

Ogonna Anunoby

08/28/2024

IT FDN 110 B Su 24: Foundations Of Programming: Python

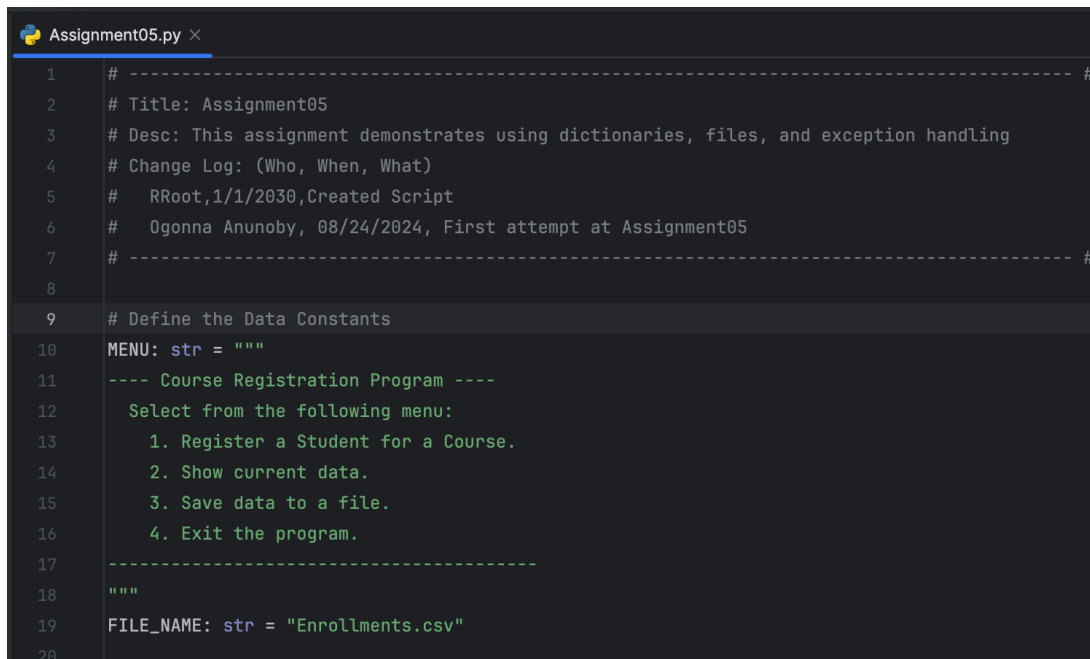
Assignment 05

Assignment 05 Report

Introduction

In Assignment 05, I wrote a script that registers students for various courses. It was similar to Assignment 04, but I used a list of dictionaries to handle student data instead of a list of lists. I also got to practice using exception handling to complete this assignment. I also got some experience using GitHub to turn in my assignment. Assignment 05 allowed me to continue developing my Python skills.

Script Testing

A screenshot of a code editor window titled "Assignment05.py". The code is written in Python and includes a header section with comments and a section for defining data constants. The header section (lines 1-8) contains a multi-line comment starting with "# ----- #" and providing details about the assignment, including the title, description, change log, and author. The data constants section (lines 9-20) defines a menu for the "Course Registration Program" and a file name "Enrollments.csv".

```
1 # ----- #
2 # Title: Assignment05
3 # Desc: This assignment demonstrates using dictionaries, files, and exception handling
4 # Change Log: (Who, When, What)
5 #   RRoot,1/1/2030,Created Script
6 #   Ogonna Anunoby, 08/24/2024, First attempt at Assignment05
7 # ----- #
8
9 # Define the Data Constants
10 MENU: str = """
11 ---- Course Registration Program ----
12 Select from the following menu:
13     1. Register a Student for a Course.
14     2. Show current data.
15     3. Save data to a file.
16     4. Exit the program.
17 -----
18 """
19 FILE_NAME: str = "Enrollments.csv"
20
```

Figure 1: Script Header and Data Constants for Assignment 05 Script Code

Figure 1 shows the header and data constants for the Assignment 05 script code. Lines 1 through 7 consist of the script header. The change log has one entry, showing that I made my first attempt of Assignment 05 on 08/28/2024. Lines 9 through 19 are where I have defined my data constants. In this script, there are two data constants. MENU is a string that displays the registration menu to the user. FILE_NAME is a string that holds the name of the file, Enrollments.csv. This file will save the registration data.

```
21 # Define the Data Variables
22 student_first_name: str = '' # Holds the first name of a student entered by the user.
23 student_last_name: str = '' # Holds the last name of a student entered by the user.
24 course_name: str = '' # Holds the name of a course entered by the user.
25 student_data: dict = {} # one row of student data
26 students: list = [] # a table of student data
27 csv_data: str = '' # Holds combined string data separated by a comma.
28 file = None # Holds a reference to an opened file.
29 menu_choice: str # Hold the choice made by the user.
30
31 # When the program starts, read the file data into table
32 # Extract the data from the CSV file into a list of dictionaries
```

Figure 2: Data Variables for Assignment 05 Script Code

Figure 2 shows the data variables for this assignment. Lines 21 through 29 define these data variables and their initial values. Here, the variables student_first_name, student_last_name, course_name, csv_data, and menu_choice are initialized as empty strings. Meanwhile, file is initialized to “None”, indicating that it has been initialized to have no value. The variable student_data is initialized to an empty dictionary, while students has been initialized to an empty list.

```

31 # When the program starts, read the file data into table
32 # Extract the data from the CSV file into a list of dictionaries
33 try:
34     file = open(FILE_NAME, "r")
35
36     for row in file.readlines():
37         # Transform the data from the file
38         student_data = row.split(',')
39         student_data = {"FirstName": student_data[0],
40                         "LastName": student_data[1],
41                         "CourseName": student_data[2].strip()}
42         # Load it into our collection (list of lists)
43         students.append(student_data)
44         file.close()
45 except FileNotFoundError as e:
46     print("Text file must exist before running this script!\n")
47     print("-- Technical Error Message -- ")
48     print(e, e.__doc__, e.__str__(), type(e), sep="\n")
49     print("Creating file..")
50     file = open(FILE_NAME, "w")
51 except Exception as e:
52     print("There was a non-specific error!\n")
53     print("-- Technical Error Message -- ")
54     print(e, e.__doc__, e.__str__(), type(e), sep="\n")
55 finally:
56     if not file.closed:
57         file.close()
58

```

Figure 3: Opening the CSV File with Exception Handling

Figure 3 shows the code that will read the CSV file before allowing the user to make any choices. Here, the code from lines 33 through 44 show the try block where the script attempts to open the CSV file. If it does this successfully, it then proceeds to get all of the data in the CSV file. For each line that is read, the data in that line is converted into a list (line 38). Then on lines 39 through 41, the elements of the list are saved into a dictionary called `student_data` with the keys `FirstName`, `LastName`, and `CourseName`. The data stored in `students[CourseName]` is stripped of any whitespace characters.

If the file is unable to be found, the `FileNotFoundError` exception is thrown and code goes to line 45. In this except block running from lines 45 through 50, error messages are printed to the screen, as well as the error's docstring, string object, and type. In line 48, by setting the separator to `"\n"`, the print statement can print multiple items on separate lines while using a single print statement. Line 50 opens the file for in write mode, allowing an empty file to be created if it does not exist. After the `FileNotFoundError` exception block, there is a general exception block. Since exception blocks are checked in the order in which they are listed in the code, in this script, the general exception block will be checked after the `FileNotFoundError` block. The general exception block will catch any exceptions that arise in the try block that are not accounted for by the `FileNotFoundError` block. In the finally block from lines 55 through

57, which will always run, line 56 will check if the CSV file is still open. If it is, it will close it on line 56.

```
59 # Present and Process the data
60 while (True):
61
62     # Present the menu of choices
63     print(MENU)
64     menu_choice = input("What would you like to do: ")
65
66     # Input user data
67     if menu_choice == "1": # This will not work if it is an integer!
68         try:
69             # Input the data
70             student_first_name = input("Enter the student's first name: ")
71             if not student_first_name.isalpha(): # Check if only letters are entered
72                 raise ValueError("The first name should only contain letters.")
73
74             student_last_name = input("Enter the student's last name: ")
75             if not student_last_name.isalpha(): # Check if only letters are entered
76                 raise ValueError("The last name should only contain letters.")
77         except ValueError as e: # Catch value error exceptions
78             print(e) # Prints the custom message
79             print("-- Technical Error Message -- ")
80             print(e.__doc__)
81             print(e.__str__())
82             print(type(e))
83         except Exception as e: # Catch general exceptions
84             print("There was a non-specific error!\n")
85             print("-- Technical Error Message -- ")
86             print(e, e.__doc__, e.__str__(), type(e), sep="\n")
87         else:
88             course_name = input("Enter the name of the course: ")
89             student_data = {"FirstName": student_first_name, "LastName": student_last_name, "CourseName" : course_name}
90             students.append(student_data)
91             print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
92             continue
```

Figure 4: Printing The Menu and Registering a Student For a Course

Figure 4 shows the beginning of the while loop and what happens when the user makes choice 1. The entire while loop runs from lines 60 through 134. Line 60 has an infinite while loop that is always true. Line 63 prints the menu to the screen. Line 64 asks the user to choose an action. Lines 67 through 92 are for choice 1. For this choice, in the try block from lines 68 through 76, the user is asked to enter a first name on line 70. Line 71 checks if the user entered all alphabetical characters for the student's first name. If they did, the code asks the user to enter the student's second name on line 74. If they did not enter all alphabetical characters for the students's first name, a ValueError exception will be thrown, and the code will jump to the ValueError exception on line 77.

Line 75 checks if the user entered all alphabetical characters for the student's last name. If they did, the code would move to else block on line 87. The else block on lines 87 though 91 will only run if no exceptions were thrown. If non-alphabetical characters were entered for the student's last name, a ValueError exception will be thrown, and the code will jump to line 77. If the ValueError exception is thrown, the code jumps to line 77. From lines 78 through 82, error messages will be printed, as well as the error's docstring, string object,

and type, thus informing the user of the error's details. The except block for general exceptions on lines 83 through 86 are similar to the exception block on lines 77 through 82, except it will catch non-ValueError exceptions that occur in the try block (lines 68 through 76) that the programmer did not account for. Exceptions are checked in the order that they are written, so the ValueError exception will always be checked first.

The else block on lines 87 through 91 will be run if no exceptions are thrown. Here, the user is asked to enter the name of the course. Then, on lines 89 and 90, the dictionary for the student is created with FirstName, LastName, and CourseName as keys and student_first_name, student_last_name, and course_name as values. The student dictionary is appended to the students list on line 91. Then a message letting the user know that the student was registered is printed to the screen. On line 92, the "continue" keyword takes the user back to the while loop on line 60.

```
94     # Present the current data
95     elif menu_choice == "2":
96
97         # Process the data to create and display a custom message
98         print("-"*50)
99         for student in students:
100             print(f"Student {student['FirstName']} {student['LastName']} is enrolled in {student['CourseName']}")
101         print("-"*50)
102         continue
```

Figure 5: Showing Current Data to The User

Figure 5 shows the code when the user chooses choice 2, which displays the data of every currently registered student. Lines 98 and 101 print a boarder to the screen that surrounds the printed student information. This does this by printing "-" fifty times to the screen. Lines 99 and 100 iterate through each student dictionary in the students list and prints out each student's information using the student dictionary and the FirstName, LastName, and CourseName keys. Also, the "continue" keyword on line 101 ensures that the code will loop back to the beginning of the while loop on line 60.

```

104     # Save the data to a file
105     elif menu_choice == "3":
106         try:
107             file = open(FILE_NAME, "w")
108             for student in students:
109                 csv_data = f"{student['FirstName']},{student['LastName']},{student['CourseName']}\n"
110                 file.write(csv_data)
111             file.close()
112             print("The following data was saved to file!")
113             for student in students:
114                 print(f"Student {student['FirstName']} {student['LastName']} is enrolled in {student['CourseName']}")
115         except TypeError as e: # Catch type error exceptions
116             print("Please check that the data is a compatible with CSV files\n")
117             print("-- Technical Error Message -- ")
118             print(e, e.__doc__, e.__str__(), type(e), sep="\n")
119         except Exception as e: # Catch general exceptions
120             print("There was a non-specific error!\n")
121             print("-- Technical Error Message -- ")
122             print(e, e.__doc__, e.__str__(), type(e), sep="\n")
123         finally:
124             if not file.closed:
125                 file.close()
126         continue

```

Figure 6: Saving Data to The File

Figure 6 shows what happens if the user chooses choice 3. Here, registration data is saved to the CSV file. Here in the try block, on line 106, the code attempts to open the file in write mode. Then, it loops through the dictionaries of student data in the students list, extracts each student's data into the desired format for CSV files, writes it to the CSV file, then closes the file. If this try block fails due to the data being incompatible with the file, the code will throw a `TypeError` exception and jump to line 115. Lines 117 and 118 show error messages that will print to the screen. On line 119, the error, its docstring, its string object, and its error type will print on different lines (as specified by the separator (`sep`) being set to `"\n"`). The except block shown in lines 119 through 122 is similar to the except block on lines 115 through 118. This block is a general exceptions block. It is placed after the `TypeError` exception block, so that it is checked after it. This allows it to catch any other non-`TypeError` exceptions that may arise. In the finally block on lines 123 through 126, no matter if an exception was thrown or not, the code will check if the file is still open. If it is, the file will be closed. Also, the `"continue"` keyword on line 126 ensures that the code will loop back to the beginning of the while loop on line 60.

```

127
128     # Stop the loop
129     elif menu_choice == "4":
130         break # out of the loop
131     else:
132         print("Please only choose option 1, 2, or 3")
133
134     print("Program Ended")
135

```

Figure 7: Ending the Program

Figure 7 shows what happens when the user chooses 4 or any other option that is not 1, 2, or 3. If they choose 4, the code breaks out of the infinite while loop. It then runs line 132, printing “Ending program...” to the screen and terminates the program. If an option other than 1, 2, 3, or 4 is chosen, the user is reminded that only 1, 2, 3, and 4 are valid options. Then the while loop repeats again.

Script Testing

After writing the code for the script, I needed to test it. First, I saved the script using the name Assignment05.py to my desktop. I ran the script and tested all of the options to ensure that my script worked as I expected.

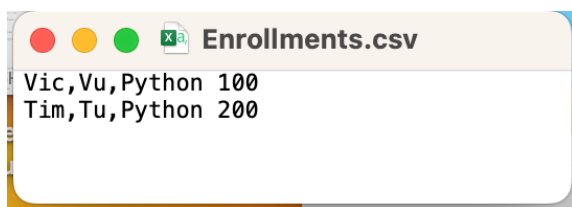


Figure 7: Enrollments.csv Before Running the Script

Figure 7 shows the contents of Enrollments.csv before running the script. I ran the script in PyCharm and chose choice 2 to display each registered student. The contents of the CSV file were printed to the screen, as expected, as shown in Figure 8.

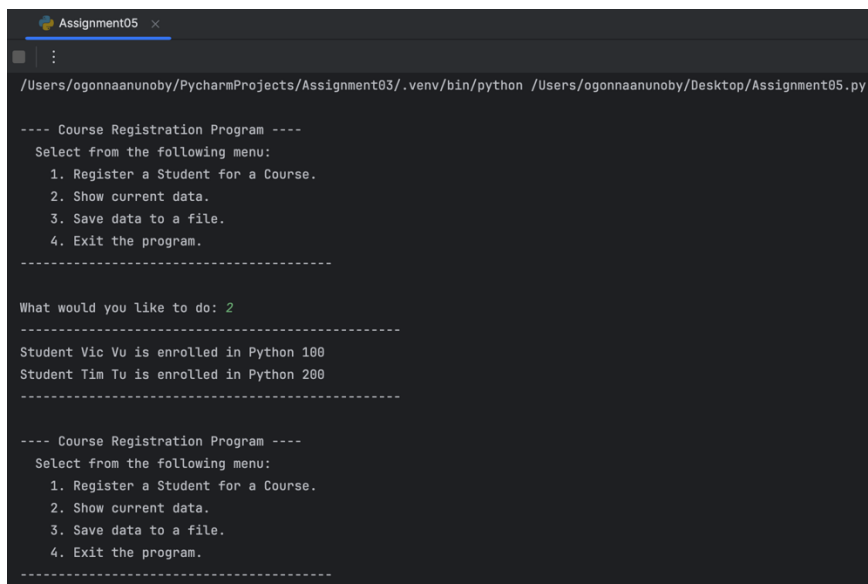
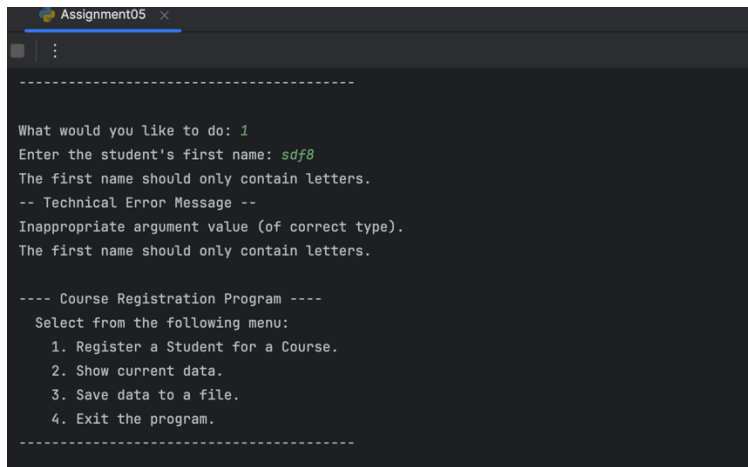


Figure 8: Students Enrolled Before Running the Script

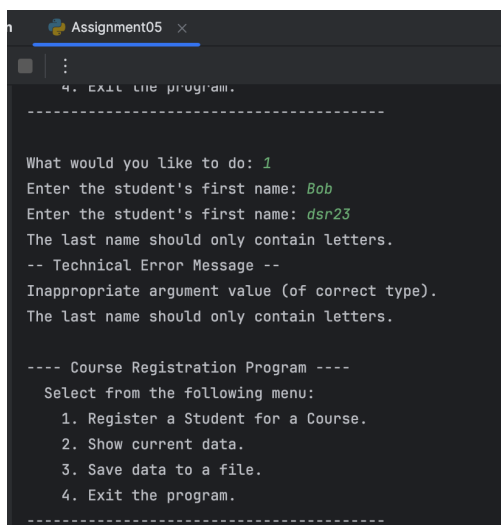
Next, I made choice 1 to register a student for the course. I entered a name with numbers in it and the ValueError exception was thrown.



```
-----  
What would you like to do: 1  
Enter the student's first name: sdf8  
The first name should only contain letters.  
-- Technical Error Message --  
Inappropriate argument value (of correct type).  
The first name should only contain letters.  
  
---- 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 8: ValueError Exception for the Student's First Name

Then, I made choice 1 again and entered all alphabetic characters for the student's first name. The code then prompted me to enter the student's last name. I entered a name with numbers in it, and the ValueError exception was thrown again.



```
4. Exit the program.  
-----  
What would you like to do: 1  
Enter the student's first name: Bob  
Enter the student's first name: dsr23  
The last name should only contain letters.  
-- Technical Error Message --  
Inappropriate argument value (of correct type).  
The last name should only contain letters.  
  
---- 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.  
-----
```

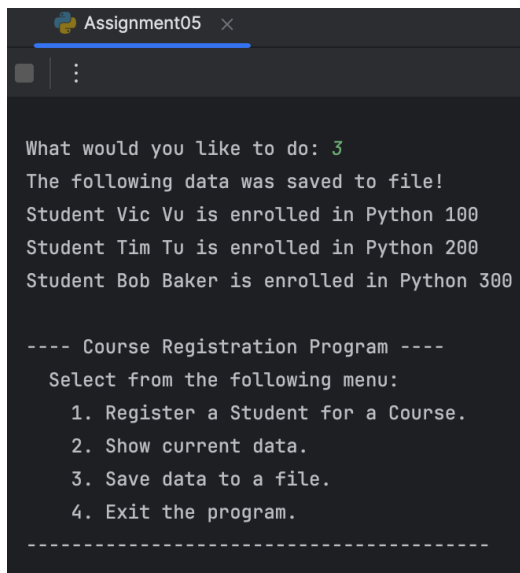
Figure 9: ValueError Exception for the Student's Last Name

I then entered the name of the course, and the student was registered for the course.


```
-----  
  
What would you like to do: 1  
Enter the student's first name: Bob  
Enter the student's last name: Baker  
Enter the name of the course: Python 300  
You have registered Bob Baker for Python 300.  
  
---- 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: Successfully Registering a Student

Figure 10 shows me making choice 3, displaying the data and saving the data to Enrollments.csv.



```
Assignment05 x  
-----  
  
What would you like to do: 3  
The following data was saved to file!  
Student Vic Vu is enrolled in Python 100  
Student Tim Tu is enrolled in Python 200  
Student Bob Baker is enrolled in Python 300  
  
---- 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 10: Saving All of the Registered Students

In Figure 11, I enter an invalid choice, and I get an error message and am prompted to try again. I then choose 4 and exit the program.

```
What would you like to do: 5
Please only choose option 1, 2, 3, or 4

---- 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 10: Entering an invalid Option and Exiting the Program

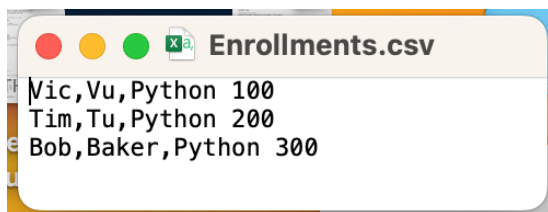


Figure 11: Showing All Registered Students in the CSV File

Figure 11 shows that each registered student has been saved to Enrollments.csv. I also ran the script in the terminal using the same commands as I did in PyCharm. I started with the Enrollments.csv file shown in Figure 11. Then, I ran the script in the terminal with the same results. The script output in the terminal is shown in Figures 12 and 13. Figure 14 shows the final Enrollments.csv file I got.

```
[Ogonnas-MBP:Desktop ogonnaanunoby$ python3 Assignment05.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: 2
```

```
-----
Student Vic Vu is enrolled in Python 100
Student Tim Tu is enrolled in Python 200
Student Bob Baker is enrolled in Python 300
-----
```

```
---- 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: ssdkm9
The first name should only contain letters.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The first name should only contain letters.
<class 'ValueError'>
```

```
---- 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: we5jn
The last name should only contain letters.
-- Technical Error Message --
Inappropriate argument value (of correct type).
The last name should only contain letters.
<class 'ValueError'>
```

```
---- 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 12: Terminal Script Results, Part 1

```
-- Technical Error Message --
Inappropriate argument value (of correct type).
The last name should only contain letters.
<class 'ValueError'>

---- 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: Salias
Enter the name of the course: Python 400
You have registered Sue Salias for Python 400.

---- 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!
Student Vic Vu is enrolled in Python 100
Student Tim Tu is enrolled in Python 200
Student Bob Baker is enrolled in Python 300
Student Sue Salias is enrolled in Python 400

---- 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: 5
Please only choose option 1, 2, or 3

---- 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
Ogonnas-MBP:Desktop ogonnaanunoby$
```

Figure 13: Terminal Script Results, Part 2

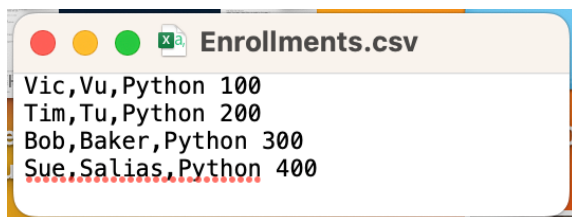


Figure 14: Again, Showing All Registered Students in the CSV File

Finally, I uploaded my report and python script to the GitHub repository I created for this course. This can be found at <https://github.com/944695/IntroToProg-Python>.

Conclusion

In conclusion, Assignment 05 allowed me to practice the skills I gained from Lesson 5. It allowed me to practice exception handling, as well as using manipulating data in the form of a list of dictionaries. I also got experience using GitHub. Overall, completing Assignment 03 has helped me continue building my Python knowledge.

Citations

Mod05-Notes.docx

Assignment05.py