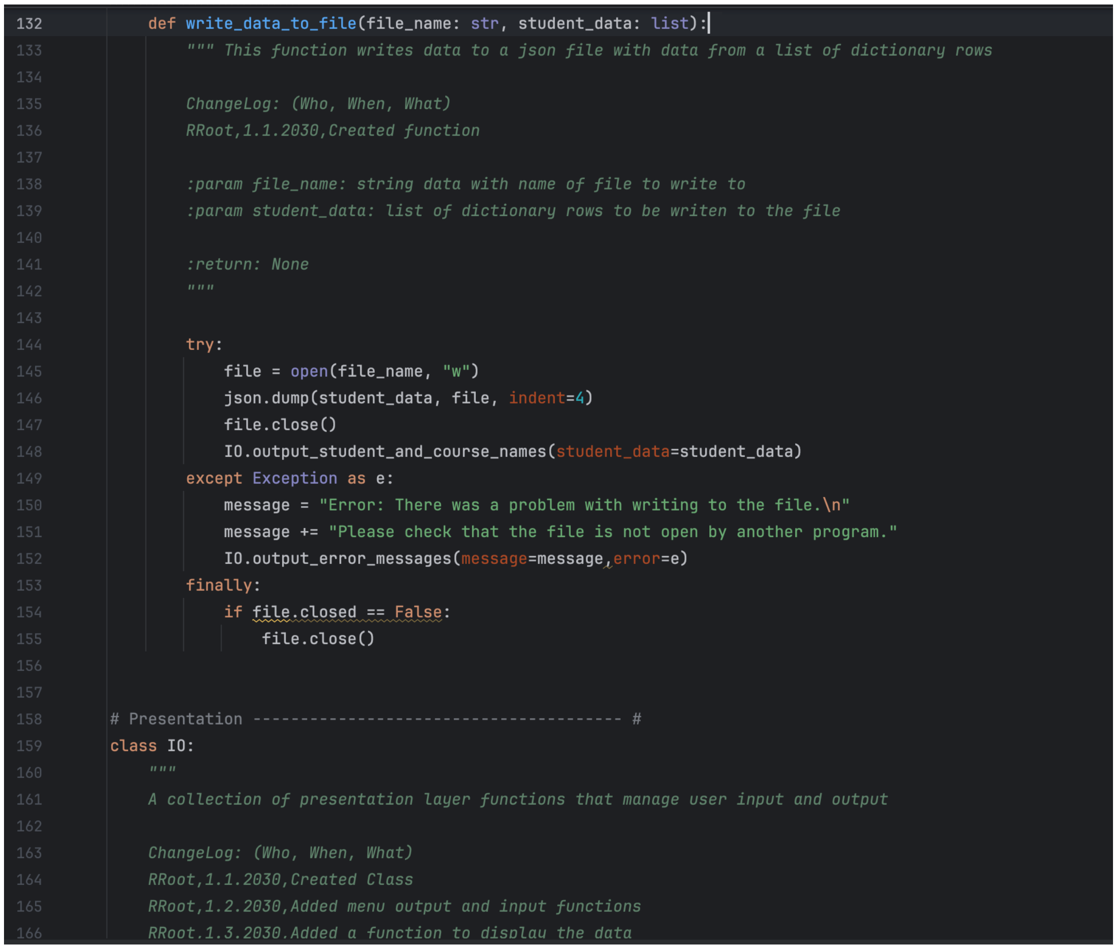
```python
# TODO Create a Student class the inherits from the Person class (Done)
class Course(Person):
    """
    A derived class representing a student

    Change log (Who, When, What)
    Harpreet Bassi, 8/10/2024, Created Script for Assignment07
    """
    def __init__(self, first_name: str = "", last_name: str = "", course_name: str = ""):
        super().__init__(first_name, last_name)
        self.course_name = course_name

    @property
    def course_name(self):
        return self.course_name

    @course_name.setter
    def course_name(self, value: str):
        if not value:
            raise ValueError("The course name cannot be empty.")
        self.course_name = value

    def __str__(self):
        return f"{super().__str__()} is enrolled in {self.course_name}"
# TODO call to the Person constructor and pass it the first_name and last_name data (Done)
# TODO add a assignment to the course_name property using the course_name parameter (Done)
# TODO add the getter for course_name (Done)
# TODO add the setter for course_name (Done)
# TODO Override the __str__() method to return the Student data (Done)



# Processing --------------------------------------- #
2 usages
class FileProcessor:
    """
```

Figure 3: Screenshot of my Final Script in the PyCharm development program



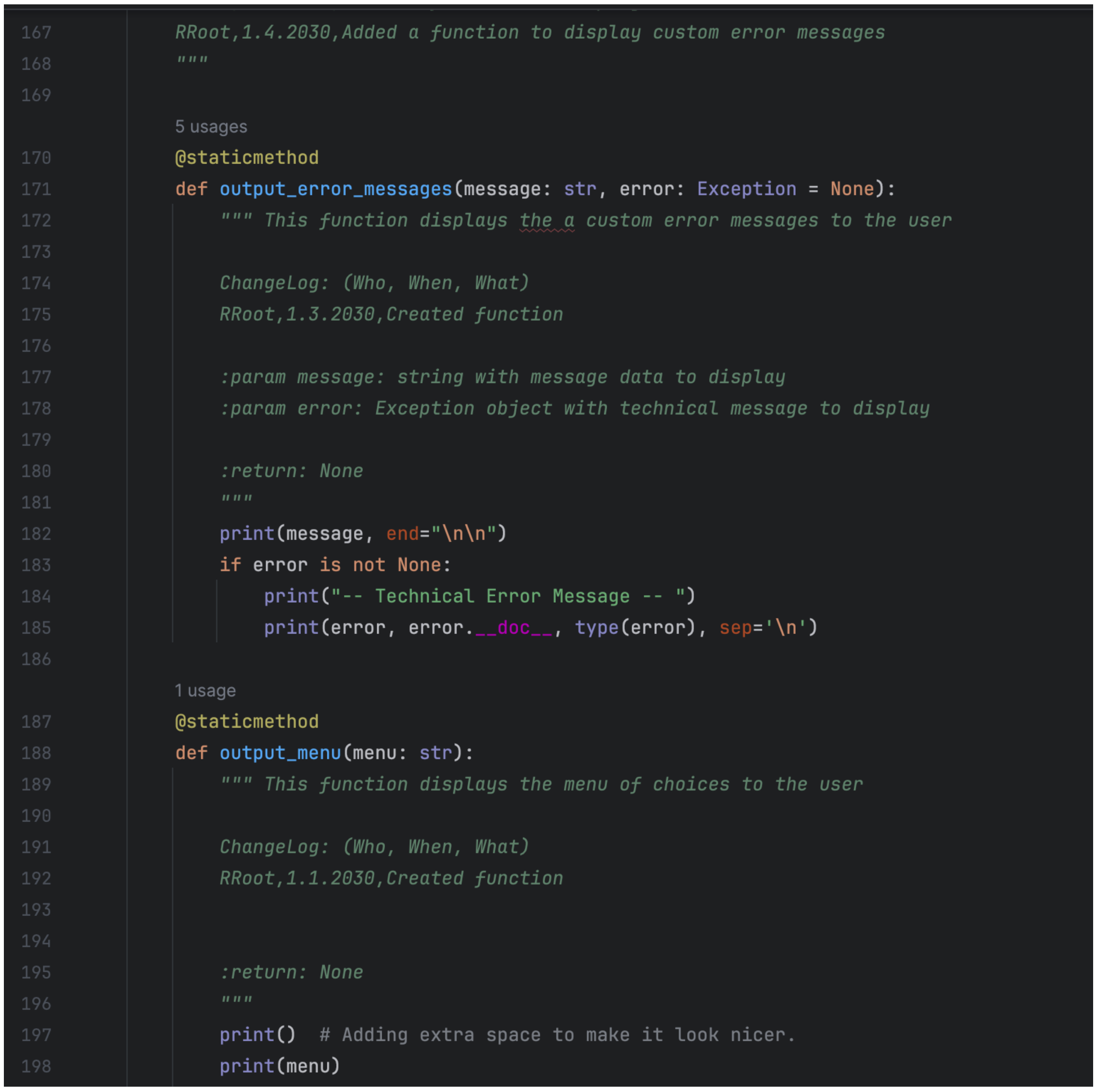
```
class FileProcessor:
    """
    A collection of processing layer functions that work with Json files

    ChangeLog: (Who, When, What)
    Harpreet Bassi, 8/10/2024, Created Class
    """
    1 usage
    @staticmethod
    def read_data_from_file(file_name: str, student_data: list):
        """ This function reads data from a json file and loads it into a list of dictionary rows

        ChangeLog: (Who, When, What)
        RRoot,1.1.2030,Created function

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

        :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

    1 usage
    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
```

Figure 4: Screenshot of my Final Script in the PyCharm development program



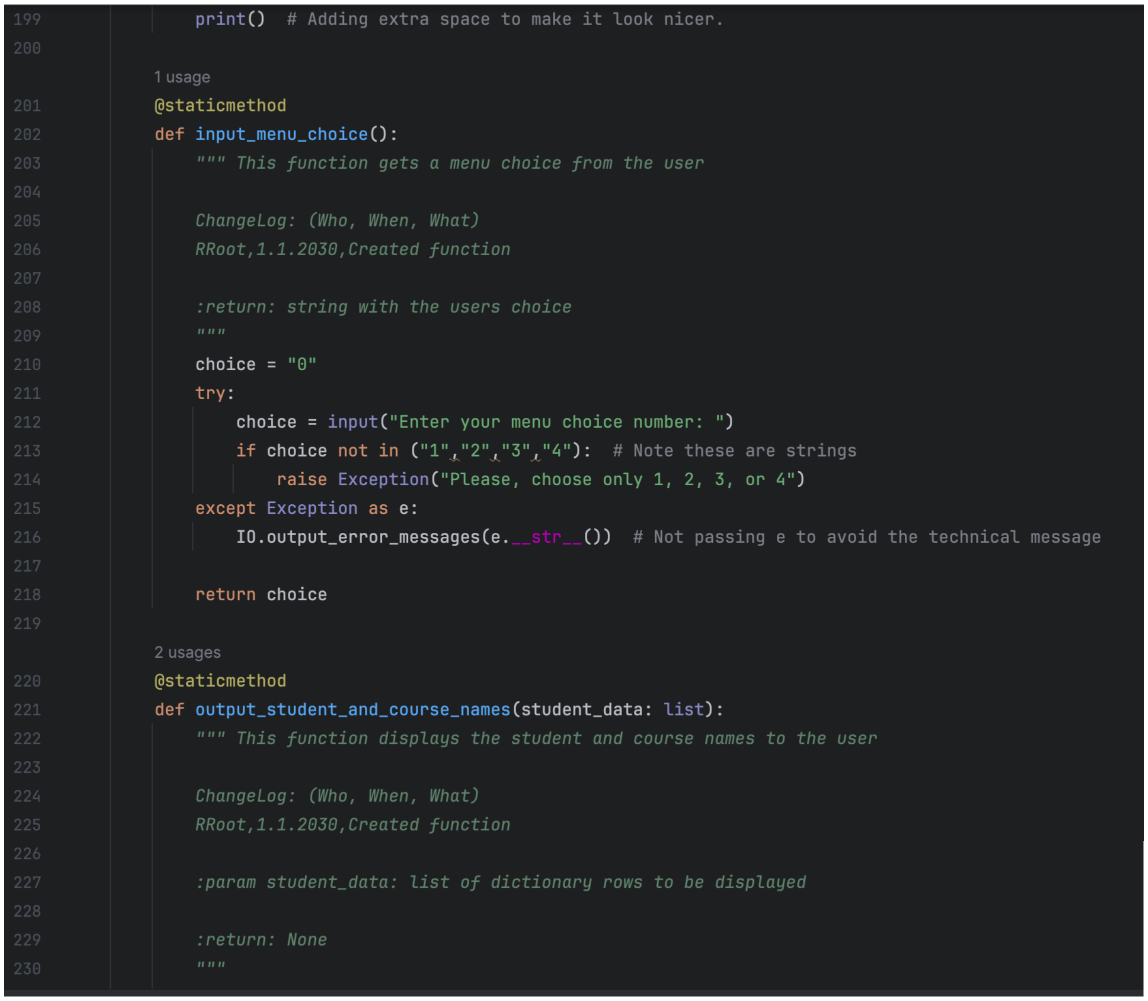
```
    def write_data_to_file(file_name: str, student_data: list):
        """ This function writes data to a json file with data from a list of dictionary rows

        ChangeLog: (Who, When, What)
        RRoot,1.1.2030,Created function

        :param file_name: string data with name of file to write to
        :param student_data: list of dictionary rows to be writen to the file

        :return: None
        """

        try:
            file = open(file_name, "w")
            json.dump(student_data, file, indent=4)
            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()


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

    ChangeLog: (Who, When, What)
    RRoot,1.1.2030,Created Class
    RRoot,1.2.2030,Added menu output and input functions
    RRoot,1.3.2030,Added a function to display the data
```

Figure 5: Screenshot of my Final Script in the PyCharm development program



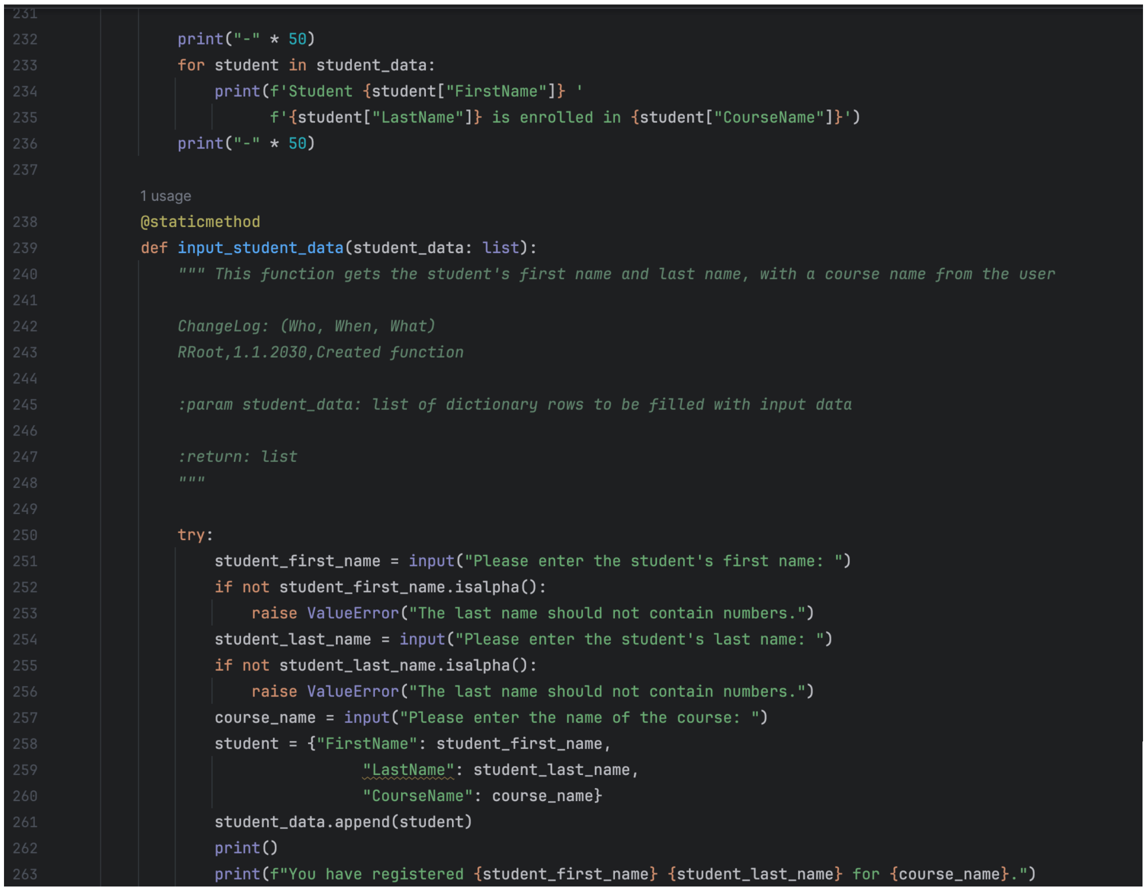
```
    RRoot,1.4.2030,Added a function to display custom error messages
    """

    5 usages
    @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)
        RRoot,1.3.2030,Created function

        :param message: string with message data to display
        :param error: Exception object with technical message to display

        :return: None
        """
        print(message, end="\n\n")
        if error is not None:
            print("-- Technical Error Message -- ")
            print(error, error.__doc__, type(error), sep='\n')

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

        ChangeLog: (Who, When, What)
        RRoot,1.1.2030,Created function


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

Figure 6: Screenshot of my Final Script in the PyCharm development program



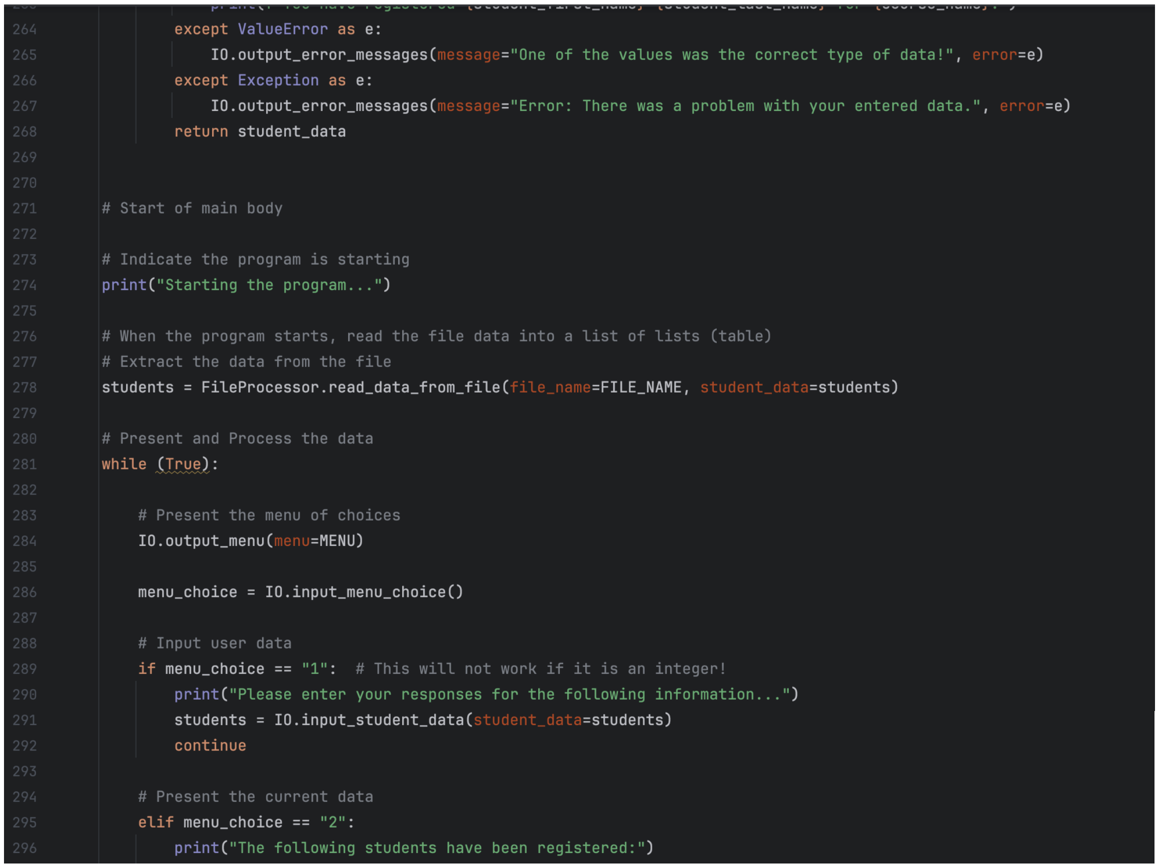
```
        print()  # Adding extra space to make it look nicer.

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

        ChangeLog: (Who, When, What)
        RRoot,1.1.2030,Created function

        :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

    2 usages
    @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,1.1.2030,Created function

        :param student_data: list of dictionary rows to be displayed

        :return: None
        """
```

Figure 7: Screenshot of my Final Script in the PyCharm development program



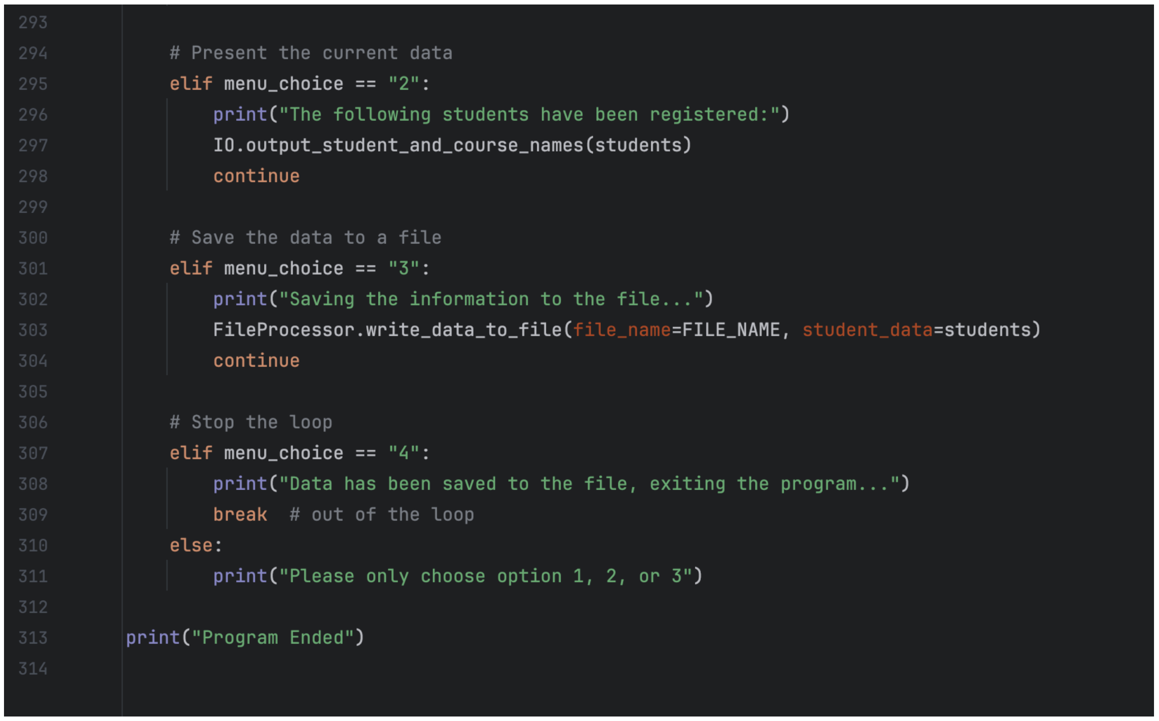
```python
        print("-" * 50)
        for student in student_data:
            print(f'Student {student["FirstName"]} '
                  f'{student["LastName"]} is enrolled in {student["CourseName"]}')
        print("-" * 50)

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

        ChangeLog: (Who, When, What)
        RRoot,1.1.2030,Created function

        :param student_data: list of dictionary rows to be filled with input data

        :return: list
        """

        try:
            student_first_name = input("Please 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("Please 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()
            print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
```

Figure 8: Screenshot of my Final Script in the PyCharm development program



```
        except ValueError as e:
            IO.output_error_messages(message="One of the values was the correct type of 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


# Start of main body

# Indicate the program is starting
print("Starting the program...")

# 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=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!
        print("Please enter your responses for the following information...")
        students = IO.input_student_data(student_data=students)
        continue

    # Present the current data
    elif menu_choice == "2":
        print("The following students have been registered:")
```

Figure 9: Screenshot of my Final Script in the PyCharm development program



```

    # Present the current data
    elif menu_choice == "2":
        print("The following students have been registered:")
        IO.output_student_and_course_names(students)
        continue

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

    # Stop the loop
    elif menu_choice == "4":
        print("Data has been saved to the file, exiting the program...")
        break  # out of the loop
    else:
        print("Please only choose option 1, 2, or 3")

print("Program Ended")

```

Figure 10: Screenshot of my Final Script in the PyCharm development program

Troubleshooting

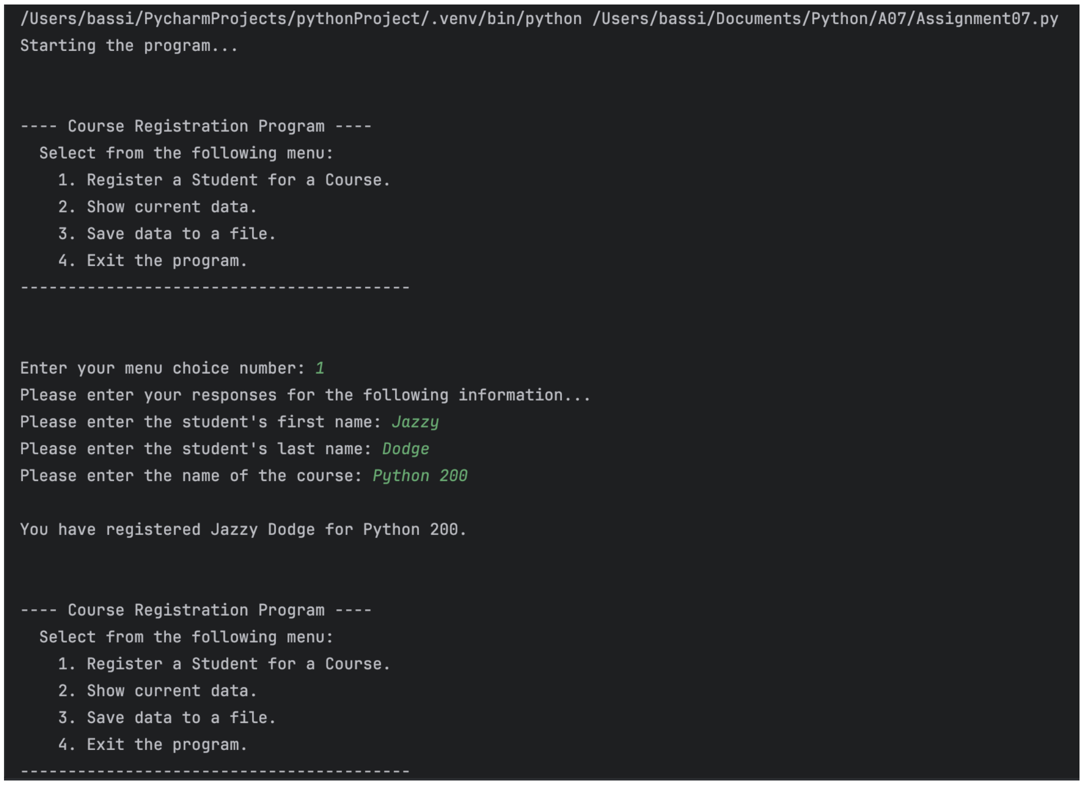
I did not run into many errors in this script; therefore, I did not have much troubleshooting to do for Assignment 07.

Save the Script

I created a folder in Documents called "Python" and saved my python script as "Assignment07.py".

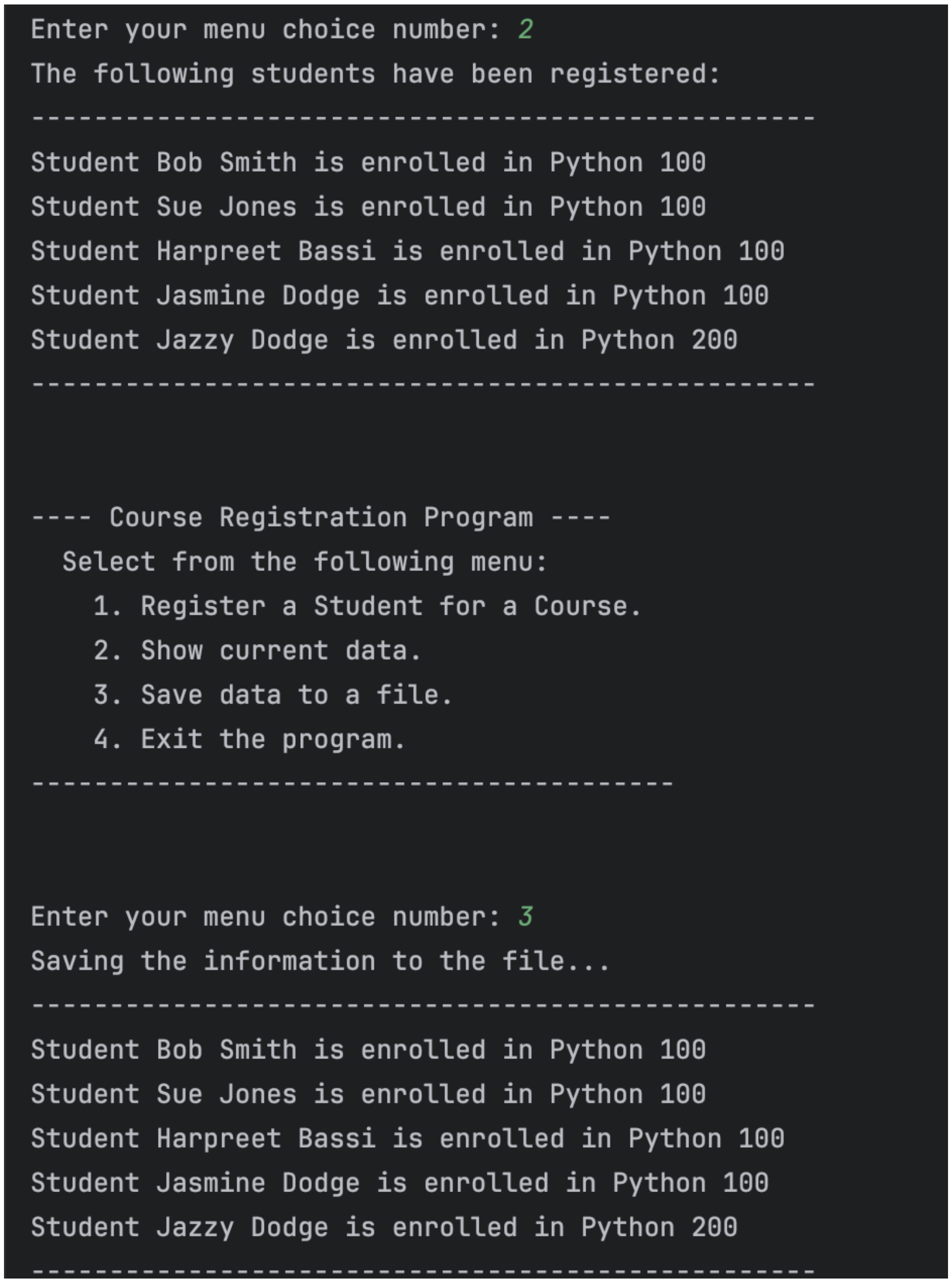
Running the Script in Terminal and PyCharm

I opened the Terminal console on my MacBook and went to the folder where the python script is located using the cd (change directory) command. I started off with using the cd command for Documents. I mainly used this for the Documents for tracking purposes for myself. I then went onto using the cd command for the Python folder in Documents, used the cd command again for the "A07" folder where the script is located, and then went onto using "python3 Assignment07.py" which then directed me to the code. Once I inputted the information the code prompted me to, I was able to successfully have the program run on both PyCharm (Figure 11-13) and Terminal (Figure 14-14.5).



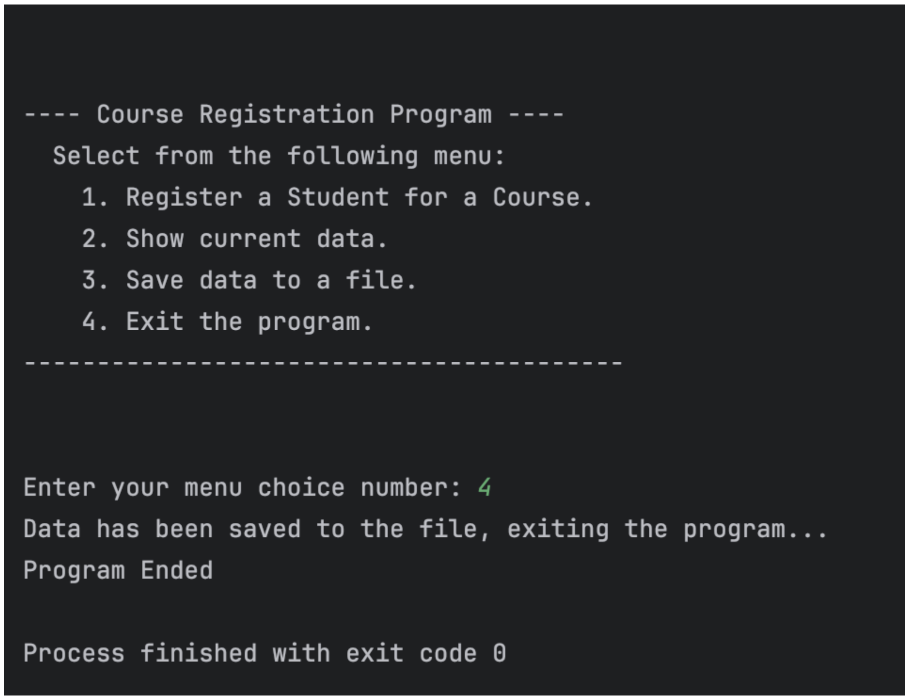
```
/Users/bassi/PycharmProjects/pythonProject/.venv/bin/python /Users/bassi/Documents/Python/A07/Assignment07.py
Starting the program...


---- 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
Please enter your responses for the following information...
Please enter the student's first name: Jazzy
Please enter the student's last name: Dodge
Please enter the name of the course: Python 200

You have registered Jazzy Dodge for 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.
-----------------------------------------
```

Figure 11: Screenshot of the final run for Assignment05.py in PyCharm.



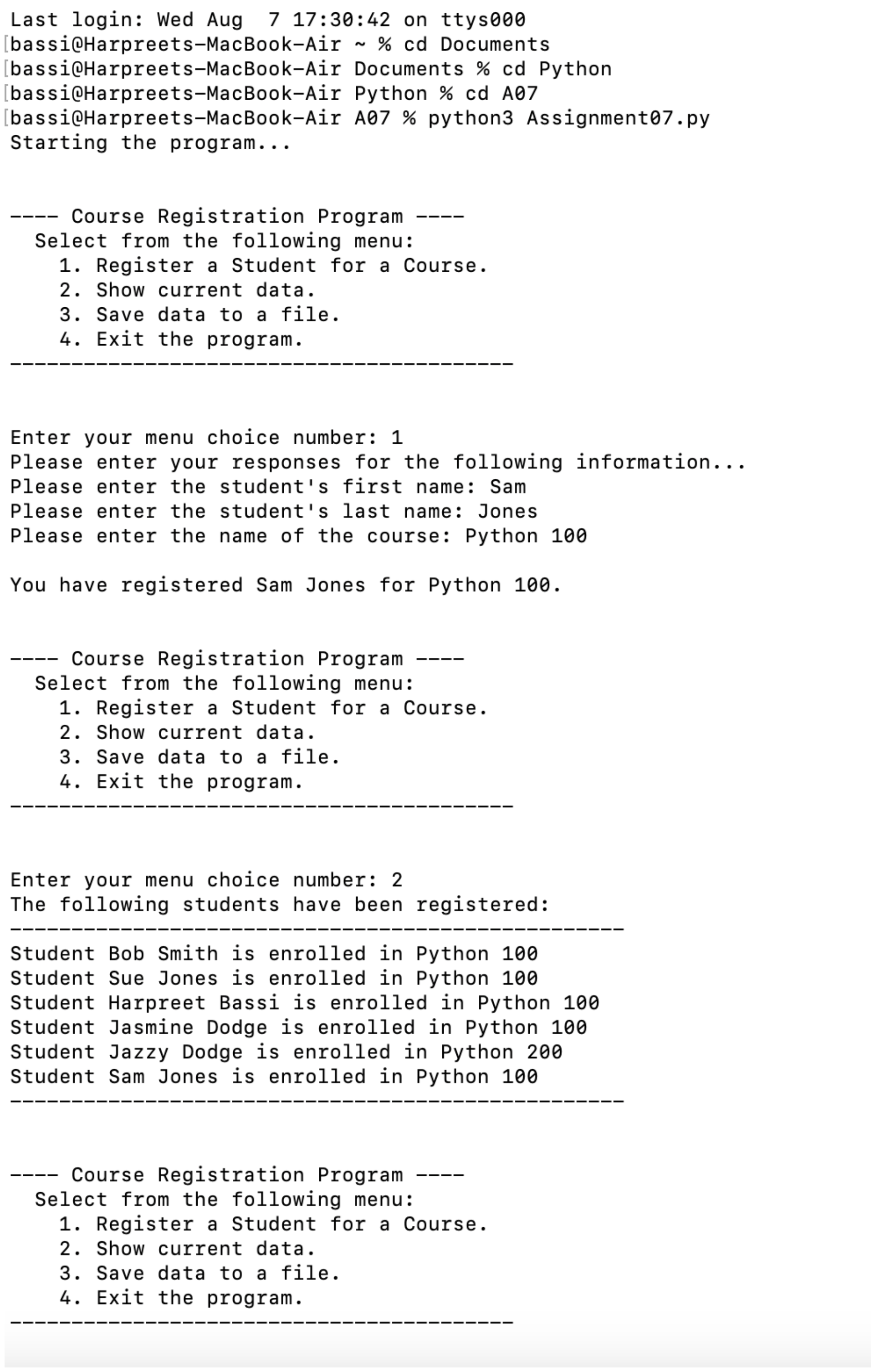
```
Enter your menu choice number: 2
The following students have been registered:
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Harpreet Bassi is enrolled in Python 100
Student Jasmine Dodge is enrolled in Python 100
Student Jazzy Dodge 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.
-----------------------------------------



Enter your menu choice number: 3
Saving the information to the file...
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Harpreet Bassi is enrolled in Python 100
Student Jasmine Dodge is enrolled in Python 100
Student Jazzy Dodge is enrolled in Python 200
--------------------------------------------------
```

Figure 12: Screenshot of the final run for Assignment05.py in PyCharm.



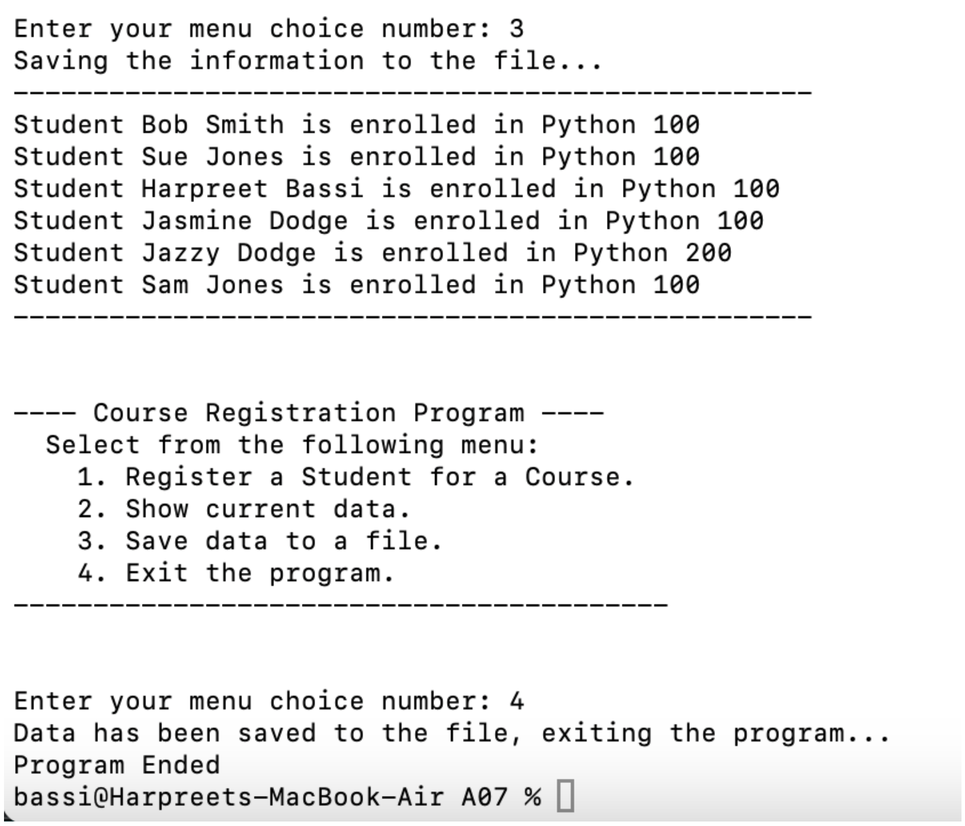
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------



Enter your menu choice number: 4
Data has been saved to the file, exiting the program...
Program Ended

Process finished with exit code 0
```

Figure 13: Screenshot of the final run for Assignment05.py in PyCharm.



```
Last login: Wed Aug  7 17:30:42 on ttys000
bassi@Harpreets-MacBook-Air ~ % cd Documents
bassi@Harpreets-MacBook-Air Documents % cd Python
bassi@Harpreets-MacBook-Air Python % cd A07
bassi@Harpreets-MacBook-Air A07 % python3 Assignment07.py
Starting the program...


---- 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
Please enter your responses for the following information...
Please enter the student's first name: Sam
Please enter the student's last name: Jones
Please enter the name of the course: Python 100

You have registered Sam Jones for 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: 2
The following students have been registered:
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Harpreet Bassi is enrolled in Python 100
Student Jasmine Dodge is enrolled in Python 100
Student Jazzy Dodge is enrolled in Python 200
Student Sam Jones 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.
-----------------------------------------
```

Figure 14: Screenshot of the commands to locate the folder, script, and run Assignment05.py

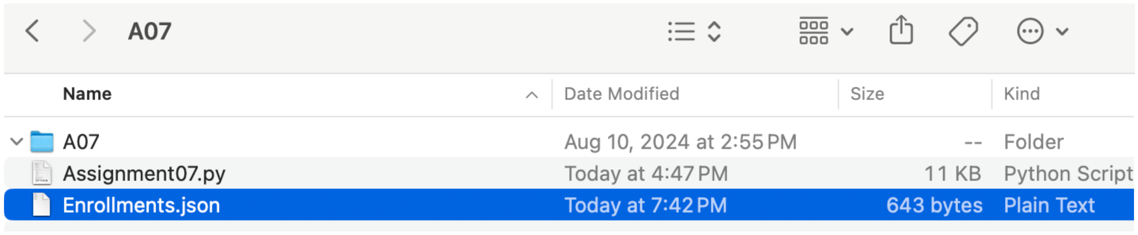


```
Enter your menu choice number: 3
Saving the information to the file...
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Harpreet Bassi is enrolled in Python 100
Student Jasmine Dodge is enrolled in Python 100
Student Jazzy Dodge is enrolled in Python 200
Student Sam Jones 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: 4
Data has been saved to the file, exiting the program...
Program Ended
bassi@Harpreets-MacBook-Air A07 %
```

Figure 14.5: Screenshot of the commands to locate the folder, script, and run Assignment05.py

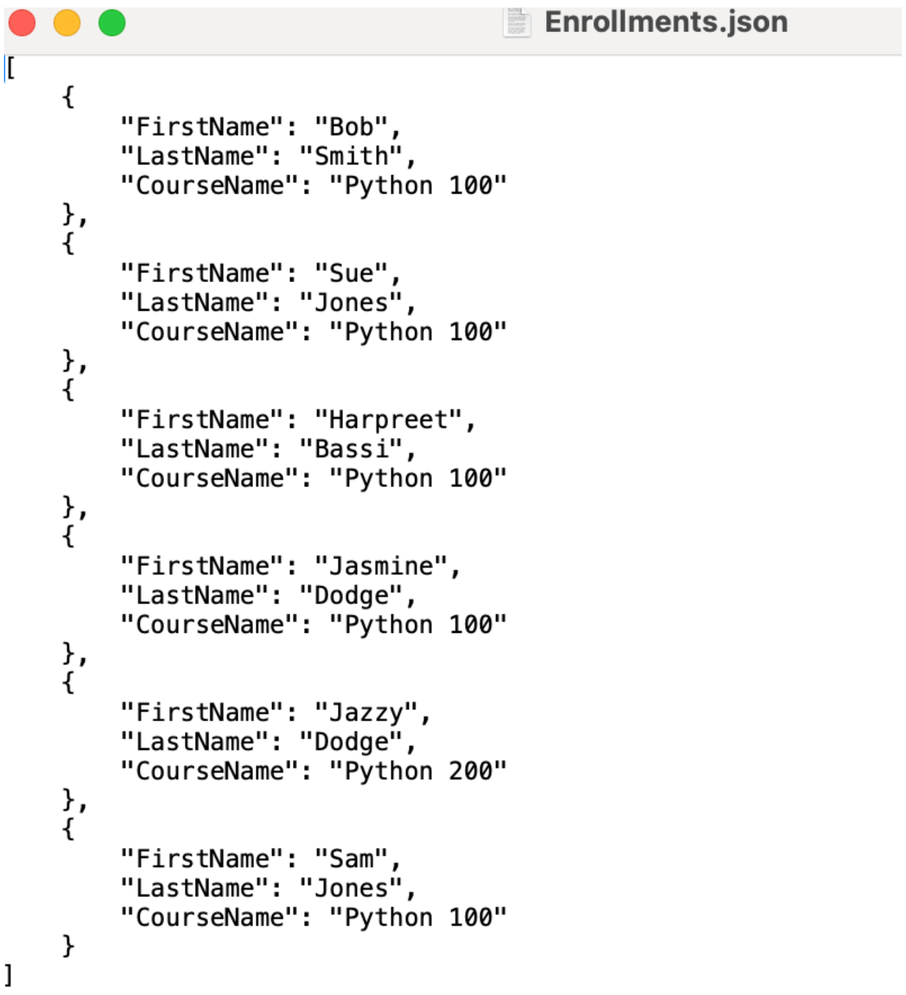
File Processing

Once I ran the code on PyCharm and Terminal, I then went to my Python folder which would have the JSON file saved. As seen in Figure 15, I could see the JSON file that was labeled as “Enrollments.json” just like how I had indicated in my code. I opened the file, which then opened to the TextEdit in Mac, as shown in Figure 16. As you can see in Figure 15 and 16, the code successfully implemented the information in a JSON file.



Name	Date Modified	Size	Kind
A07	Aug 10, 2024 at 2:55 PM	--	Folder
Assignment07.py	Today at 4:47 PM	11 KB	Python Script
Enrollments.json	Today at 7:42 PM	643 bytes	Plain Text

Figure 15: Screenshot of Python folder which includes the JSON file.



Enrollments.json

```
[
    {
        "FirstName": "Bob",
        "LastName": "Smith",
        "CourseName": "Python 100"
    },
    {
        "FirstName": "Sue",
        "LastName": "Jones",
        "CourseName": "Python 100"
    },
    {
        "FirstName": "Harpreet",
        "LastName": "Bassi",
        "CourseName": "Python 100"
    },
    {
        "FirstName": "Jasmine",
        "LastName": "Dodge",
        "CourseName": "Python 100"
    },
    {
        "FirstName": "Jazzy",
        "LastName": "Dodge",
        "CourseName": "Python 200"
    },
    {
        "FirstName": "Sam",
        "LastName": "Jones",
        "CourseName": "Python 100"
    }
]
```

Figure 16: Screenshot of JSON file opened in TextEdit on Mac.

Summary

Applying the Module 07 lecture notes and video, I was able to implement and execute an effective Python program that demonstrated my understanding of how to create and use classes to manage data. Also, learned how to implement this data in a JavaScript Object Notation (JSON) file and GitHub. This led me to effectively create a JSON data file of all the information needed regarding a student's registration for a Python course.