ARSENE HYACINTHE DINA NGOLLO
07/31/2024
IT FDN 110 A
Assignment 05
https://github.com/Andybowell/IntroToProg-PythonMod05

Python Program for Student Registration: Integrating Advanced Collections of Data and Error Handling

Introduction

This week's assignment focuses on creating a Python script using the PyCharm IDE to demonstrate the use of constants, variables, print statements, and string formatting for displaying a message about a student's registration for a Python course. Building on Assignment 4, this task will reinforce our understanding of basic Python syntax and concepts through practical application. It will also enhance our skills with additional elements such as while loops, programming menus, conditional logic, and data processing using dictionaries. Additionally, this assignment involves managing student registration data and ensuring data integrity through proper error handling and validation.

Creating the Program

In Assignment 5, we build upon the foundational steps from Assignment 4 by integrating collections of data and error handling. This assignment extends our knowledge to solidify our understanding of Python syntax and concepts, providing more practical applications and advanced techniques. To streamline our work, we build upon the previous assignment.

We began by defining essential constants to ensure consistent values throughout the program. The MENU constant was set to a string containing formatted menu options for user interaction. We also defined the FILE_NAME constant to specify the name of the JSON file ("Enrollments.json") where student enrollment data would be stored.

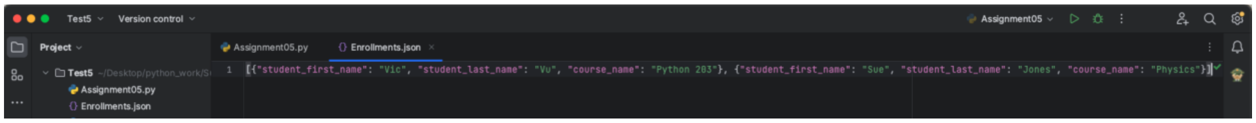
Next, we initialized variables to manage user input and data. student_first_name, student_last_name, and course_name were set as empty strings to store user input for student details. The json_data string was initialized to empty, intended to hold the combined string data separated by commas.
The file variable, initially set to None, was designated to reference the opened file. We prepared the menu_choice string to hold the user's menu selection.
Additionally, student_data was created as an empty dictionary to hold individual student data, and the students list was set up to maintain a table of student data (a list of dictionaries).

At the start of the program, we read the contents of the "Enrollments.json" file into the students list. The file was opened in read mode, and its data was extracted and transformed into a list of dictionaries. Structured error handling was implemented to manage potential errors during file reading operations, such as file not found or JSON decode errors.

We implemented a while loop to continuously present a menu to the user and process their choices until they decided to exit the program. The program offers four main menu options:

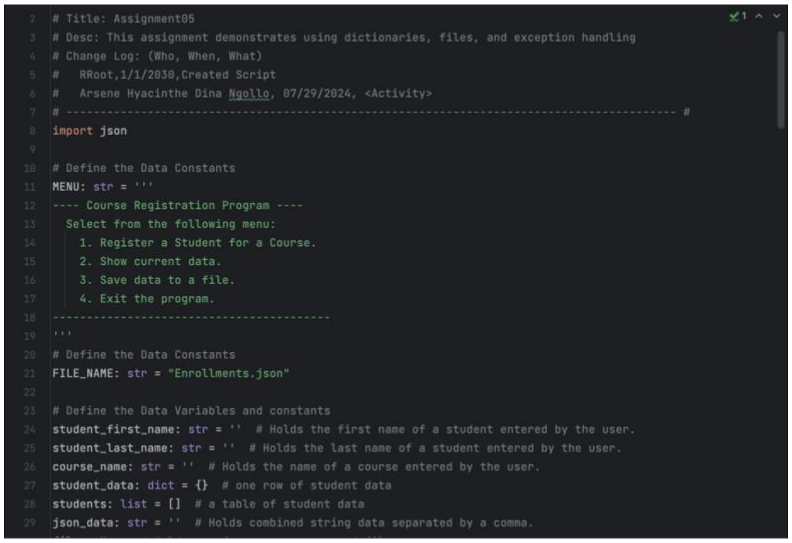
1. **Register a Student for a Course:**
   - When this option is selected, the user is prompted to enter the student's first name, last name, and course name. The input data is stored in the respective variables, formatted as a dictionary, and appended to the students list. Structured error handling was implemented to manage potential input errors, ensuring that names did not contain numbers.
2. **Show Current Data:**
   - This option displays the current list of student registrations by iterating through the students list and printing each entry. A custom message for each student was created and displayed.
3. **Save Data to a File:**
   - Selecting this option opens the "Enrollments.json" file in write mode. The contents of the students list are written to the file using the json.dump() function, ensuring each student's data is properly formatted and saved. Structured error handling was implemented to manage potential file writing errors, including checking if the data is in valid JSON format.
4. **Exit the Program:**
   - Choosing this option prints a message indicating that the program has ended and breaks the loop, terminating the program gracefully.

We tested the program by entering various student names and course names to ensure it functioned correctly. The program displayed user input accurately and saved the data to the "Enrollments.json" file as expected. Multiple registrations were conducted to confirm the program's ability to display and save data. The program was run both in PyCharm and from the console or terminal to ensure it worked correctly in different environments.



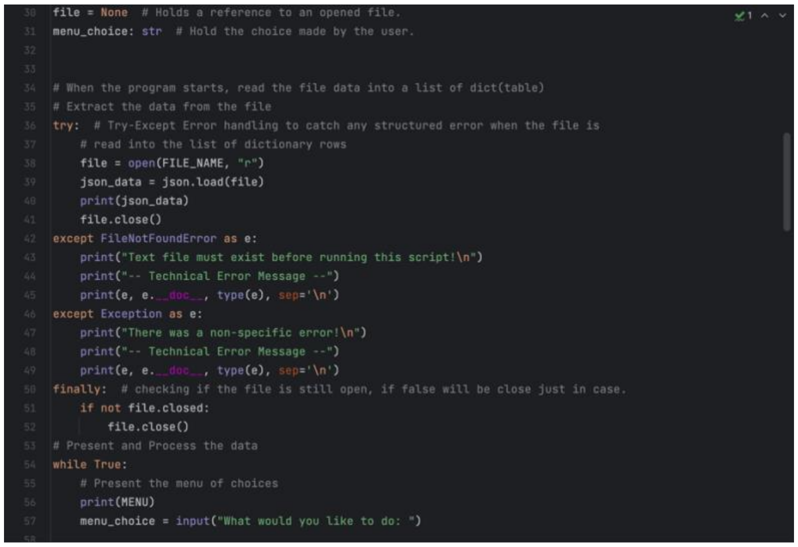
```
[{"student_first_name": "Vic", "student_last_name": "Vu", "course_name": "Python 203"}, {"student_first_name": "Sue", "student_last_name": "Jones", "course_name": "Physics"}]
```

Figure 1-Enrollments.json



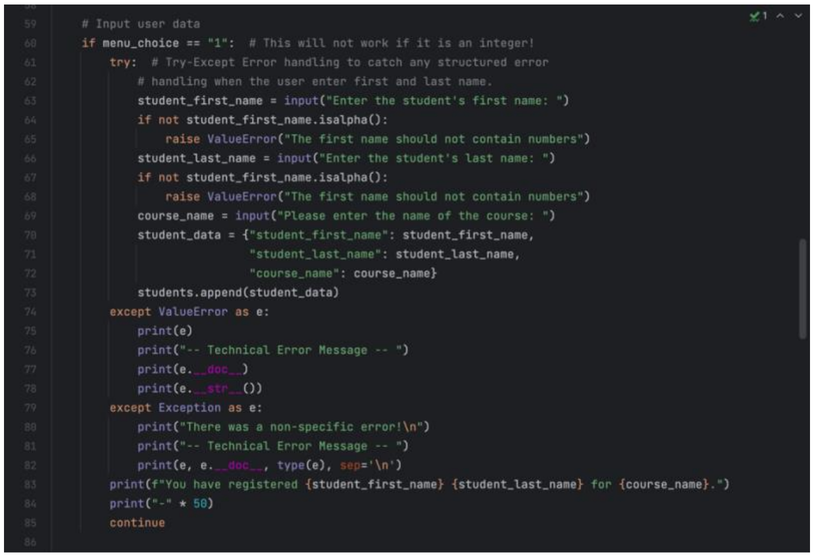
```python
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   RRoot,1/1/2030,Created Script
#   Arsene Hyacinthe Dina Ngollo, 07/29/2024, <Activity>
# ------------------------------------------------------------------------------------------ #
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"

# 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
json_data: str = ''  # Holds combined string data separated by a comma.
```

Figure 2-Importing and defining constants and variables



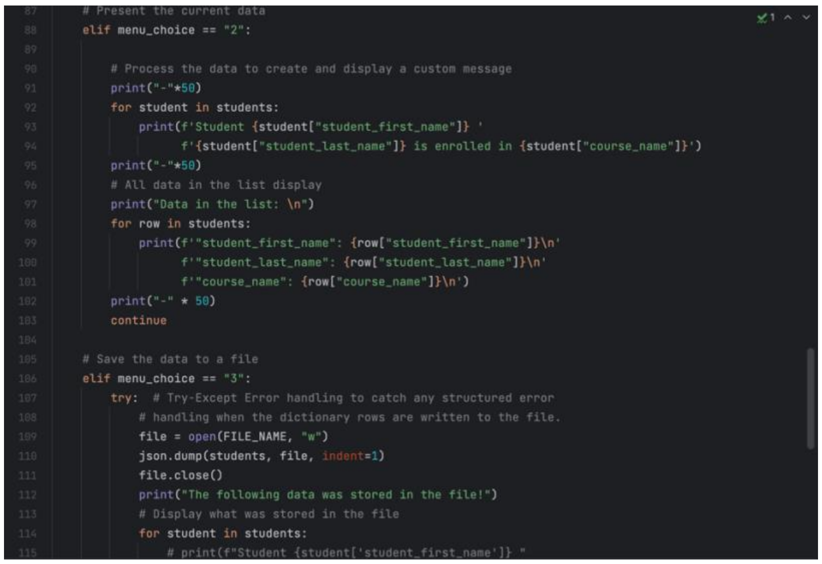
```python
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.


# When the program starts, read the file data into a list of dict(table)
# Extract the data from the file
try:  # Try-Except Error handling to catch any structured error when the file is
    # read into the list of dictionary rows
    file = open(FILE_NAME, "r")
    json_data = json.load(file)
    print(json_data)
    file.close()
except FileNotFoundError as e:
    print("Text file must exist before running this script!\n")
    print("-- Technical Error Message --")
    print(e, e.__doc__, type(e), sep='\n')
except Exception as e:
    print("There was a non-specific error!\n")
    print("-- Technical Error Message --")
    print(e, e.__doc__, type(e), sep='\n')
finally:  # checking if the file is still open, if false will be close just in case.
    if not file.closed:
        file.close()
# Present and Process the data
while True:
    # Present the menu of choices
    print(MENU)
    menu_choice = input("What would you like to do: ")
```

Figure 3-Opening json file in reading mode



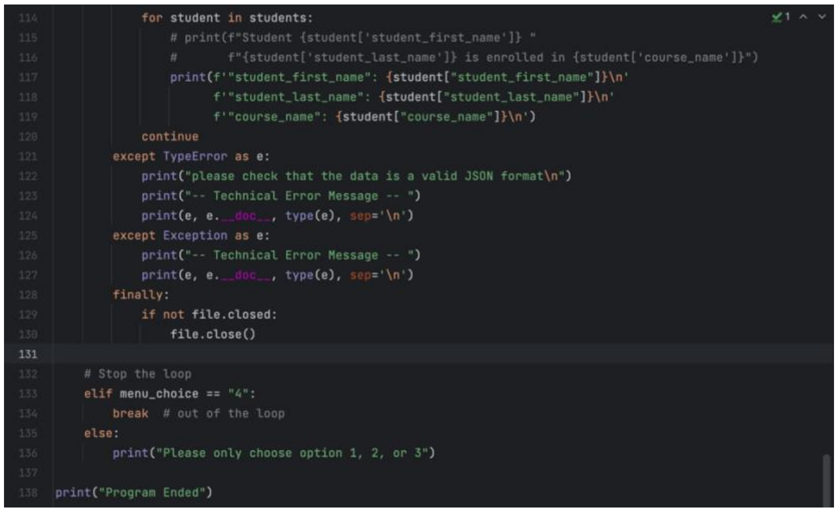
```python
    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        try:  # Try-Except Error handling to catch any structured error
            # handling when the user enter first and last name.
            student_first_name = input("Enter the student's first 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_first_name.isalpha():
                raise ValueError("The first name should not contain numbers")
            course_name = input("Please enter the name of the course: ")
            student_data = {"student_first_name": student_first_name,
                            "student_last_name": student_last_name,
                            "course_name": course_name}
            students.append(student_data)
        except ValueError as e:
            print(e)
            print("-- Technical Error Message -- ")
            print(e.__doc__)
            print(e.__str__())
        except Exception as e:
            print("There was a non-specific error!\n")
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep='\n')
        print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
        print("-" * 50)
        continue

```

Figure 4. Menu Choice Code 1



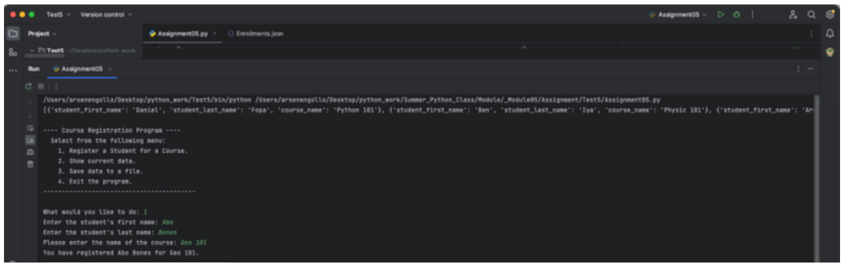
```python
    # Present the current data
    elif menu_choice == "2":

        # Process the data to create and display a custom message
        print("-"*50)
        for student in students:
            print(f'Student {student["student_first_name"]} '
                  f'{student["student_last_name"]} is enrolled in {student["course_name"]}')
        print("-"*50)
        # All data in the list display
        print("Data in the list: \n")
        for row in students:
            print(f'"student_first_name": {row["student_first_name"]}\n'
                  f'"student_last_name": {row["student_last_name"]}\n'
                  f'"course_name": {row["course_name"]}\n')
        print("-" * 50)
        continue

    # Save the data to a file
    elif menu_choice == "3":
        try:  # Try-Except Error handling to catch any structured error
            # handling when the dictionary rows are written to the file.
            file = open(FILE_NAME, "w")
            json.dump(students, file, indent=1)
            file.close()
            print("The following data was stored in the file!")
            # Display what was stored in the file
            for student in students:
                # print(f"Student {student['student_first_name']} "
```

Figure 5. Menu Choice Code 2 and 3



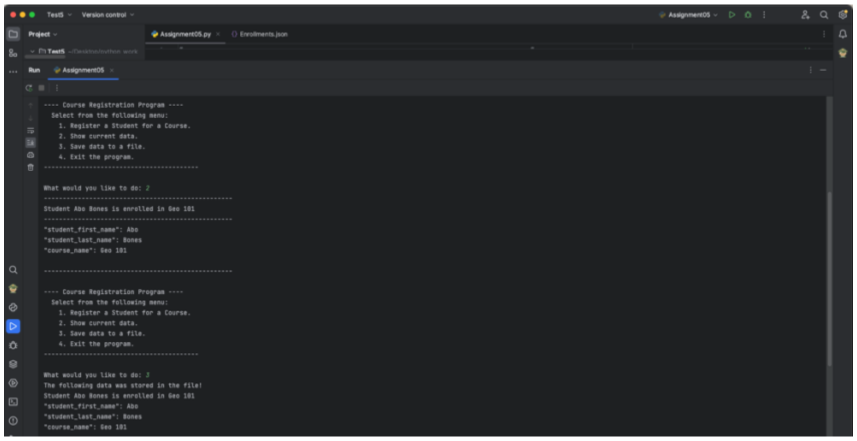
```python
            for student in students:
                # print(f"Student {student['student_first_name']} "
                #       f"{student['student_last_name']} is enrolled in {student['course_name']}")
                print(f'"student_first_name": {student["student_first_name"]}\n'
                      f'"student_last_name": {student["student_last_name"]}\n'
                      f'"course_name": {student["course_name"]}\n')
            continue
        except TypeError as e:
            print("please check that the data is a valid JSON format\n")
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep='\n')
        except Exception as e:
            print("-- Technical Error Message -- ")
            print(e, e.__doc__, type(e), sep='\n')
        finally:
            if not file.closed:
                file.close()

    # 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. Menu Choice Code 4



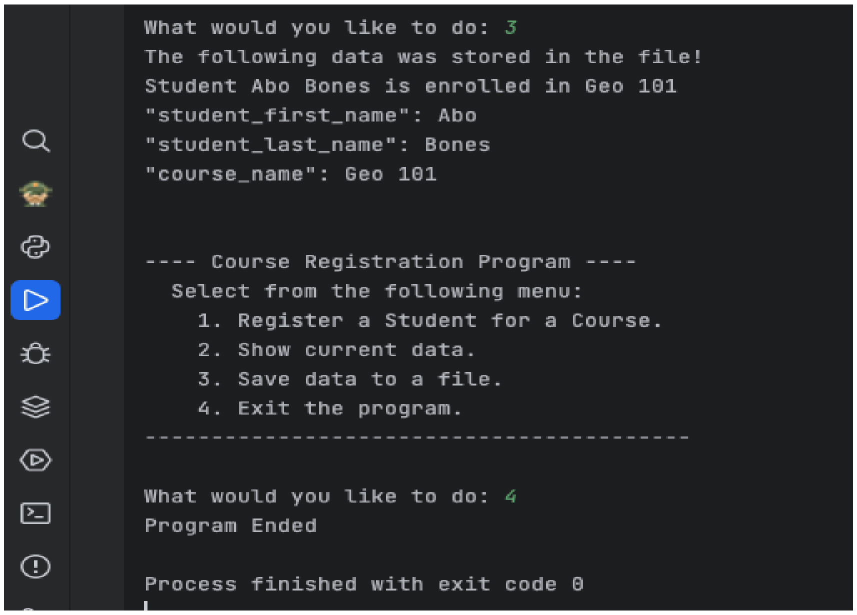
```
/Users/arsenengollo/Desktop/python_work/Test5/bin/python /Users/arsenengollo/Desktop/python_work/Summer_Python_Class/Module/_Module05/Assignment/Test5/Assignment05.py
[{'student_first_name': 'Daniel', 'student_last_name': 'Fopa', 'course_name': 'Python 101'}, {'student_first_name': 'Ben', 'student_last_name': 'Iya', 'course_name': 'Physic 101'}, {'student_first_name': 'Ar

---- 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: 1
Enter the student's first name: Abo
Enter the student's last name: Bones
Please enter the name of the course: Geo 101
You have registered Abo Bones for Geo 101.
```

Figure 7. Testing on PyCharm IDE



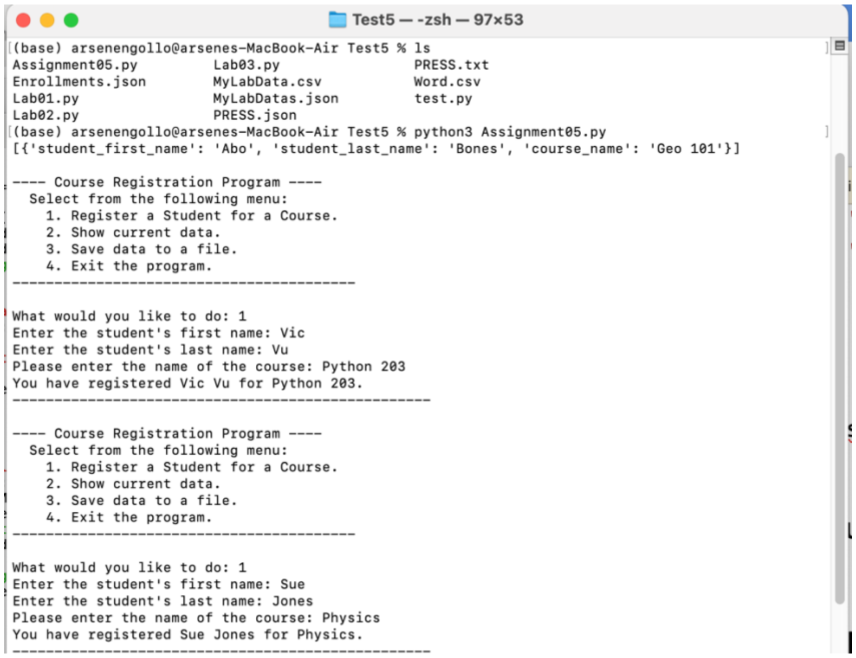
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------

What would you like to do: 2
--------------------------------------------------
Student Abo Bones is enrolled in Geo 101
--------------------------------------------------
"student_first_name": Abo
"student_last_name": Bones
"course_name": Geo 101

--------------------------------------------------

---- 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 stored in the file!
Student Abo Bones is enrolled in Geo 101
"student_first_name": Abo
"student_last_name": Bones
"course_name": Geo 101
```

Figure 8. Testing on PyCharm IDE (suite)



```
What would you like to do: 3
The following data was stored in the file!
Student Abo Bones is enrolled in Geo 101
"student_first_name": Abo
"student_last_name": Bones
"course_name": Geo 101


---- 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
Program Ended

Process finished with exit code 0
```

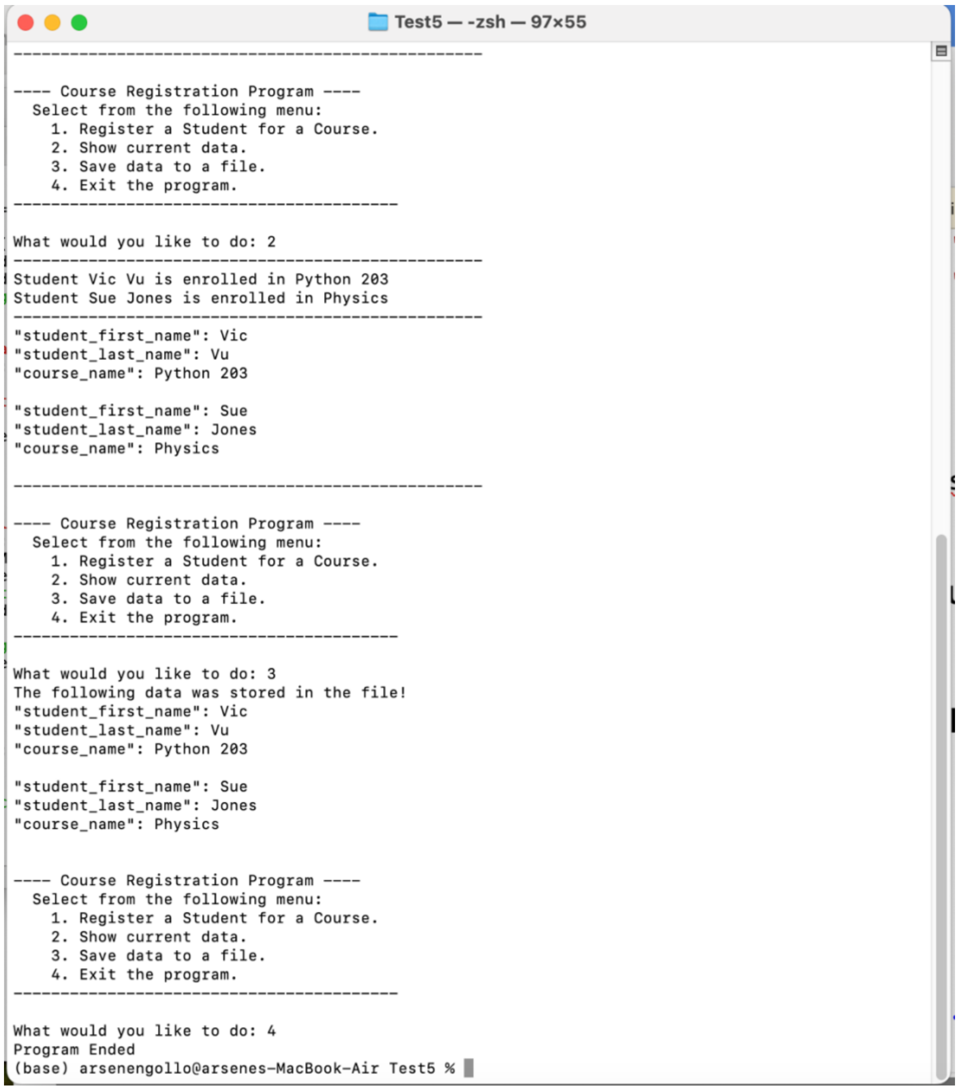
Figure 9. Testing on PyCharm IDE (suite)



Test5 — -zsh — 97×53

```
(base) arsenengollo@arsenes-MacBook-Air Test5 % ls
Assignment05.py         Lab03.py                PRESS.txt
Enrollments.json        MyLabData.csv           Word.csv
Lab01.py                MyLabDatas.json         test.py
Lab02.py                PRESS.json
(base) arsenengollo@arsenes-MacBook-Air Test5 % python3 Assignment05.py
[{'student_first_name': 'Abo', 'student_last_name': 'Bones', 'course_name': 'Geo 101'}]

---- 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: 1
Enter the student's first name: Vic
Enter the student's last name: Vu
Please enter the name of the course: Python 203
You have registered Vic Vu for Python 203.
--------------------------------------------------

---- 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: 1
Enter the student's first name: Sue
Enter the student's last name: Jones
Please enter the name of the course: Physics
You have registered Sue Jones for Physics.
--------------------------------------------------
```

Figure 10. Testing on terminals



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

---- 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 Vic Vu is enrolled in Python 203
Student Sue Jones is enrolled in Physics
--------------------------------------------------
"student_first_name": Vic
"student_last_name": Vu
"course_name": Python 203

"student_first_name": Sue
"student_last_name": Jones
"course_name": Physics

--------------------------------------------------

---- 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 stored in the file!
"student_first_name": Vic
"student_last_name": Vu
"course_name": Python 203

"student_first_name": Sue
"student_last_name": Jones
"course_name": Physics


---- 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
Program Ended
(base) arsenengollo@arsenes-MacBook-Air Test5 %
```

Figure 11. Testing on terminals (suite)

Summary

In this Python assignment, we focused on creating a script using the PyCharm IDE to demonstrate constants, variables, print statements, and string formatting. Building upon previous assignments, we enhanced the task by integrating data processing techniques using lists and dictionaries. Additionally, we employed new concepts to prevent errors, including:

- **Handling FileNotFoundError:** Managed scenarios where the file might not exist.
- **Managing json.JSONDecodeError:** Addressed cases where the file content is not in valid JSON format.

- **Ensuring Valid User Inputs:** Checked that the user provides valid first names, last names, and course names, verifying that inputs are not empty strings and do not contain invalid characters.
- **Implementing ValueError:** Caught invalid inputs, such as names containing numbers.
- **Managing Exceptions During File Writing:** Ensured data is saved correctly by handling exceptions that might occur during file writing operations.
- **Implementing TypeError:** Verified that the data being written is in valid JSON format.

This exercise reinforced our understanding of Python's syntax and concepts, and it prepared us for more complex tasks in future assignments.