Mariusz Sokol
Feb, 05, 2025
Writing a paper
Assingnment05

https://github.com/Mariusz-uw

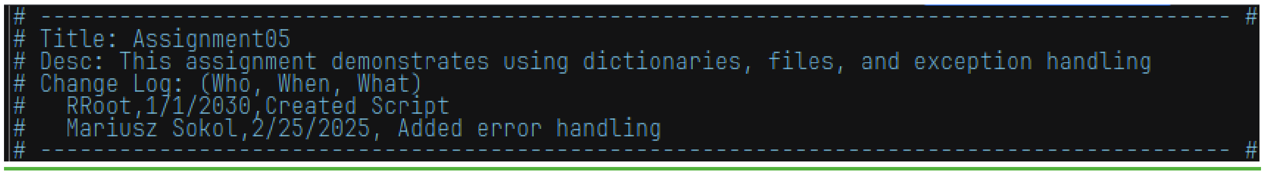
Introduction

This program will explore some techniques of structured error handling a program.

Python like other programming languages provides tools that make it easier for the programmer to prompt program users on any occurring errors while using/running the program. Also how the programmer strategically can build the program to make it easier for the user to overcome some of the errors that the user can experience when using a program.

Script

This script shows this files title, it writes a description of what the program demonstrates.



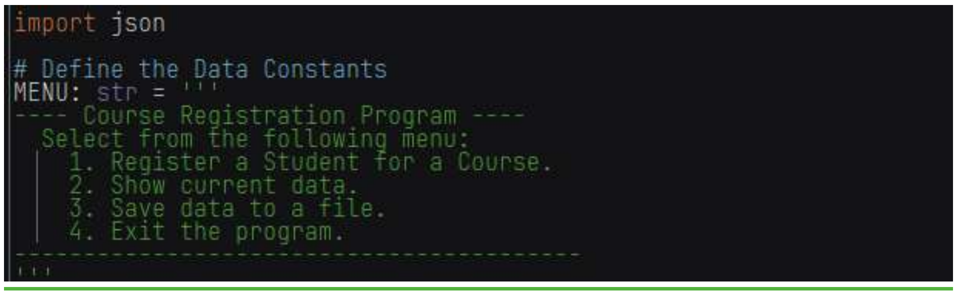
```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
# Change Log: (Who, When, What)
#   RRoot,1/1/2030,Created Script
#   Mariusz Sokol,2/25/2025, Added error handling
# ------------------------------------------------------------------------------------------ #
```

Figure: 1 Presenting the programs elementary information.

Menu option

First a "json" module is imported that will make it possible to handle data in a particular format.

Then a menu of four options is created for the user to interact with as will be presented later in this document.



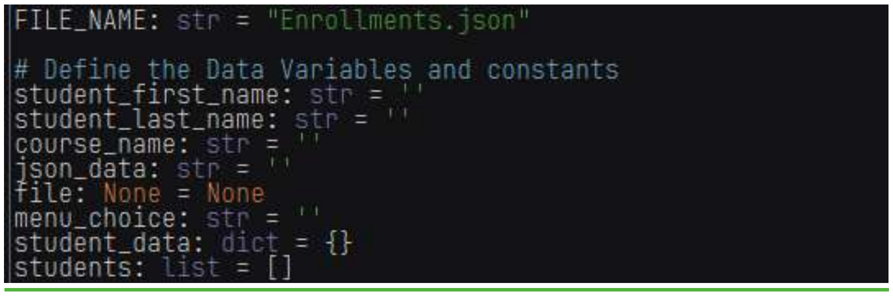
```
import json

# 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.
-----------------------------------------
'''
```

Figure: 2 Library import and menu construct to interact with the user in later code sections.

Variables

This illustration shows that a file is created with a json extension and assigned to a constant.

Then there are some empty variables for use later in the program.

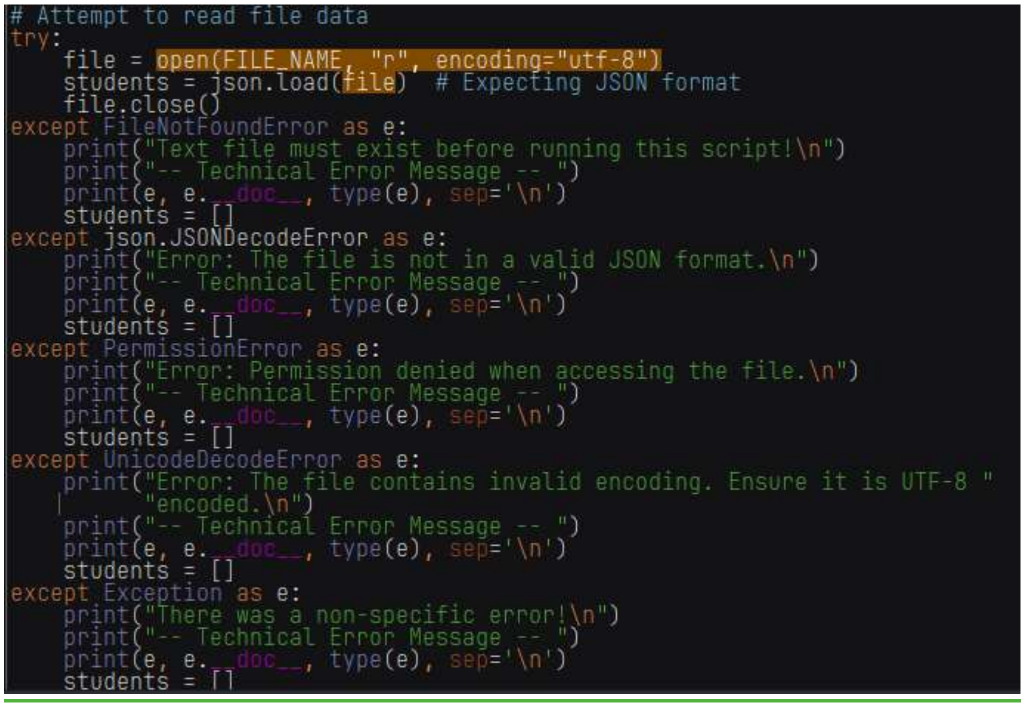


```
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables and constants
student_first_name: str = ''
student_last_name: str = ''
course_name: str = ''
json_data: str = ''
file: None = None
menu_choice: str = ''
student_data: dict = {}
students: list = []
```

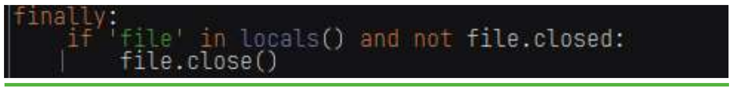
Figure: 3 Declaring variables of various sorts crucial for the program to run properly.

Reading file data and Structured Error Handling

This portion of the code is responsible for reading the file data into a list of dictionaries while implementing structured error handling. First “exception” catches error occurrences relating to opening and reading the file. Second “exception” handles cases where file is present but contains corrupt or improperly formatted JSON data. Third “exception” is raised when permission to read the file is lacking e.g., file locked or admin privileges not present. If the json file contains characters that are not valid in UTF-8 encoding, then the fourth “exception” gets raised. The last “except” block handles other error occurrences that might get raised. The “students = []” at the end of each except code block resets the variable to an empty list. The “finally” code block makes sure that the file gets properly closed even if an exception occurs.



```
# Attempt to read file data
try:
    file = open(FILE_NAME, "r", encoding="utf-8")
    students = json.load(file)  # Expecting JSON format
    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')
    students = []
except json.JSONDecodeError as e:
    print("Error: The file is not in a valid JSON format.\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
    students = []
except PermissionError as e:
    print("Error: Permission denied when accessing the file.\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
    students = []
except UnicodeDecodeError as e:
    print("Error: The file contains invalid encoding. Ensure it is UTF-8 "
          "encoded.\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
    students = []
except Exception as e:
    print("There was a non-specific error!\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
    students = []
```



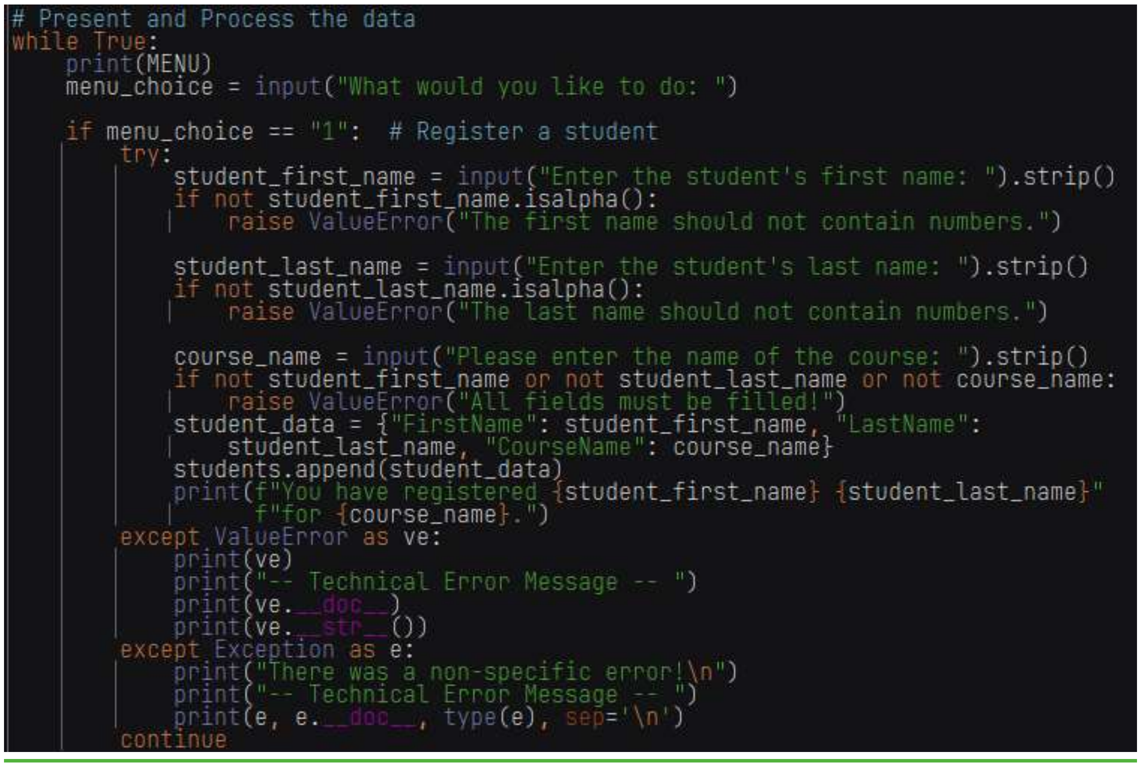
```
finally:
    if 'file' in locals() and not file.closed:
        file.close()
```

Figure: 4 The part of code most prone to receiving an error, hence the amount of "except" blocks required.

<u>First option input</u>

In the code section below menu options (MENU) is first displayed. The user gets then prompted to input a choice in form of an integer (1-4). The loop keeps running until the user selects option 4 which exits the program. The following code prompts the user for first name, last name, and course name. Student first name and last name are checked for alphabetic characters and that no fields are left empty. The input is then stored into a dictionary "student_data", then the dictionary is appended to the students list which then stores all student input.

The "exception" at line 91 occurs when an operation receives an argument of invalid value. The "exception" at line 96 triggers in other cases.

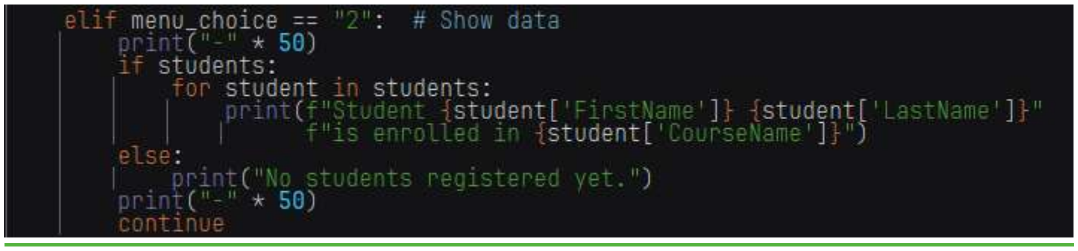


```python
# Present and Process the data
while True:
    print(MENU)
    menu_choice = input("What would you like to do: ")

    if menu_choice == "1":  # Register a student
        try:
            student_first_name = input("Enter the student's first name: ").strip()
            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: ").strip()
            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: ").strip()
            if not student_first_name or not student_last_name or not course_name:
                raise ValueError("All fields must be filled!")
            student_data = {"FirstName": student_first_name, "LastName":
                student_last_name, "CourseName": course_name}
            students.append(student_data)
            print(f"You have registered {student_first_name} {student_last_name}"
                  f"for {course_name}.")
        except ValueError as ve:
            print(ve)
            print("-- Technical Error Message -- ")
            print(ve.__doc__)
            print(ve.__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')
        continue
```

Figure: 5 The part of code responsible for running when input option 1 is chosen.

Second option input

This code section runs when user selects option “2” from the menu. The “if students:” line, checks if the list contains any data to display. The f-string code is then run to display student’s data. In case no data present in “students” the else code line is run.



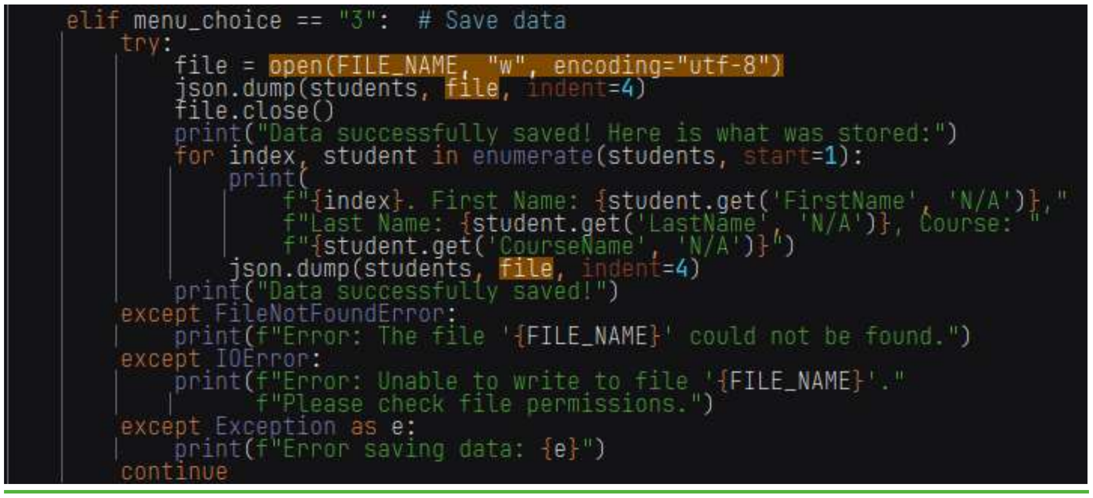
```python
    elif menu_choice == "2":  # Show data
        print("-" * 50)
        if students:
            for student in students:
                print(f"Student {student['FirstName']} {student['LastName']}"
                      f"is enrolled in {student['CourseName']}")
        else:
            print("No students registered yet.")
        print("-" * 50)
        continue
```

Figure: 6 By choosing input option 2 the following code executes.

Third option input

Choosing input option “3”, code snippet below executes. A file is opened in write mode. Json data is transferred into the file, then the file is closed. Student data is then printed by using enumerate function. The “exception - file not found” is triggered in case of missing file.

I/O “exception” activates if the disk is full or file locked. Last “exception” activates in case of other unexpected errors.

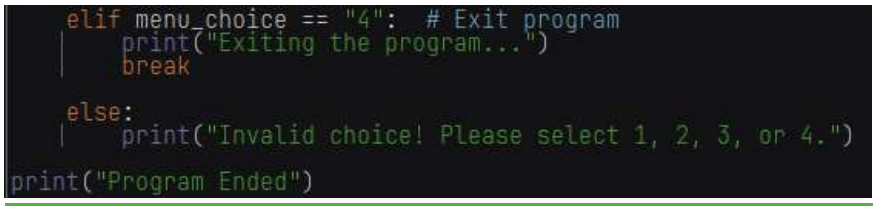


```
    elif menu_choice == "3":  # Save data
        try:
            file = open(FILE_NAME, "w", encoding="utf-8")
            json.dump(students, file, indent=4)
            file.close()
            print("Data successfully saved! Here is what was stored:")
            for index, student in enumerate(students, start=1):
                print(
                    f"{index}. First Name: {student.get('FirstName', 'N/A')},"
                    f"Last Name: {student.get('LastName', 'N/A')}, Course: "
                    f"{student.get('CourseName', 'N/A')}")
                json.dump(students, file, indent=4)
            print("Data successfully saved!")
        except FileNotFoundError:
            print(f"Error: The file '{FILE_NAME}' could not be found.")
        except IOError:
            print(f"Error: Unable to write to file '{FILE_NAME}'."
                  f"Please check file permissions.")
        except Exception as e:
            print(f"Error saving data: {e}")
        continue
```

Figure: 7 Above illustrated the part of code that executes when input option 3 is chosen.

Fourth option input

Finally If option input “4” is chosen the program breaks from the loop and the program stops.

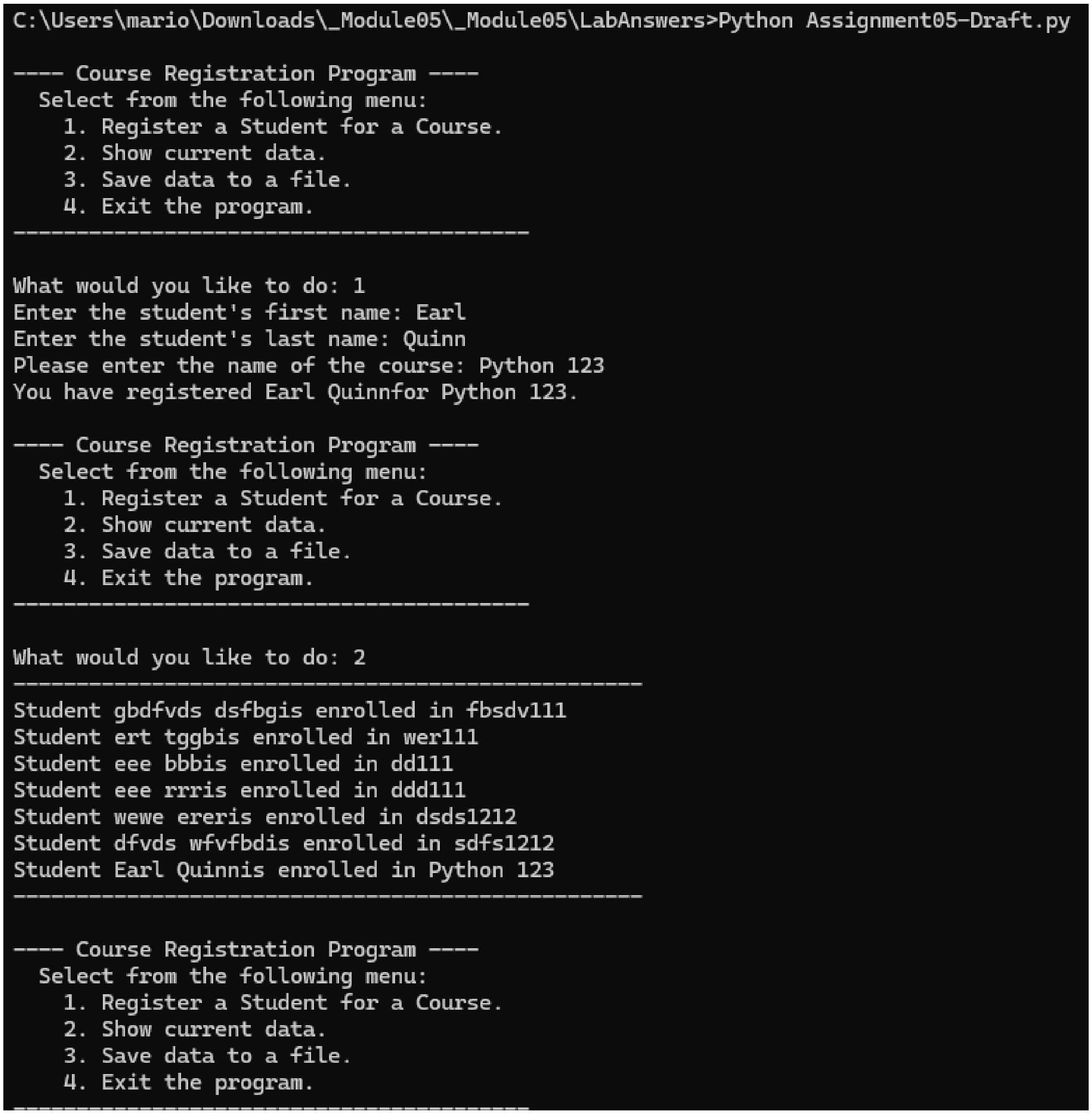


```
    elif menu_choice == "4":  # Exit program
        print("Exiting the program...")
        break

    else:
        print("Invalid choice! Please select 1, 2, 3, or 4.")

print("Program Ended")
```

Figure: 8 Last part of the program the code that executes when choosing input option 4.

The entire program run in Command Prompt

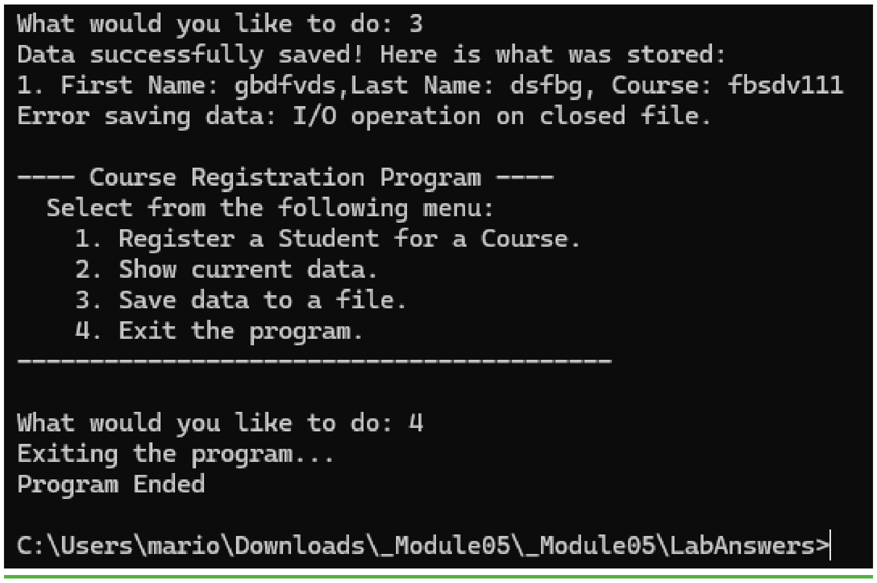
Here is the same program run in Command Prompt



```
C:\Users\mario\Downloads\_Module05\_Module05\LabAnswers>Python Assignment05-Draft.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.
-----------------------------------------

What would you like to do: 1
Enter the student's first name: Earl
Enter the student's last name: Quinn
Please enter the name of the course: Python 123
You have registered Earl Quinnfor Python 123.

---- 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 gbdfvds dsfbgis enrolled in fbsdv111
Student ert tggbis enrolled in wer111
Student eee bbbis enrolled in dd111
Student eee rrris enrolled in ddd111
Student wewe ereris enrolled in dsds1212
Student dfvds wfvfbdis enrolled in sdfs1212
Student Earl Quinnis enrolled in Python 123
--------------------------------------------------

---- 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: 9 Illustration showing the entire program run in Command Prompt.

Rest of the program from Command Prompt



```
What would you like to do: 3
Data successfully saved! Here is what was stored:
1. First Name: gbdfvds,Last Name: dsfbg, Course: fbsdv111
Error saving data: I/O operation on closed file.

---- 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
Exiting the program...
Program Ended

C:\Users\mario\Downloads\_Module05\_Module05\LabAnswers>
```

Figure: 10 Illustration above shows the second part of a program run in Command prompt.

Summary

This presentation of a program shows how valuable and user friendly it can be for the user to figure out what the error is about when encountering one while using the program.