Jesse Huminski

11/04/2024

IT FDN 100 A

Assignment 04

Accessing Data from Files

Introduction

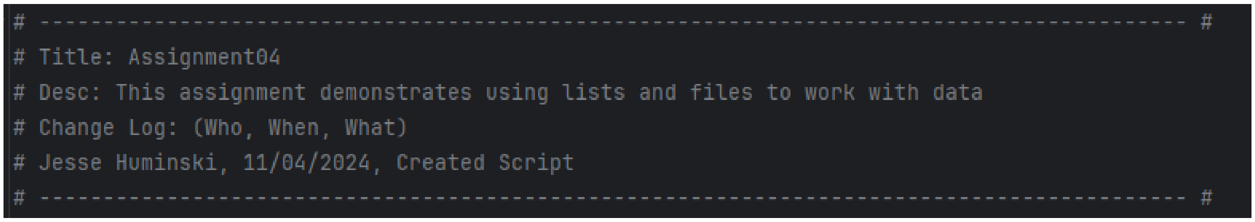
Module 04 continued to build upon the knowledge of setting variables and dove deeper into using Python code to write to files, the concept of data collections and how to determine the type of collection that our code should utilize. We explored data persistence within our code, allowing us to access data input from previous iterations of the code after the program has been closed. This module also introduced us to debugging and how we can identify the cause of errors and fix unexpected behavior from our code.

Our assignment this week required us to create a program that will accept input from the user and save it to a file that we can later access even after the program has been closed. This program should also be able to display previously collected data and display it in a way that is readable for users.

Please Note: I have included a file called “Enrollments.CSV” along in the zipped directory as the code would not run properly if this file does not exist.

Creating the Program

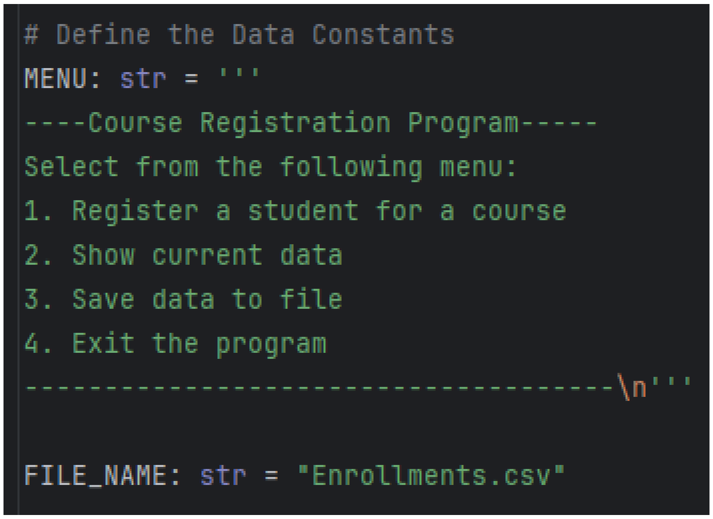
As always, we started off with the header at the top of the code detailing the title and description of the code, along with the log of significant changes. (Image 1.1)



```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment04
# Desc: This assignment demonstrates using lists and files to work with data
# Change Log: (Who, When, What)
# Jesse Huminski, 11/04/2024, Created Script
# ------------------------------------------------------------------------------------------ #
```

Image 1.1: The header of the code.

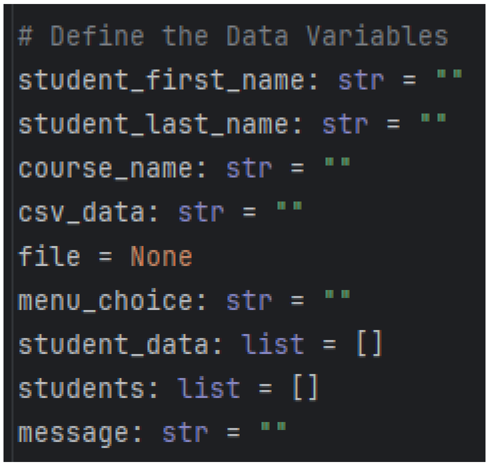
From there, we defined our constants, the **MENU** string and the **FILENAME** string. These are called throughout the code and will not change. (Image 1.2)



```
# 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 file
4. Exit the program
-------------------------------------\n'''

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

Image 1.2: The data constants, including the menu of options for the user

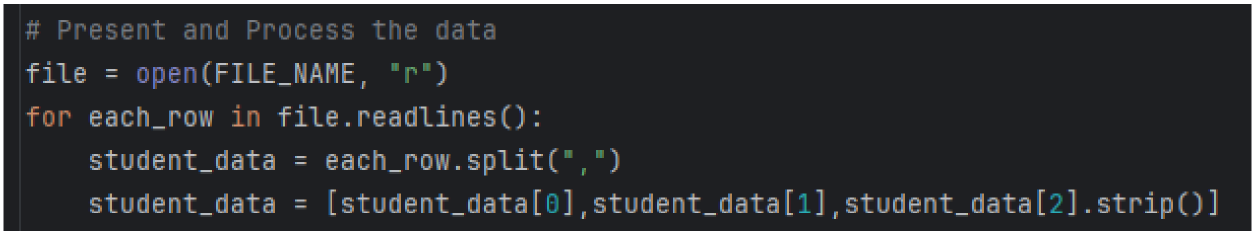
The data variables are defined at the beginning of the code as well to inform other potential collaborators what type of data types we should be expecting throughout the code. (Image 1.3)



```
# Define the Data Variables
student_first_name: str = ""
student_last_name: str = ""
course_name: str = ""
csv_data: str = ""
file = None
menu_choice: str = ""
student_data: list = []
students: list = []
message: str = ""
```

Image 1.3: The data variables including data type expectations

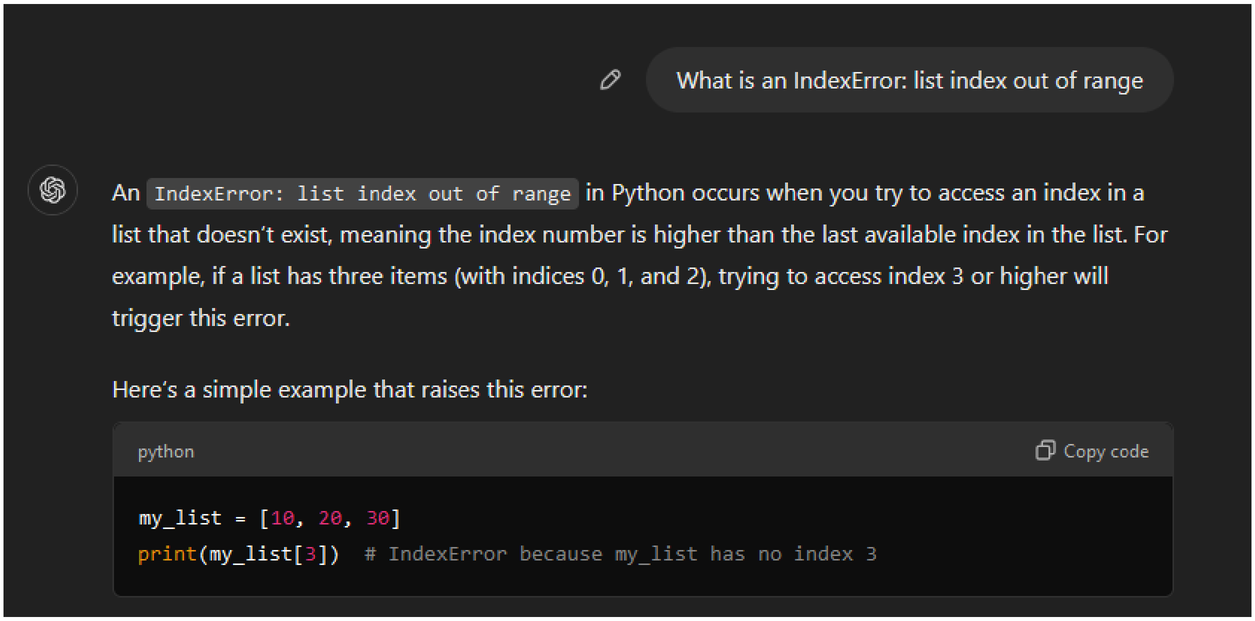
The code calls to the file “Enrollments.CSV” and opens it in read-mode using the (“r”) function. This allows for the code to read the data that is stored in the file. I set a “for” loop up to iterate through each line of the file using **file.readlines()**. The for loop allows the code to iterate through each line and store it in “each_row.” I then used the **.split(“,”)** function to split the data at the comma and create a list of values for the student’s previously inputted data. The data is then stored in a list assigned to the variable **“student_data.”** in which I used indices to delineate the first name (index [0]), the last name (index [1]), and the course name (index [2]). (Image 1.4)



```
# Present and Process the data
file = open(FILE_NAME, "r")
for each_row in file.readlines():
    student_data = each_row.split(",")
    student_data = [student_data[0],student_data[1],student_data[2].strip()]
```

Image 1.4: Opening and reading data from the file.

At this point, I had some issue here because there was a bug in the file “Enrollments.CSV” that was causing and “IndexError: list index out of range.” I could not figure out through the debugging console as everything appeared fine and all of the values were being assigned appropriately. I ended up asking ChatGPT a few questions about this issue as I was uncertain about some things. I asked ChatGPT “What is an IndexError: list index out of range?” I wanted to ensure that my understanding of the error was correct in that this error means that my code is pointing to an index that doesn’t exist. (Image 1.5)



What is an IndexError: list index out of range

An `IndexError: list index out of range` in Python occurs when you try to access an index in a list that doesn’t exist, meaning the index number is higher than the last available index in the list. For example, if a list has three items (with indices 0, 1, and 2), trying to access index 3 or higher will trigger this error.

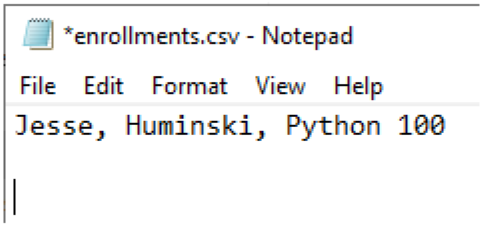
Here’s a simple example that raises this error:

```python
my_list = [10, 20, 30]
print(my_list[3])  # IndexError because my_list has no index 3
```

Image 1.5: My inquiry to ChatGPT about an IndexError

I had assumed that sentiment was the issue. But, I was uncertain about the true nature of the problem.

I was a bit confused as to what the cause of this error was for quite a while. I was working through multiple aspects of troubleshooting before I found the root cause. At some point throughout testing the code, two (2) new line characters ("\n") had been printed to the CSV file. (Image 1.6)



```
Jesse, Huminski, Python 100

```

Image 1.6: Opening the CSV file to observe that there are two "new lines."

I had suspected that this was the root of the problem but wasn't entirely sure. So, I had to ask ChatGPT the following question:

Q. "Does a \n count as a character in an index?"

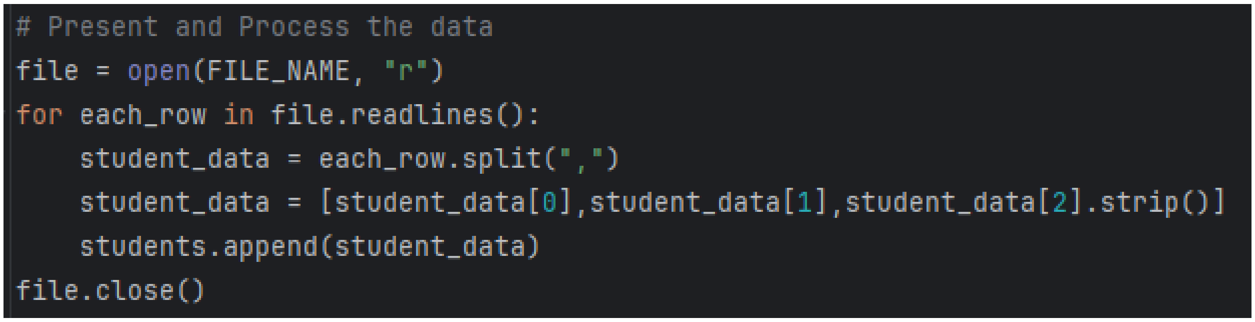
ChatGPT: "Yes, the newline character \n does count as a character in an index in Python strings. It takes up one position in the string, just like any other character. So, if a string contains a newline character, it will be included in the string's length and can be accessed by its index."

I then followed up with:

Q. Would two new line characters in a CSV file cause this line of code (insert pasted code) to lead to an IndexError?

ChatGPT: Yes, the CSV file has an extra empty line at the end, which is causing the IndexError. When the code tries to split this empty line, it doesn't produce three elements, leading to an "index out of range" error.

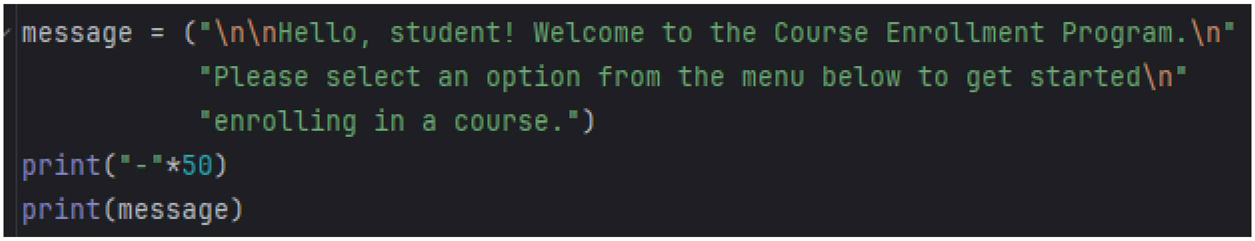
After quite some time, I finally found the cause of the problem. After deleting the extra new line character from the CSV file, I tested the program again and it seemed to be working as intended. I then appended **"student_data"** to the list variable **"students"** and closed the file. Here is an image of the full code. (Image 1.7)



```
# Present and Process the data
file = open(FILE_NAME, "r")
for each_row in file.readlines():
    student_data = each_row.split(",")
    student_data = [student_data[0],student_data[1],student_data[2].strip()]
    students.append(student_data)
file.close()
```

Image 1.7: The full code reading data from the CSV file and storing it into "students."

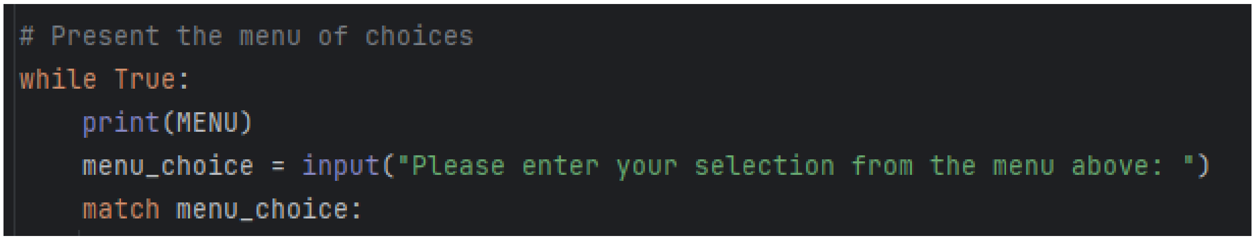
I wanted the program to display a message to the user welcoming them to the program and giving them an idea as to what the program is expecting from them. So, I printed this message (image 1.8).



```
message = ("\n\nHello, student! Welcome to the Course Enrollment Program.\n"
           "Please select an option from the menu below to get started\n"
           "enrolling in a course.")
print("-"*50)
print(message)
```

Image 1.8: Welcome message detailing the program's expectations.

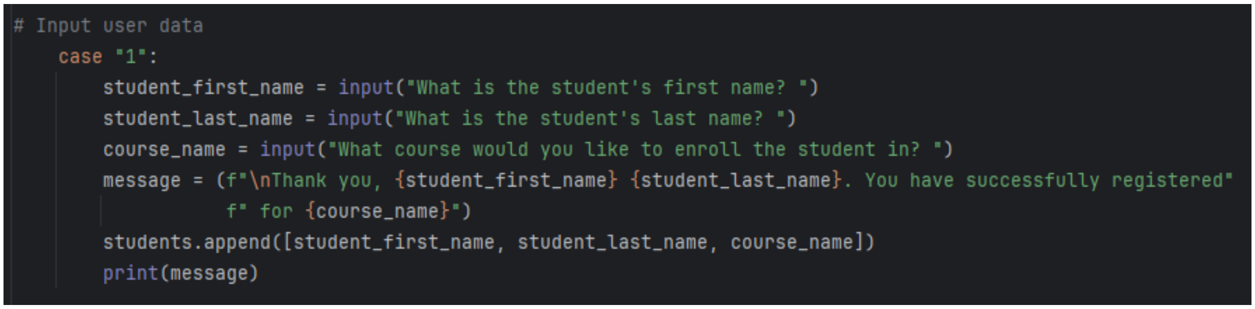
I started a "while" loop at this point of the code to ensure that the program with continue to bring us to this point until we decided to exit the program on our own accord. I printed the constant "**MENU**" and used the **"menu_choice"** variable to request input from the user. I then also decided to use the match and case practice over if/elif for the menu portion of the program. (Image 1.9)



```
# Present the menu of choices
while True:
    print(MENU)
    menu_choice = input("Please enter your selection from the menu above: ")
    match menu_choice:
```

Image 1.9: Starting a loop at the menu of options for the user and matching to the menu_choice variable

We need to request data from the user. To start this, we match “1” with the following code (image 1.10)

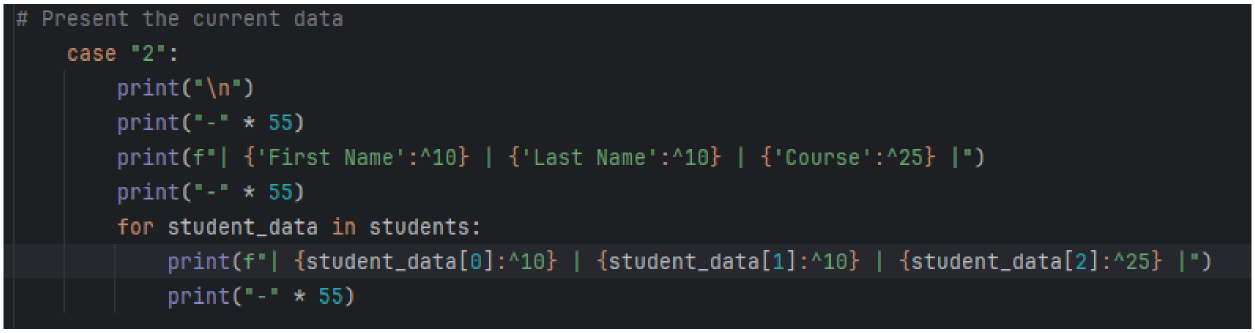


```
# Input user data
    case "1":
        student_first_name = input("What is the student's first name? ")
        student_last_name = input("What is the student's last name? ")
        course_name = input("What course would you like to enroll the student in? ")
        message = (f"\nThank you, {student_first_name} {student_last_name}. You have successfully registered"
                   f" for {course_name}")
        students.append([student_first_name, student_last_name, course_name])
        print(message)
```

Image 1.10: Gathering student data

The above code requests data from the user, including their first name, last name, and course they wish to sign up for. Upon receiving all of the data, the code then appends the input student data to the **“students”** list of lists and a message is printed using an f-function to display the input information back to the user as a SUCCESS message.

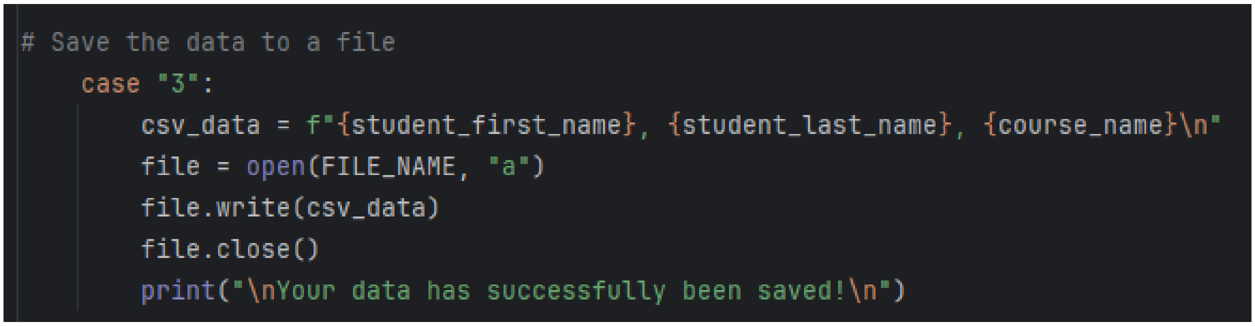
When a user selections option 2, the code prints the previously collected student data (from previous runs of the program) as well as any new data input by the user from this session. I formatted the printout a bit so it looks like a chart of sorts by using the print function to print hyphens a specific amount of times. I learned through some research that we can center align printed text utilizing “:^10.” (Image 1.11)



```
# Present the current data
    case "2":
        print("\n")
        print("-" * 55)
        print(f"| {'First Name':^10} | {'Last Name':^10} | {'Course':^25} |")
        print("-" * 55)
        for student_data in students:
            print(f"| {student_data[0]:^10} | {student_data[1]:^10} | {student_data[2]:^25} |")
            print("-" * 55)
```

Image 1.11: Displaying all collected student data in an easily readable chart to the user.

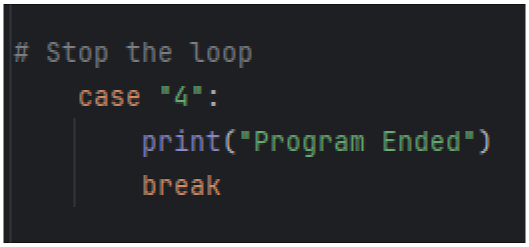
Option 3 of the menu takes the data input by the user and formats it for a CSV file. Using an f-string, I stored the student’s first and last name and the course name into the variable **“csv_data.”** The code then opens the constant **“FILENAME”** and write the CSV data to the file. The file is closed and a success message appears. (Image 1.12)



```
# Save the data to a file
    case "3":
        csv_data = f"{student_first_name}, {student_last_name}, {course_name}\n"
        file = open(FILE_NAME, "a")
        file.write(csv_data)
        file.close()
        print("\nYour data has successfully been saved!\n")
```

Image 1.12: Saving collected data to the file

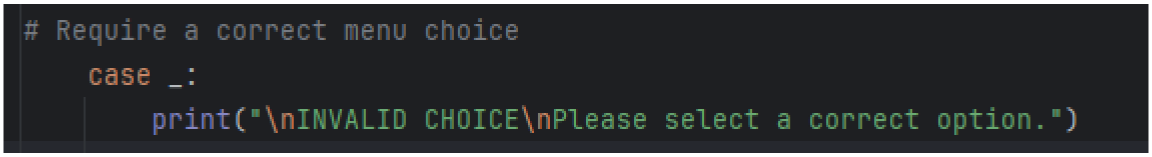
Option 4 allows us to break out of the loop to ensure that the user can end the cycle. Selecting this option will print a message informing the user that the program has ended. (Image 1.13)



```
# Stop the loop
    case "4":
        print("Program Ended")
        break
```

Image 1.13: Breaking from the loop and ending the program

As in the last assignment, I added an option to inform the user if their input for the menu is invalid. This informs the user that they need to enter an appropriate option from the printed menu. (Image 1.14)

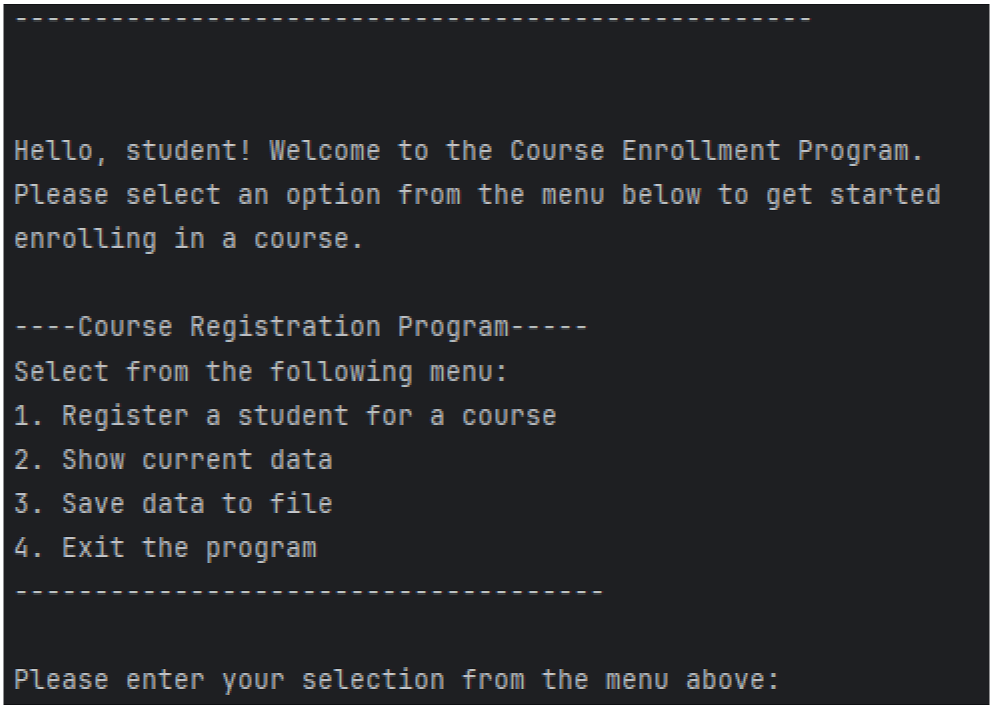


```
# Require a correct menu choice
    case _:
        print("\nINVALID CHOICE\nPlease select a correct option.")
```

Image 1.14: Letting the user know their chosen option is invalid

Running the program

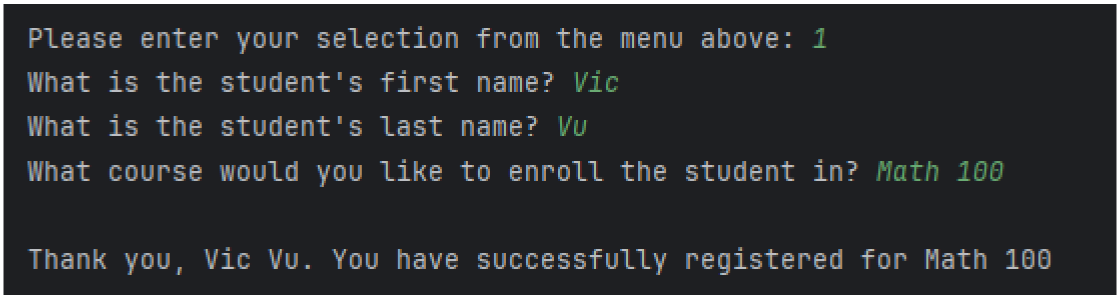
Now, it’s time to test that the program is working as expected. In PyCharm, I started the program using Shift+F10. (Image 2.1)



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


Hello, student! Welcome to the Course Enrollment Program.
Please select an option from the menu below to get started
enrolling in a course.

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

Please enter your selection from the menu above:
```

Image 2.1: Starting the program

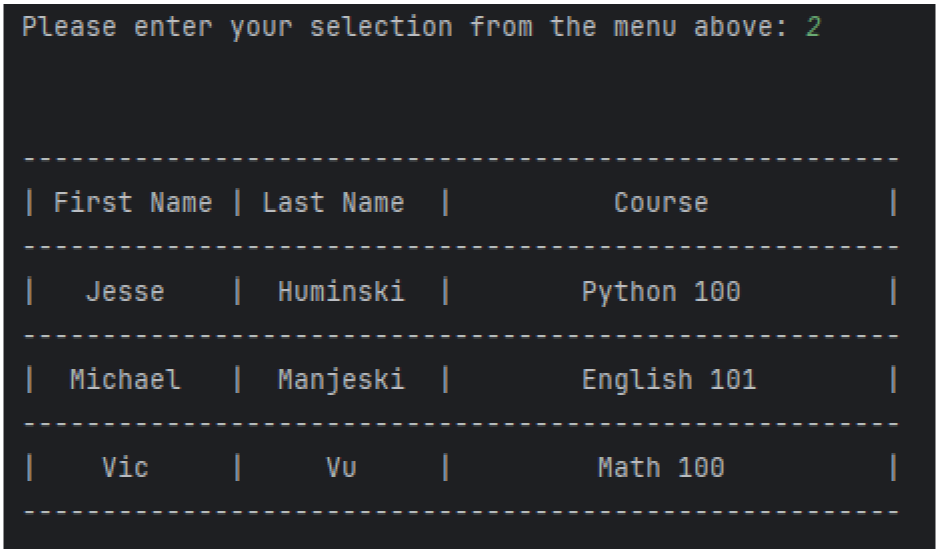
So far, so good! The menu is printed and the user is prompted to enter an option from the menu above. Let’s enter our new data. I am going to enroll student “Vic Vu” to “Math 100.” (Image 2.2)



```
Please enter your selection from the menu above: 1
What is the student's first name? Vic
What is the student's last name? Vu
What course would you like to enroll the student in? Math 100

Thank you, Vic Vu. You have successfully registered for Math 100
```

Image 2.2: Option 1 prompts the user to enter their name and course. A success message is printed

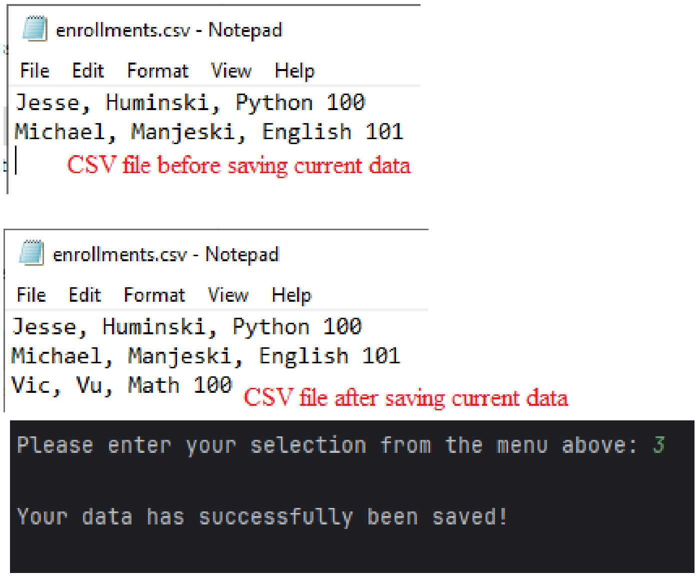
Now it’s time to see if the data is captured and displayed as expected. (Image 2.3)



```
Please enter your selection from the menu above: 2



-------------------------------------------------------
| First Name | Last Name  |          Course           |
-------------------------------------------------------
|   Jesse    |  Huminski  |        Python 100         |
-------------------------------------------------------
|  Michael   |  Manjeski  |        English 101        |
-------------------------------------------------------
|    Vic     |     Vu     |         Math 100          |
-------------------------------------------------------
```

Image 2.3: Option 2 displays currently captured data and reads previously captured data from the file appropriately. Chart is displayed for readability.

Now we need to test that the data is being saved to the file correctly. Option 3 should saved the currently collected student data to the CSV file. (Image 2.4)


enrollments.csv - Notepad
File Edit Format View Help

```
Jesse, Huminski, Python 100
Michael, Manjeski, English 101
```

CSV file before saving current data

enrollments.csv - Notepad
File Edit Format View Help

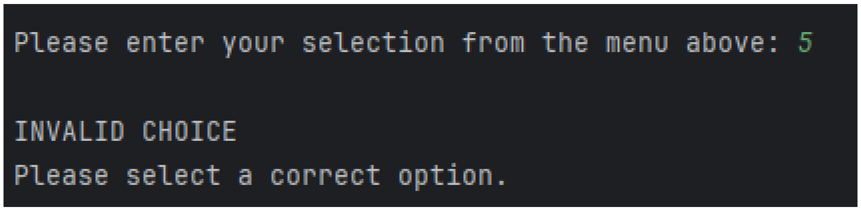
```
Jesse, Huminski, Python 100
Michael, Manjeski, English 101
Vic, Vu, Math 100
```

CSV file after saving current data

```
Please enter your selection from the menu above: 3

Your data has successfully been saved!
```

Image 2.4: CSV data before and after the code writes data to the file. Success message is also displayed

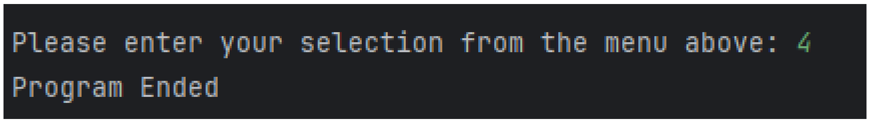
Before we exit, I want to make sure that the input check is correct. So, I enter “5” in the selection. (Image 2.5)



```
Please enter your selection from the menu above: 5

INVALID CHOICE
Please select a correct option.
```

Image 2.5: Error message informing the user their selection is incorrect.

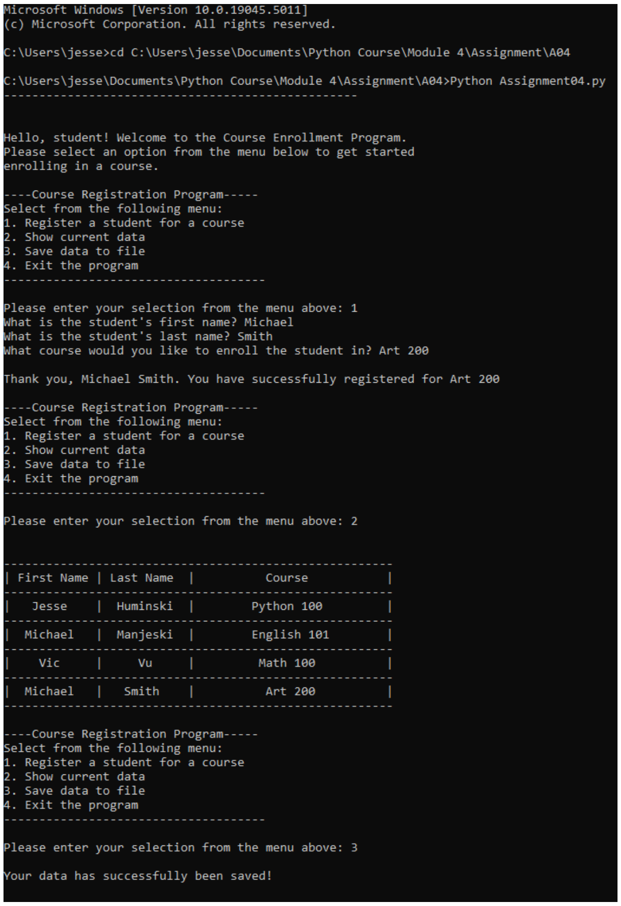
Now, let’s break out of the loop and end the program. **(**Image 2.6)



```
Please enter your selection from the menu above: 4
Program Ended
```

Image 2.6: Exiting the program

And, now it’s time to test the program in the command shell. (Image 2.7)



```
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jesse>cd C:\Users\jesse\Documents\Python Course\Module 4\Assignment\A04

C:\Users\jesse\Documents\Python Course\Module 4\Assignment\A04>Python Assignment04.py
--------------------------------------------------


Hello, student! Welcome to the Course Enrollment Program.
Please select an option from the menu below to get started
enrolling in a course.

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

Please enter your selection from the menu above: 1
What is the student's first name? Michael
What is the student's last name? Smith
What course would you like to enroll the student in? Art 200

Thank you, Michael Smith. You have successfully registered for Art 200

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

Please enter your selection from the menu above: 2


-------------------------------------------------------
| First Name | Last Name  |          Course           |
-------------------------------------------------------
|   Jesse    |  Huminski  |        Python 100         |
-------------------------------------------------------
|  Michael   |  Manjeski  |        English 101        |
-------------------------------------------------------
|    Vic     |     Vu     |         Math 100          |
-------------------------------------------------------
|  Michael   |   Smith    |          Art 200          |
-------------------------------------------------------

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

Please enter your selection from the menu above: 3

Your data has successfully been saved!
```

Image 2.7: Ensuring command shell success

Summary

Module 04 expanded on our knowledge of working with files to store data and ensure data persistence for later iterations of program runs. This program utilizes that knowledge by creating code that allows a student to sign up for a course and have that data saved for future use. This program allows us on the backend of things see the data input by the users and view the file once the program has ended. Additionally, the code allows the user to see what students have used the program and read display that data back to them in a easy-to-read chart.