Jenna Ho
Nov. 22, 2023
Foundations of Programming: Python
Assignment07
https://github.com/jennah2001/IntroToProg-Python-Mod07

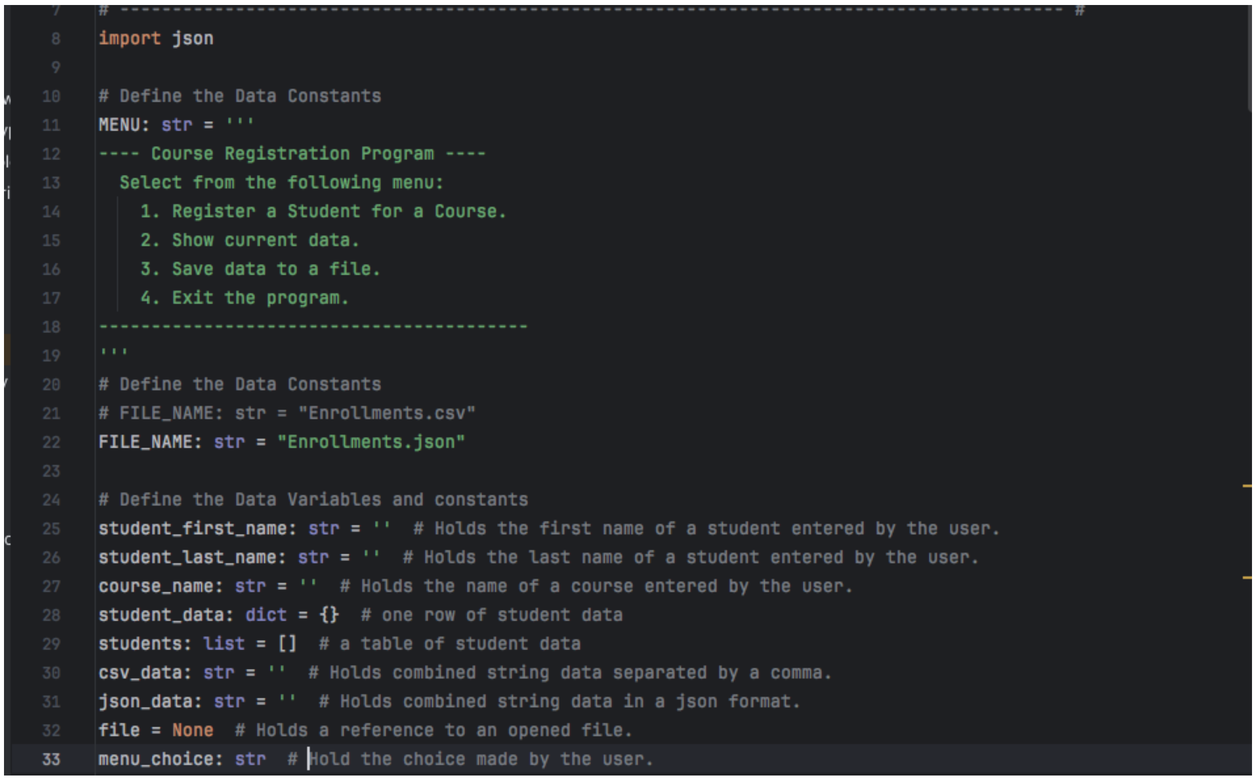
Module07 Completing the Assignment

Introduction

In Module07, the use of functions, classes and the separation of concerns pattern are introduced. Classes can be used to describe objects, and functions are used to perform operations on data described. By using functions and classes, we can shorten our code, organize and assign our data better. JSON files are also used in this assignment, which was introduced in the previous lectures. In addition, Github is being used to store the codes we have written as well as this document.

Data Constants

First import file is stated at the top of the file. Then the data constants are stated. The data variables stayed the same as past assignments, this includes the student's first name, last name, course name, menu choice and more.



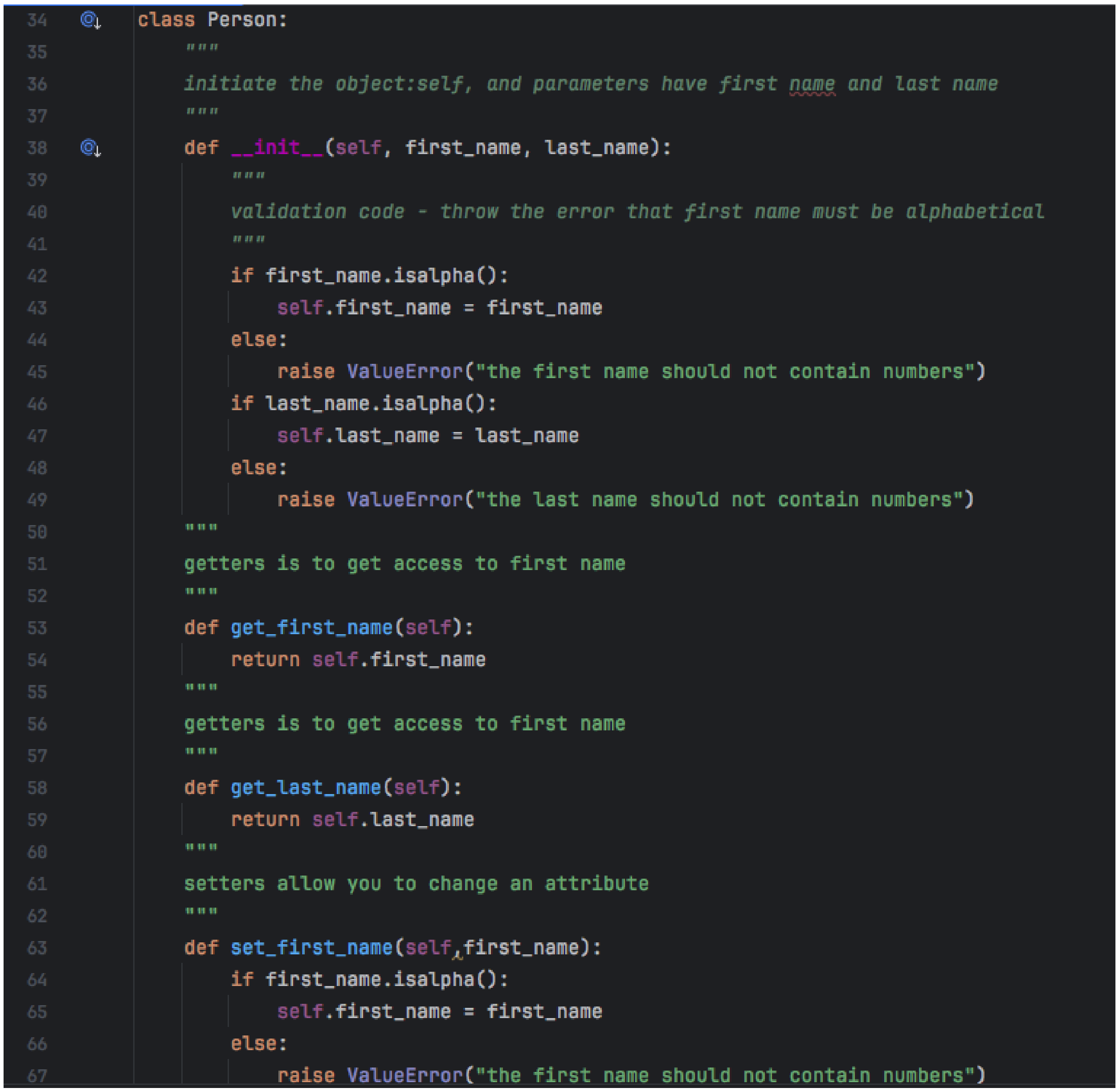
```
import json

# 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.csv"
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables and constants
student_first_name: str = ''  # Holds the first name of a student entered by the user.
student_last_name: str = ''  # Holds the last name of a student entered by the user.
course_name: str = ''  # Holds the name of a course entered by the user.
student_data: dict = {}  # one row of student data
students: list = []  # a table of student data
csv_data: str = ''  # Holds combined string data separated by a comma.
json_data: str = ''  # Holds combined string data in a json format.
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
```

Figure 1: Data constants

Define Classes

For this assignment, there’s a requirement of using classes. The program should have one class named FIleProcessor, one class named IO, one class named Person, and one class named Student. It is important to note that all classes need to have descriptive document strings. There are also a number of functions required for the assignment, this includes output error messages, output menu, input menu choice, output student courses, input student data, read data from file, as well as write data to file.

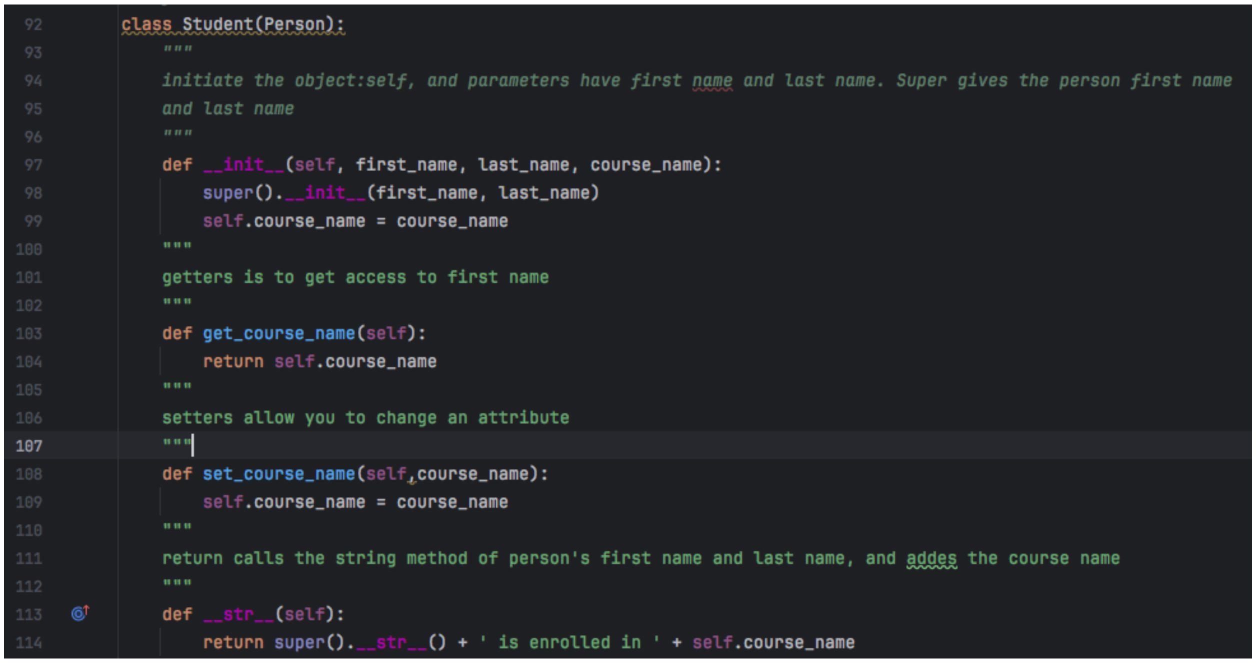
For class Person, I used def and__init__ to initiate the object, self, as well as having first name, last name as parameters. Then I used if else statements to throw any error that's not alphabetical. I used getters and setters to access and mutate attributes in the class Person. I used getters to get access to first name, last name, and change the attributes first name and last name by using setters. Then I wrote an return for string self to return the person’s first name and last name.



```python
class Person:
    """
    initiate the object:self, and parameters have first name and last name
    """
    def __init__(self, first_name, last_name):
        """
        validation code - throw the error that first name must be alphabetical
        """
        if first_name.isalpha():
            self.first_name = first_name
        else:
            raise ValueError("the first name should not contain numbers")
        if last_name.isalpha():
            self.last_name = last_name
        else:
            raise ValueError("the last name should not contain numbers")
    """
    getters is to get access to first name
    """
    def get_first_name(self):
        return self.first_name
    """
    getters is to get access to first name
    """
    def get_last_name(self):
        return self.last_name
    """
    setters allow you to change an attribute
    """
    def set_first_name(self,first_name):
        if first_name.isalpha():
            self.first_name = first_name
        else:
            raise ValueError("the first name should not contain numbers")
```

Figure 2: Class Person

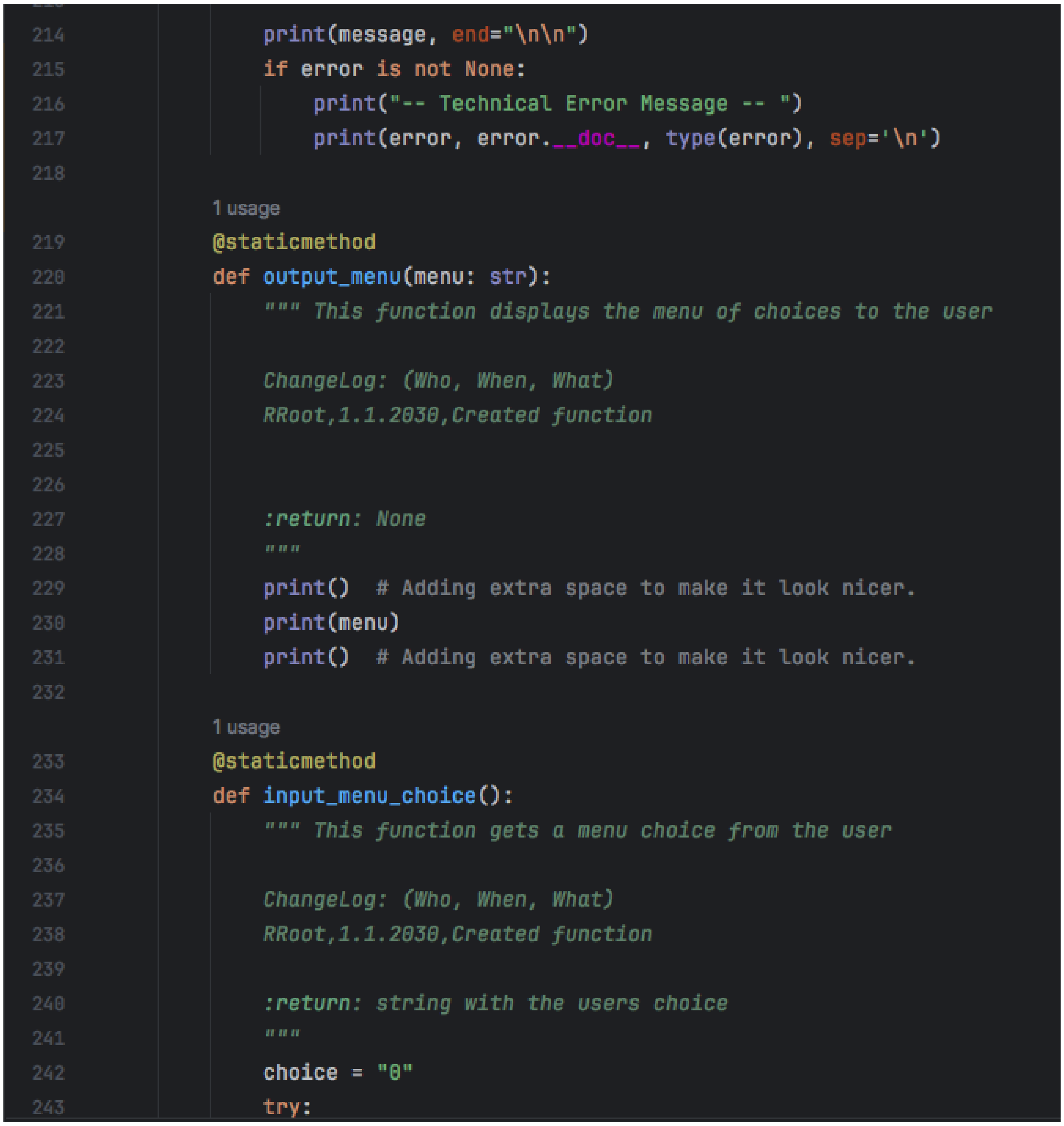
For class Students, students inherit everything from the class person, but not vice versa. I repeated the same steps from class Person to initiate the object self, and have first name and last name as parameters. I used getters and setters to access and mutate the attribute course name. Then wrote a return to call for the strong method of a person's first name, last name, being enrolled in course name.



```
class Student(Person):
    """
    initiate the object:self, and parameters have first name and last name. Super gives the person first name
    and last name
    """
    def __init__(self, first_name, last_name, course_name):
        super().__init__(first_name, last_name)
        self.course_name = course_name
    """
    getters is to get access to first name
    """
    def get_course_name(self):
        return self.course_name
    """
    setters allow you to change an attribute
    """
    def set_course_name(self,course_name):
        self.course_name = course_name
    """
    return calls the string method of person's first name and last name, and addes the course name
    """
    def __str__(self):
        return super().__str__() + ' is enrolled in ' + self.course_name
```

Figure 3: Class Student

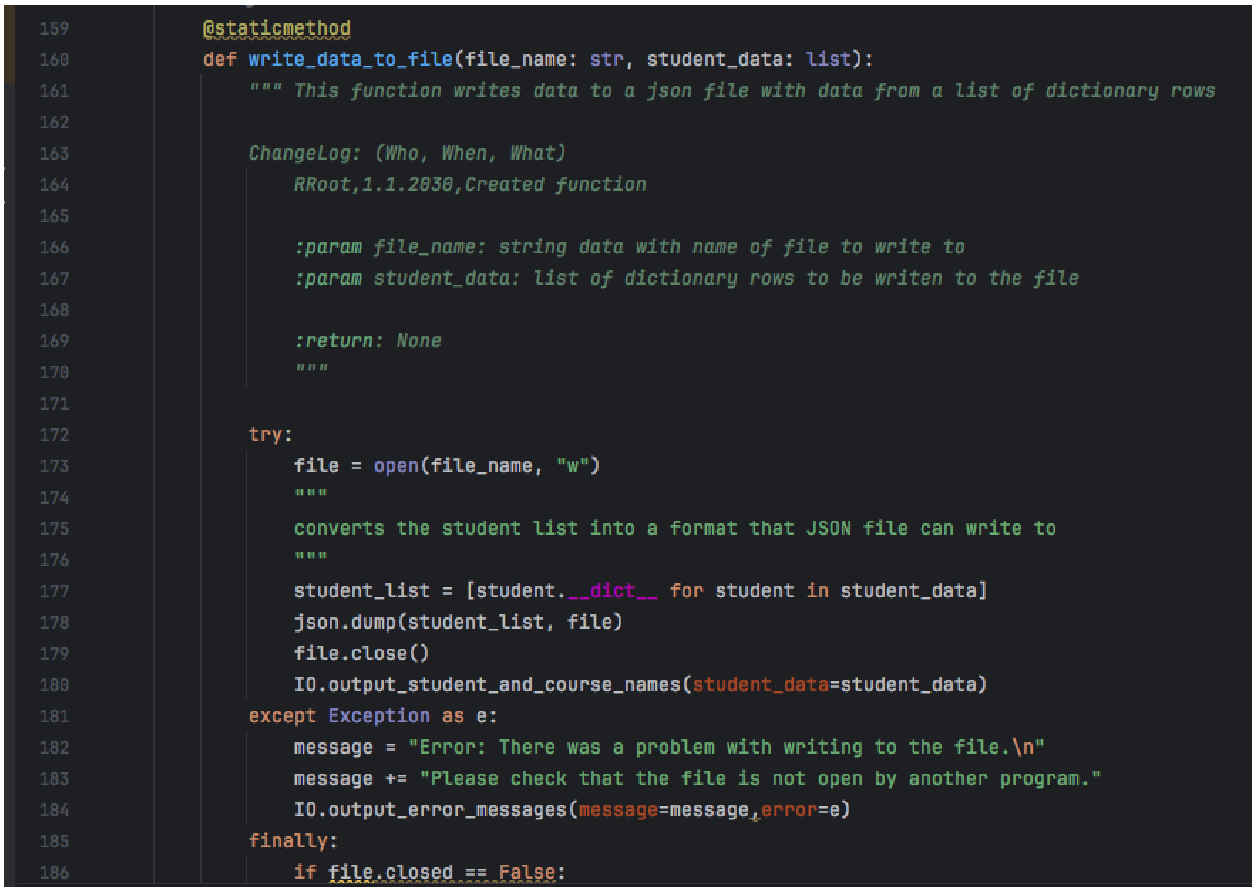
For Class IO, it is a class for all inputs and outputs of student data. My code stayed about the same as last assignment, where I used @staticmethod decorator to have a function pass as a parameter. Then I used def to define the output error messages as a function, and printed out the error messages. I repeat the use of @staticmethod decorator, and def to print out menu and menu choice. Then I added return menu choice to return back to menu choice. I also repeated the same steps for output student courses, which add students into the student list. Same steps for defining input student data as well, where it adds students into student data.



```
            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)
        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:
```

Figure 4: CLass IO

For class FileProcessor, it is a class that reads the file, inputs the student data, and writes student data to the file. I defined __init__ as the constructor for the class IO,a dn the self parameter refers to the object io. This is to call the instance variable within its own class. Then to read front the file and put it in the student list, I again used @staticmethod decorator and def to open and read the file, then JSON file is loaded, then file will close. I also used Exception to catch any error, and print out the message that there is an error. Then I used finally to close the file if it is not closed. I repeated the same steps for writing data to a file, where I open the file and write to it, then JSON answers and writes the student to the file. I then again used Exception to catch any error and close the file.



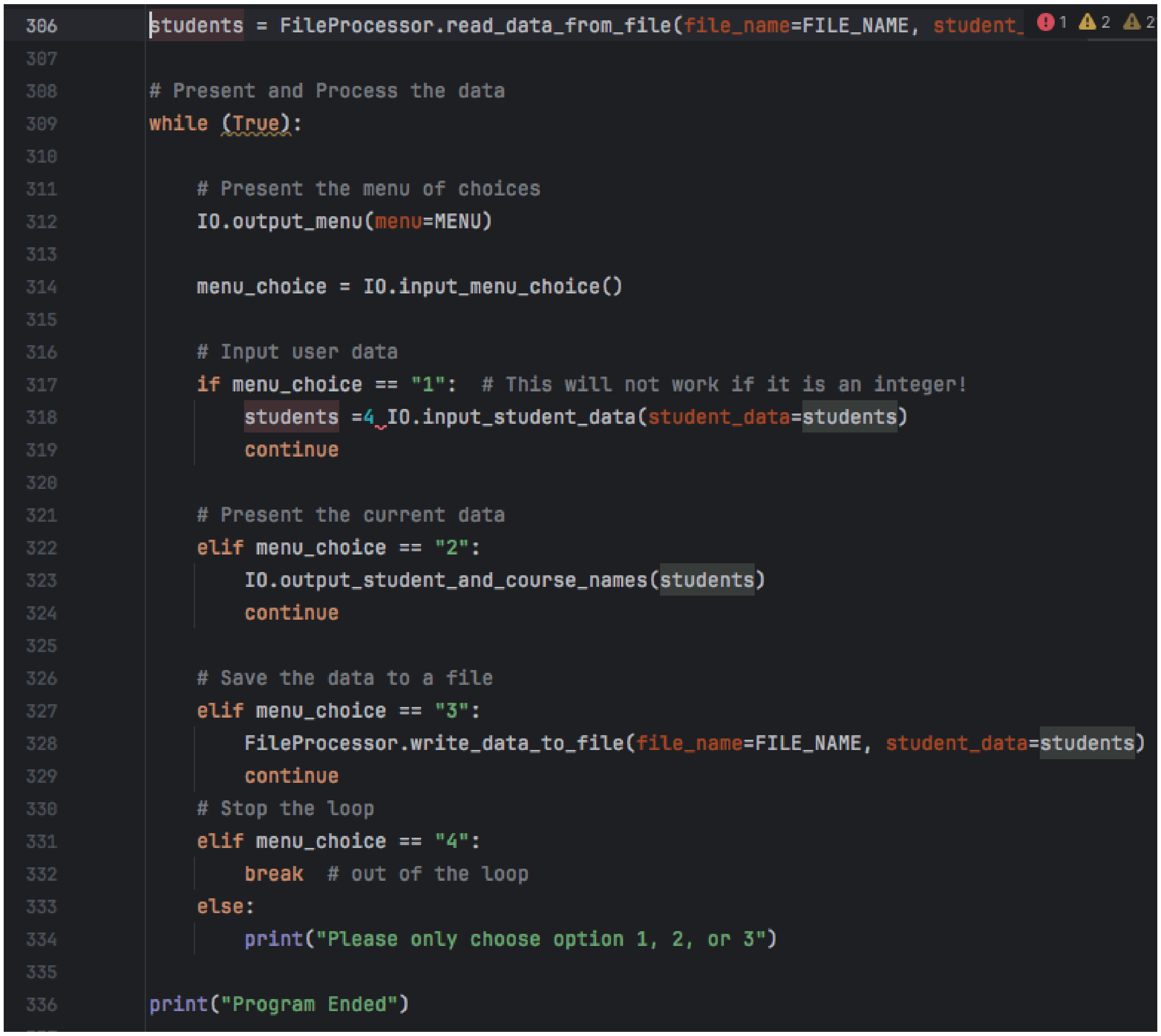
```python
    @staticmethod
    def write_data_to_file(file_name: str, student_data: list):
        """ This function writes data to a json file with data from a list of dictionary rows

        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")
            """
            converts the student list into a format that JSON file can write to
            """
            student_list = [student.__dict__ for student in student_data]
            json.dump(student_list, 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:
```

Figure 5: Class File Processor

Options

When the program starts, we want the file data to be read into a list of lists. To call the read data from the file function, I defined the fileprocessor as well as students. To be able to read class objects and call it, I defined io. The next step is to present and process the data.

Menu options are presented using true statement, and loops and commands are used for each option. I call out the menu using functions. Read data from a file is used, and it comes from the class File Processor. For option 1, student's first name, last name, and course name need to be stored from the inputs. The first names and last names must be alphabetical, so an if not statement is written to detect any input that's not alphabetical. Then the program can proceed once the correct information is inputted, and the data is appended. For option 2, I didn't change anything from the last assignment. For option 3, I wrote a line of code to save the data to a file using write to file. For option 4, I close the program. It remained the same as last assignment.

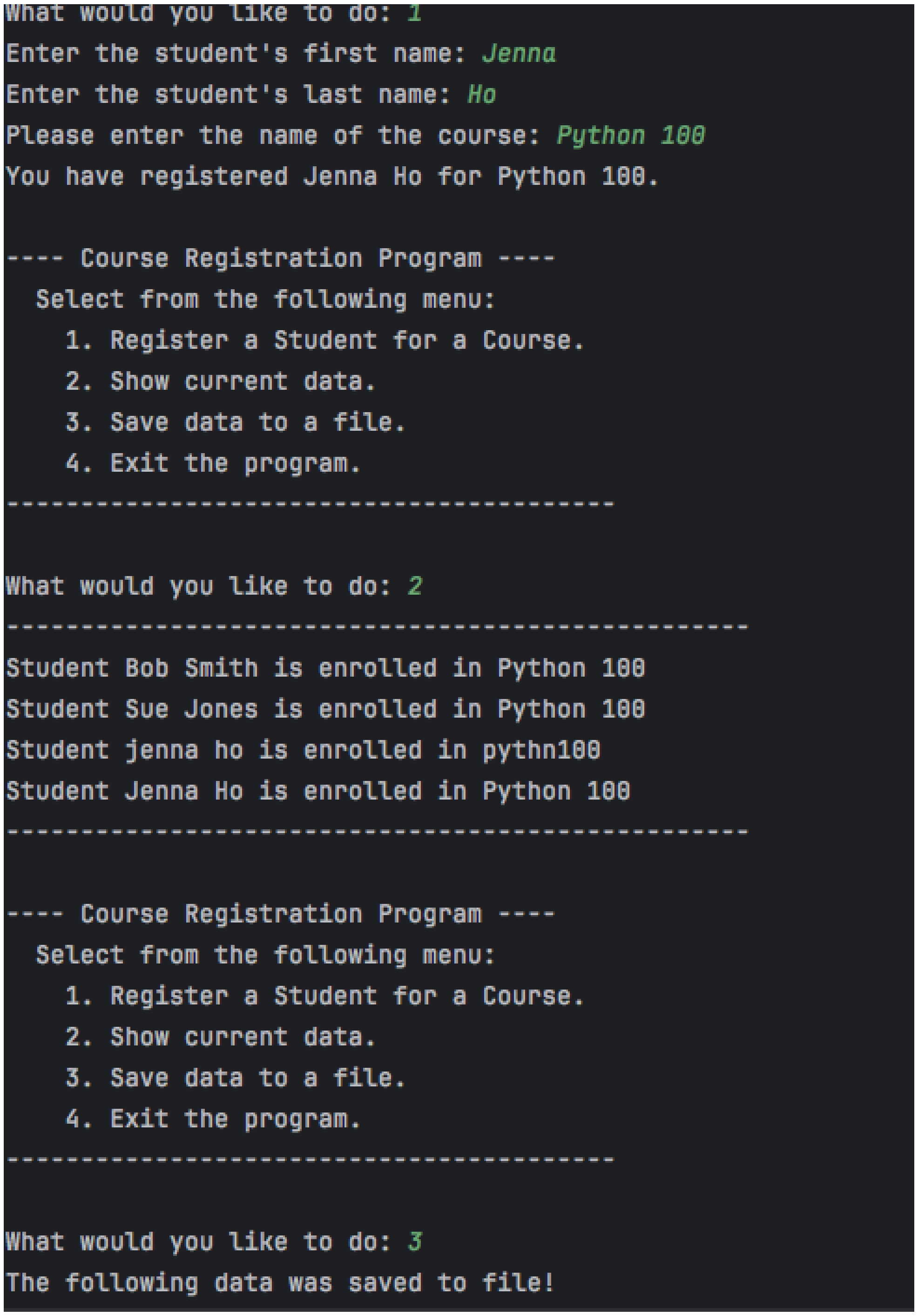


```
students = FileProcessor.read_data_from_file(file_name=FILE_NAME, student_

# 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 =4 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 6: Writing Options

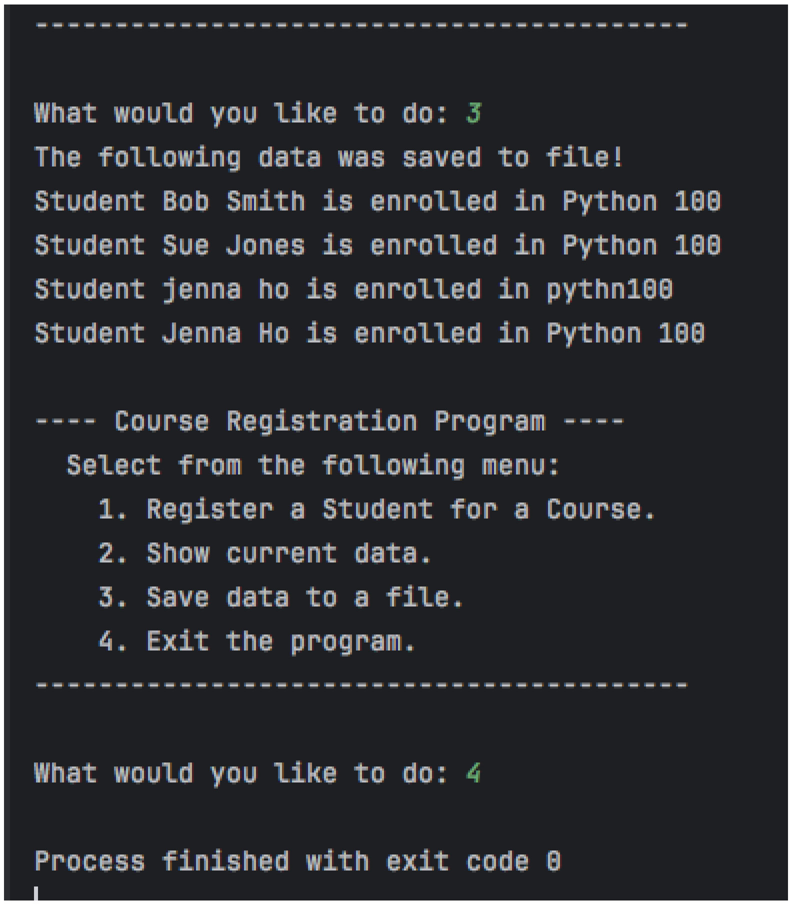
Results

My code runs successfully in both PyCharm and the terminal. I am able to write inputs when choosing option1, and the JSON file data as well as my input from option 1 is presented when choosing option2. Option 3 saves the data, and option 4 quits the program.



```
What would you like to do: 1
Enter the student's first name: Jenna
Enter the student's last name: Ho
Please enter the name of the course: Python 100
You have registered Jenna Ho 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.
-----------------------------------------


What would you like to do: 2
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student jenna ho is enrolled in pythn100
Student Jenna Ho 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.
-----------------------------------------


What would you like to do: 3
The following data was saved to file!
```

Figure 5: Results Part 1



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

What would you like to do: 3
The following data was saved to file!
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student jenna ho is enrolled in pythn100
Student Jenna Ho 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.
-----------------------------------------

What would you like to do: 4

Process finished with exit code 0
```

Figure 6: Results Part 2

Summary

In module07 assignment, I am able to include the use of functions and classes successfully, and have the program include class properties as well. I utilized class lectures, outside resources, as well as the demos from the module file to complete this assignment.