Mine Hantal Wernsing
5/12/2024
IT FDN110 B
A05 – Advanced Collections and Error Handling
[GitHub](#)

Creating and Testing a Python Script for Data Processing Advanced Collections and Error Handling

Introduction

A Python program that demonstrates using constants, variables, and print statements to display a message about a student's registration for a Python course with the use of data processing using dictionaries and exception handling is created and tested in Python IDE (Integrated Development Environment); PyCharm and Command Prompt successfully.
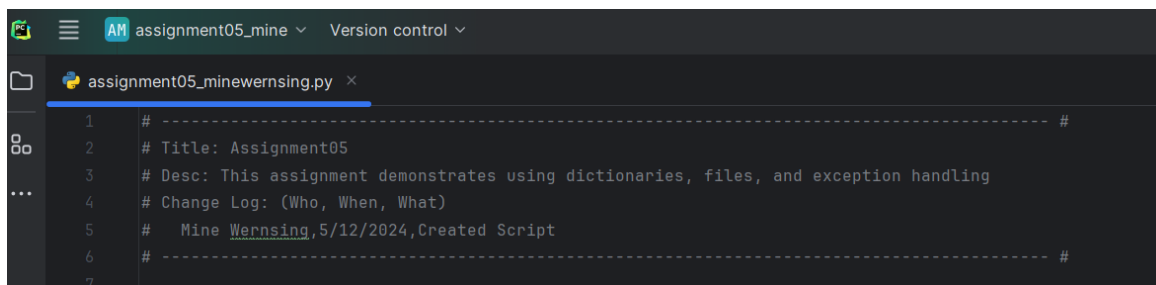
Creating the Script

File Name:

The file is named assignment05_minewernsing.py (Fig. 1)

Script Header:

The script header includes the title, description, change log and is updated with name and the current date (Fig. 1).


```
1   # -------------------------------------------------------------------------- #
2   # Title: Assignment05
3   # Desc: This assignment demonstrates using dictionaries, files, and exception handling
4   # Change Log: (Who, When, What)
5   #    Mine Wernsing,5/12/2024,Created Script
6   # -------------------------------------------------------------------------- #
7
```

Figure 1. File name and script header.

Constants:

The constant **MENU: str** is set to the value:

```
---- 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

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

The constant **FILE_NAME: str** is set to the value "Enrollments.csv" (Fig. 2).

Variables:

student_first_name: str is set to an empty string.

student_last_name: str is set to an empty string.

course_name: str is set to an empty string.
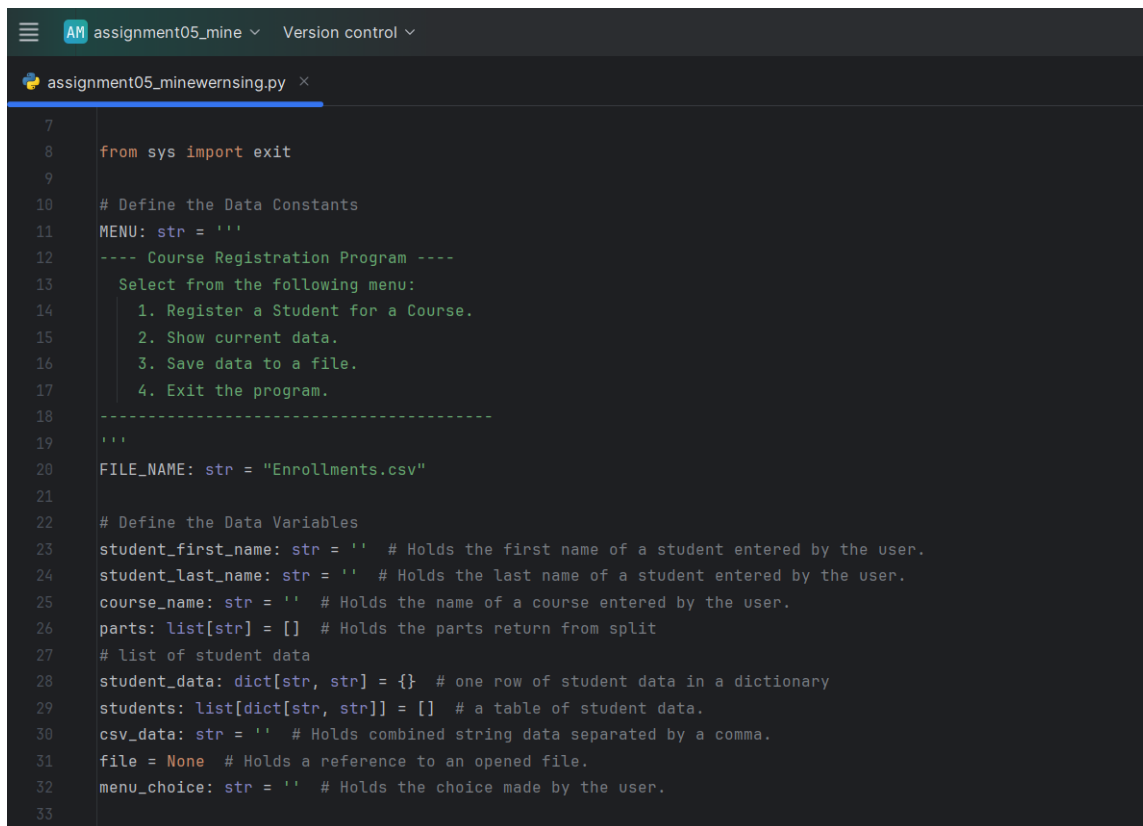
csv_data: str is set to an empty string.

file is set to None.

menu_choice: str is set to an empty string.

parts: list : list is set to an empty list.

student_data: <u>dict</u> is set to an empty dictionary.

students: list : list is set to an empty list (Fig. 2).

```
7
8     from sys import exit
9
10    # Define the Data Constants
11    MENU: str = '''
12    ---- Course Registration Program ----
13      Select from the following menu:
14        1. Register a Student for a Course.
15        2. Show current data.
16        3. Save data to a file.
17        4. Exit the program.
18    -----------------------------------------
19    '''
20    FILE_NAME: str = "Enrollments.csv"
21
22    # Define the Data Variables
23    student_first_name: str = ''  # Holds the first name of a student entered by the user.
24    student_last_name: str = ''  # Holds the last name of a student entered by the user.
25    course_name: str = ''  # Holds the name of a course entered by the user.
26    parts: list[str] = []  # Holds the parts return from split
27    # list of student data
28    student_data: dict[str, str] = {}  # one row of student data in a dictionary
29    students: list[dict[str, str]] = []  # a table of student data.
30    csv_data: str = ''  # Holds combined string data separated by a comma.
31    file = None  # Holds a reference to an opened file.
32    menu_choice: str = ''  # Holds the choice made by the user.
33
```

Figure 2. Constants and variables.

Input / Output:

On menu choice 1, the program prompts the user to enter the student's first name and last name, followed by the course name, using the input() function and stores the inputs in the respective variables (Fig. 3).

On menu choice 2, the program presents a coma-separated string by formatting the collected data using the print() function (Fig. 4).

Data collected for menu choice 1 is added to a list of dictionaries.

All data in the list is displayed when menu choice 2 is used (Fig. 8 and 12).

```python
64
65    # Present and Process the data
66    while True:
67
68        # Present the menu of choices
69        print(MENU)
70        menu_choice = input("What would you like to do: ")
71
72        # Input user data
73        if menu_choice == "1":  # This will not work if it is an integer!
74            student_first_name = input("Enter the student's first name: ")
75            try:
76                if not student_first_name.isalpha():
77                    raise ValueError("Student first name must be alphabetic.")
78                student_last_name = input("Enter the student's last name: ")
79                if not student_last_name.isalpha():
80                    raise ValueError("Student last name must be alphabetic.")
81                course_name = input("Please enter the name of the course: ")
82                if course_name == "":
83                    raise ValueError("Course name cannot be an empty string.")
84                student_data = {'first_name': student_first_name, 'last_name': student_last_name, 'course': course_name}
85
86                # Load it into our collection (list of dictionaries)
87                students.append(student_data)
88
89            except ValueError as e:
90                print(e)
91                print("---Technical Information")
92                print(e, e.__doc__, type(e), sep="\n")
93
```

Figure 3. Processing menu choice 1 and raising ValueError exception.

```python
93
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['first_name']} {student['last_name']} is enrolled in {student['course']}")
101           print("-"*50)
102
103       # Save the data to a file
104       elif menu_choice == "3":
105           try:
106               file = open(FILE_NAME, "w")
107               for student in students:
108                   csv_data = f"{student['first_name']},{student['last_name']},{student['course']}\n"
109                   file.write(csv_data)
110               file.close()
111           except Exception as e:
112               print("---Technical Information")
113               print(e, e.__doc__, type(e), sep='n')
114
115           finally:
116               if file is not None:
117                   if not file.closed:
118                       file.close()
119
120       # Stop the loop
121       elif menu_choice == "4":
122           print("Program Ended")
123           exit()
124
```

Figure 4. Processing menu choice 2, 3, 4 and catching an exception.

Processing

When the program starts, the contents of the "**Enrollments.csv**" are automatically read into a list of dictionary rows (Fig. 5).

```
33
34    # When the program starts, read the file data into a list of dictionaries (table)
35    try:
36        file = open(FILE_NAME, "r")
37        for line in file.readlines():
38            # Extract the data from the file
39            parts = line.strip().split(',')  # split() turns the string into a list
40            student_first_name = parts[0]
41            student_last_name = parts[1]
42            course_name = parts[2]
43
44            # Transform the data from the file
45            student_data = {'first_name': student_first_name, 'last_name': student_last_name, 'course': course_name}
46
47            # Load it into our collection (list of dictionaries)
48            students.append(student_data)
49
50    except FileNotFoundError as e:
51        print(f"{FILE_NAME} file not found.")
52        print("---Technical Information---")
53        print(e, e.__doc__, type(e), sep="\n")
54
55    except Exception as e:
56        print('Text entered is invalid.')  # Text file not found
57        print("---Technical Information")
58        print(e, e.__doc__, type(e), sep='\n')
59
60    finally:
61        if file is not None:
62            if not file.closed:
63                file.close()
```

Figure 5. When the program starts, the contents of the Enrollments.csv is automatically read and catching FileNotFoundError exception.

On menu choice 3, the program opens a file named "Enrollments.csv" in write mode using the open() function. It writes the content of the csv_data variable to the file using the write() function, then file is closed using the close() method. Then it displays what is stored in the file (Fig. 4).

While True, if, elif and else statements are used to create the loop for multiple entries and to break out from the loop and print "Please only choose option 1, 2, 3 or 4" for any other options entered (Fig. 6, 10 and 14).

On menu choice 4, the program ends (Fig. 6).

```
119
120        # Stop the loop
121        elif menu_choice == "4":
122            print("Program Ended")
123            exit()
124
125        else:
126            print("Please only choose option 1, 2, 3 or 4")
127
```

Figure 6. Processing menu choice 4 and other choices.

Error Handling

The program provides structured error handling when the file is read into the list of dictionary rows (Fig. 5).

The program provides structured error handling when the user enters a first name (Fig. 3, 15).

The program provides structured error handling when the user enters a last name (Fig. 3, 15).

The program provides structured error handling when the dictionary rows are written to the file (Fig. 4).

```
What would you like to do: 1
Enter the student's first name:    Sue1
Student first name must be alphabetic.
---Technical Information
Student first name must be alphabetic.
Inappropriate argument value (of correct type).
<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:
Student last name must be alphabetic.
---Technical Information
Student last name must be alphabetic.
Inappropriate argument value (of correct type).
<class 'ValueError'>
```
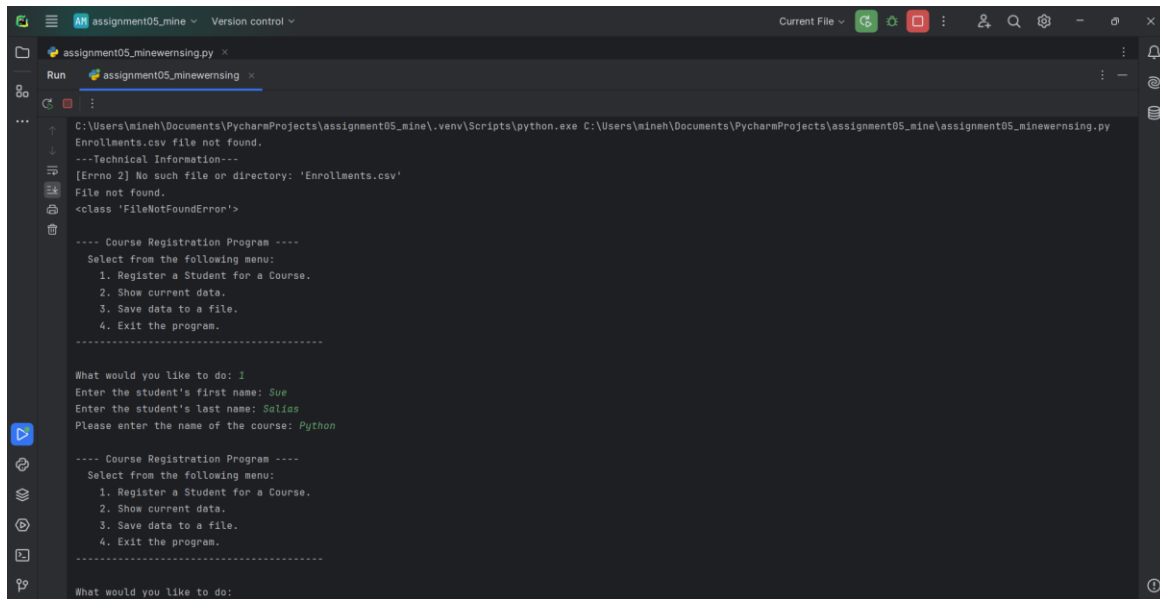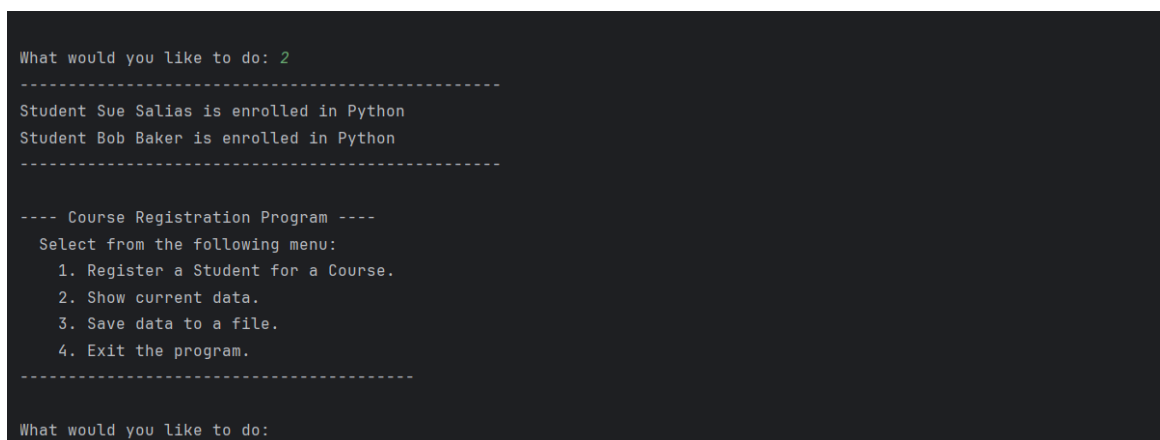
Figure 15. Error handling examples

Testing the Script

Test

The program takes the user's input for a student's first, last name, and course name (Fig. 7 and 11).

Figure 7. Testing the script in PyCharm for menu 1 choice.

The program displays the user's input for a student's first, last name, and course name (Fig. 8 and 12).

The program saves the user's input for a student's first, last name, and course name to a coma-separated string file.

Figure 8. Testing the script in PyCharm for menu 2 choice.

The program allows users to enter multiple registrations (first name, last name, course name) (Fig. 9 and 14).

The program allows users to display multiple registrations (first name, last name, course name) (Fig. 8 and 12).
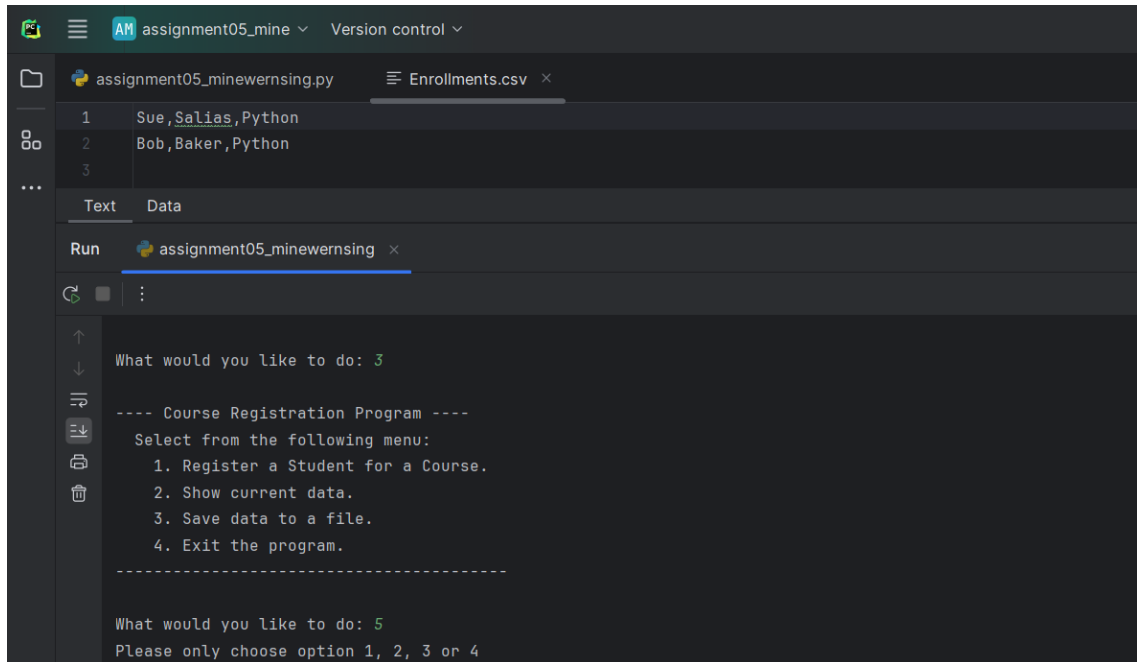

Figure 9. Testing the script in PyCharm for menu 3 choice.

The program allows users to save multiple registrations to a file (first name, last name, course name) (Fig. 9 and 13).

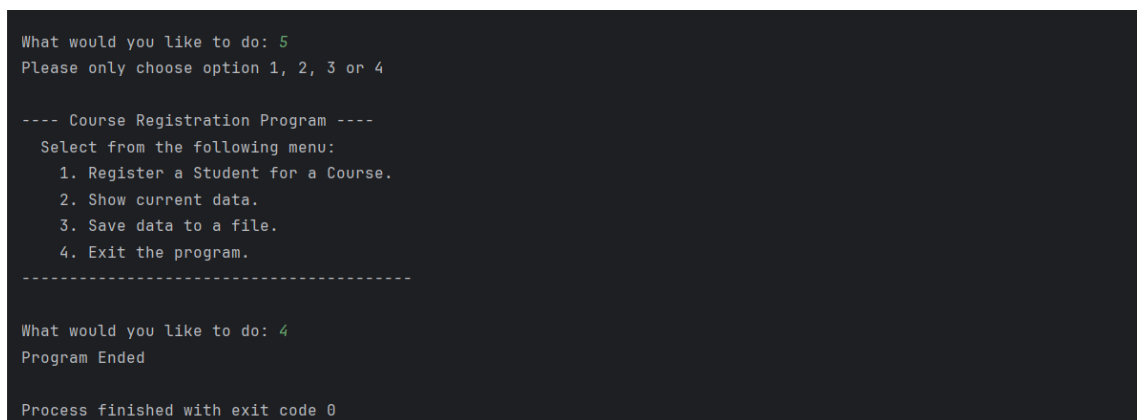The program runs correctly in both **PyCharm and The Command Prompt** (Fig 7-14).

Figure 10. Testing the script in PyCharm for menu 4 and other choices.

```
Command Prompt - python a    ×    +    ∨

Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mineh>cd Documents

C:\Users\mineh\Documents>cd PycharmProjects

C:\Users\mineh\Documents\PycharmProjects>cd assignment05_mine

C:\Users\mineh\Documents\PycharmProjects\assignment05_mine>python assignment05_minewernsing.py
Enrollments.csv file not found.
---Technical Information---
[Errno 2] No such file or directory: 'Enrollments.csv'
File not found.
<class 'FileNotFoundError'>

---- 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
Please enter the name of the course: Python

---- 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: |
```

Figure 11. Testing the script in Command Prompt for menu 1 choice.


```
What would you like to do: 2
----------------------------------------------------
Student Sue Salias is enrolled in Python
Student Bob Baker is enrolled in Python
----------------------------------------------------

---- 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: |
```

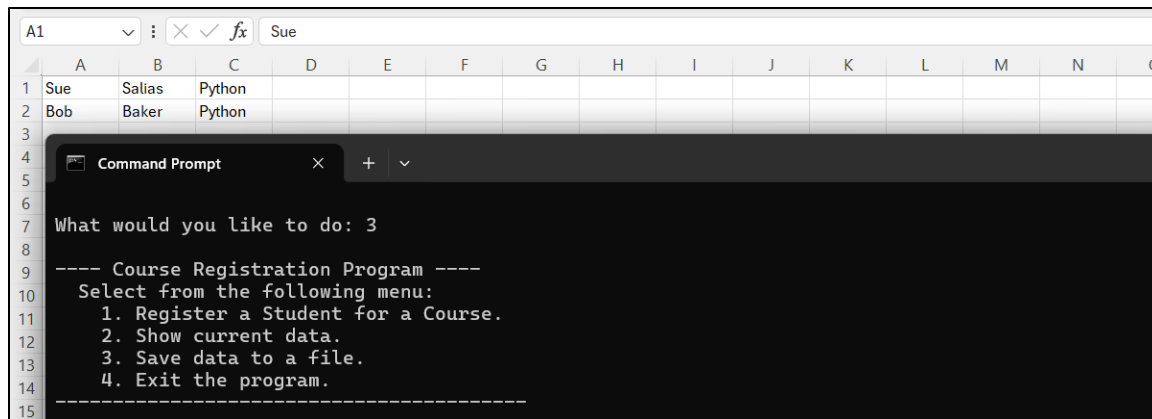Figure 12. Testing the script in Command Prompt for menu 2 choice.

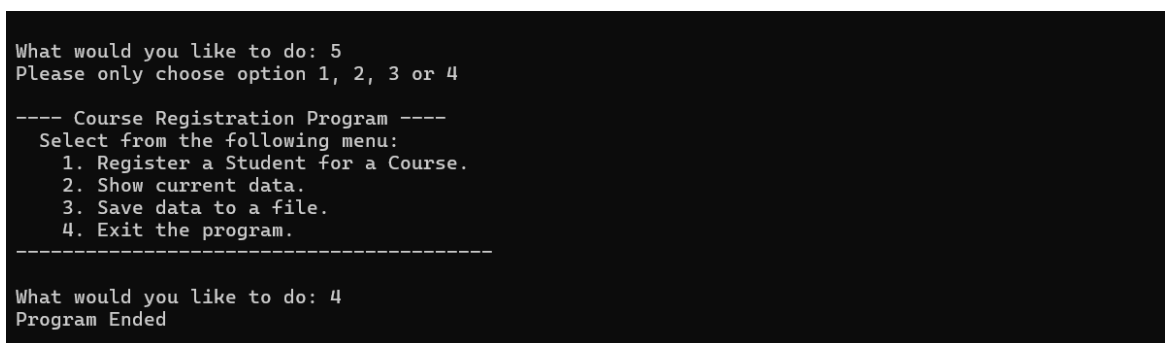Figure 13. Testing the script in Command Prompt for menu 3 choice.

Figure 14. Testing the script in Command Prompt for menu 4 and other choices.

Source Control

The script file and the knowledge document are hosted on a GitHub repository.

A link to the repository is below:

Mine Wernsing

5/12/2024

IT FDN110 B-A05

https://github.com/MineWernsing/IntroToProg-Python

A link to the repository is in the GitHub links forum below:

Topic: Assignment 05 Documents for Review! (uw.edu)

Summary

I practiced processing data using dictionaries and exception handling with a Python program I created that demonstrated using constants, variables, and print statements to display a message about a student's registration for a Python course that handled errors with exceptions that prevented the code from crashing. I tested and ran the script smoothly in both PyCharm and Command Prompt.