Andrew Yarberry

11/14/2024

ITD FDN 110B Foundations of Programming: Python

Module 05

https://github.com/Finviel314/Python110

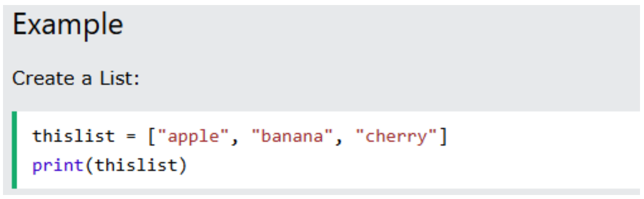
Assignment 05 Using Dictionaries and Including Error Handling

Introduction

For this assignment, we switch our program to read data into a list of dictionaries instead of a list of lists. Similarly, we take inputs from the user by adding them to a dictionary. I also include some error handling to assist with diagnosing exceptions and common errors that can occur with our code.

Dictionaries

A dictionary operates essentially as a list but is ordered by keys rather than indexes. This creates a defined logic for how the dictionary must manage and store data, further streamlining and simplifying how we can write our code. For example, below in Figure 1, we can see an unordered list presented. This list has an implicit index starting at zero and counting by one for each item in the list.

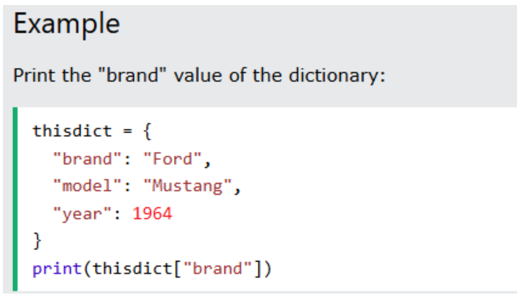

Example

Create a List:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

Figure 1: List Example[1]

Now for a dictionary, we must instead provide a key by which the data will be ordered. For the example below in Figure 2 we can see that they are providing the key brand, model, and year to order the data input into the dictionary. By ordering the data in this way it will simplify how we can call upon it for later use.

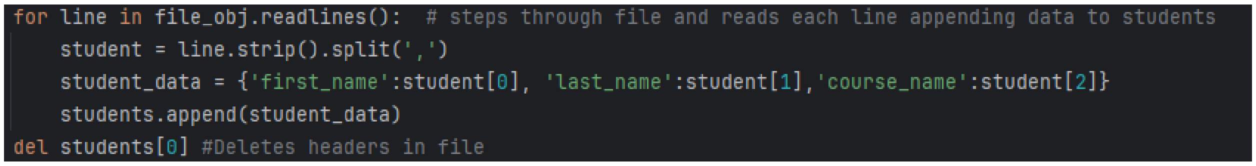

Example

Print the "brand" value of the dictionary:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict["brand"])
```

***Figure 2: Split Method Example*[2]**

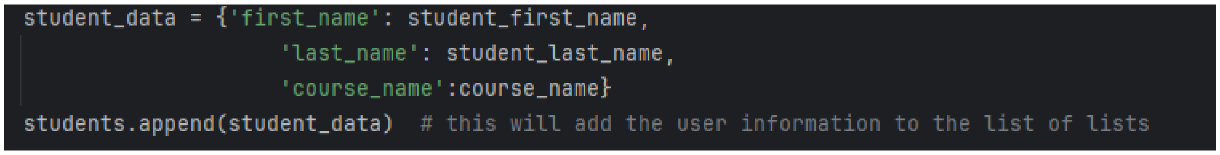
Now for my code, I first need to read data from a file and store it in a dictionary. I utilized a loop and built-in method called 'readlines' to march through the data in my file. The red line is stored in a temporary unordered list and then converted to our variable 'student_data' as a dictionary with key terms or first name, last name, and course name. Then I append the dictionary into a list of dictionaries called 'students. (Figure 3)



```
for line in file_obj.readlines():  # steps through file and reads each line appending data to students
    student = line.strip().split(',')
    student_data = {'first_name':student[0], 'last_name':student[1],'course_name':student[2]}
    students.append(student_data)
del students[0] #Deletes headers in file
```

Figure 3: Read CSV to Dict

Similarly, when taking inputs from the user we can input the variables into the dictionary using the previously established keys as shown in Figure 4.

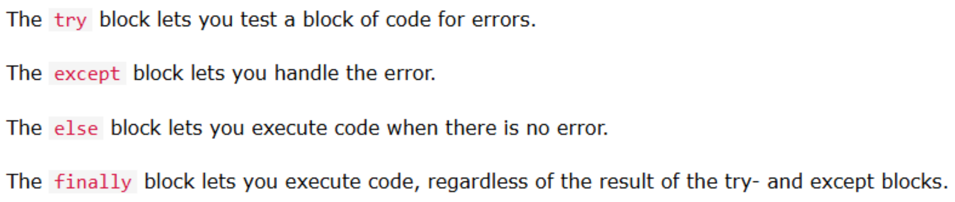


```
student_data = {'first_name': student_first_name,
                'last_name': student_last_name,
                'course_name':course_name}
students.append(student_data)  # this will add the user information to the list of lists
```

Figure 4: User Input to Dict

Error Handling

For this script, we were also tasked with including some ordered error handling. This is helpful because if we expect certain errors streamlines our understanding of what is not working within the program. Code can be tested in situ using 'Try Except' blocks.



The `try` block lets you test a block of code for errors.

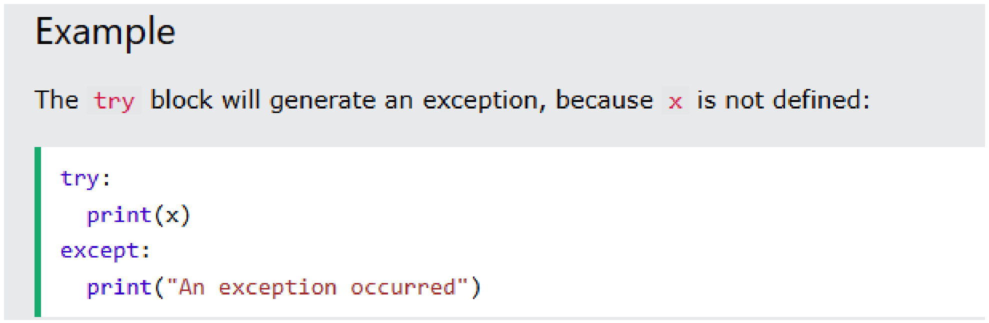
The `except` block lets you handle the error.

The `else` block lets you execute code when there is no error.

The `finally` block lets you execute code, regardless of the result of the try- and except blocks.

***Figure 3: Definitions of try, except, else, finally*[3]**

When an error occurs within our code it throws an exception, we try and except we can catch the exception and use it to inform the user of what occurred. (Figure 4)


Example

The `try` block will generate an exception, because `x` is not defined:

```
try:
  print(x)
except:
  print("An exception occurred")
```

Figure 4: try and except block example[3]

To handle errors when reading our file, I used a try and except block to handle specifically ‘file not found’ errors but also to provide feedback and documentation for any exception the program might through. (Figure 5)

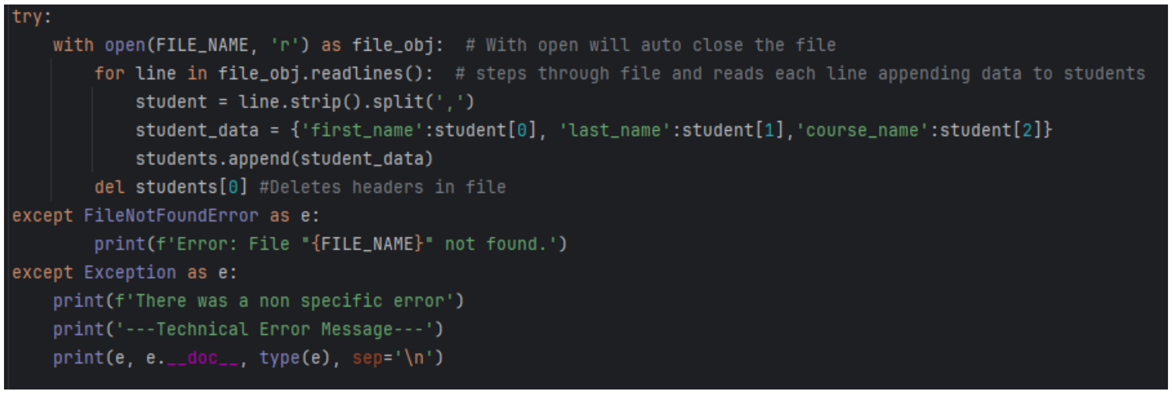
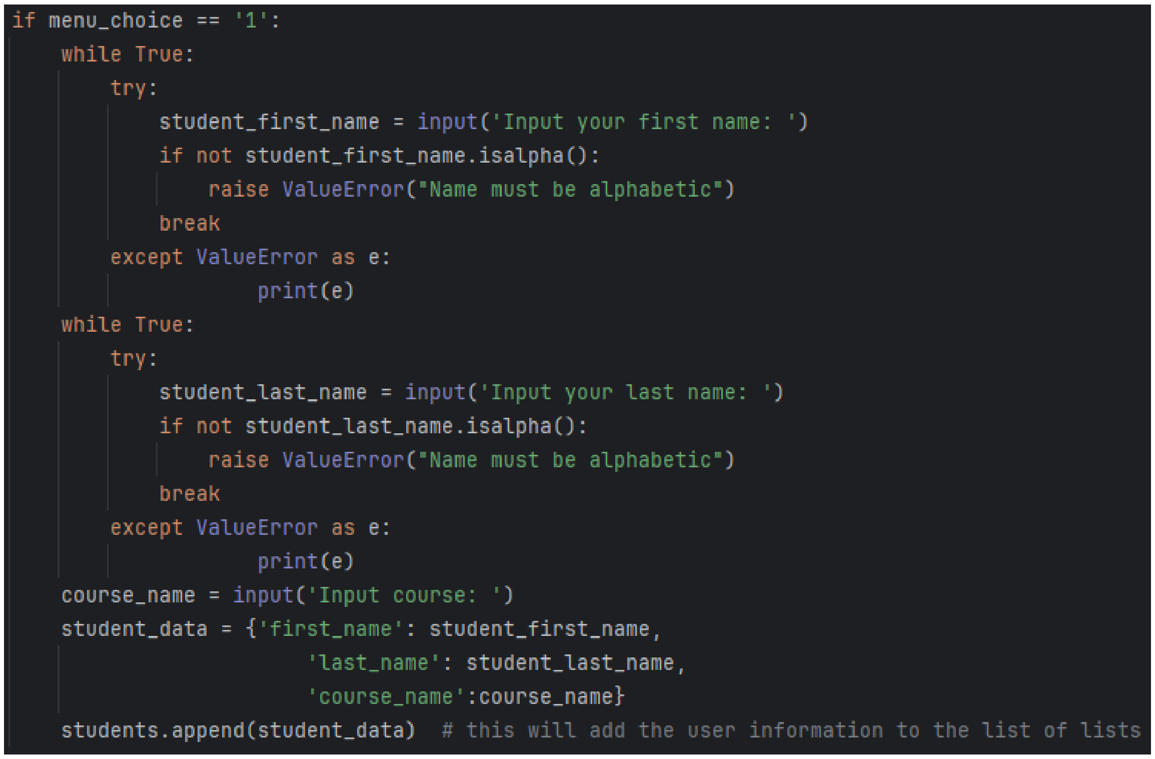

```
try:
    with open(FILE_NAME, 'r') as file_obj:  # With open will auto close the file
        for line in file_obj.readlines():  # steps through file and reads each line appending data to students
            student = line.strip().split(',')
            student_data = {'first_name':student[0], 'last_name':student[1],'course_name':student[2]}
            students.append(student_data)
        del students[0] #Deletes headers in file
except FileNotFoundError as e:
        print(f'Error: File "{FILE_NAME}" not found.')
except Exception as e:
    print(f'There was a non specific error')
    print('---Technical Error Message---')
    print(e, e.__doc__, type(e), sep='\n')
```

Figure 5: CSV read error handling

For this assignment, we specifically needed to provide error handling for instances where the user did not input alphabetic responses for the student's first and last name for this we need nest the try and except block inside a ‘while True’ loop so it will iterate over the single block of code until a valid response is given.

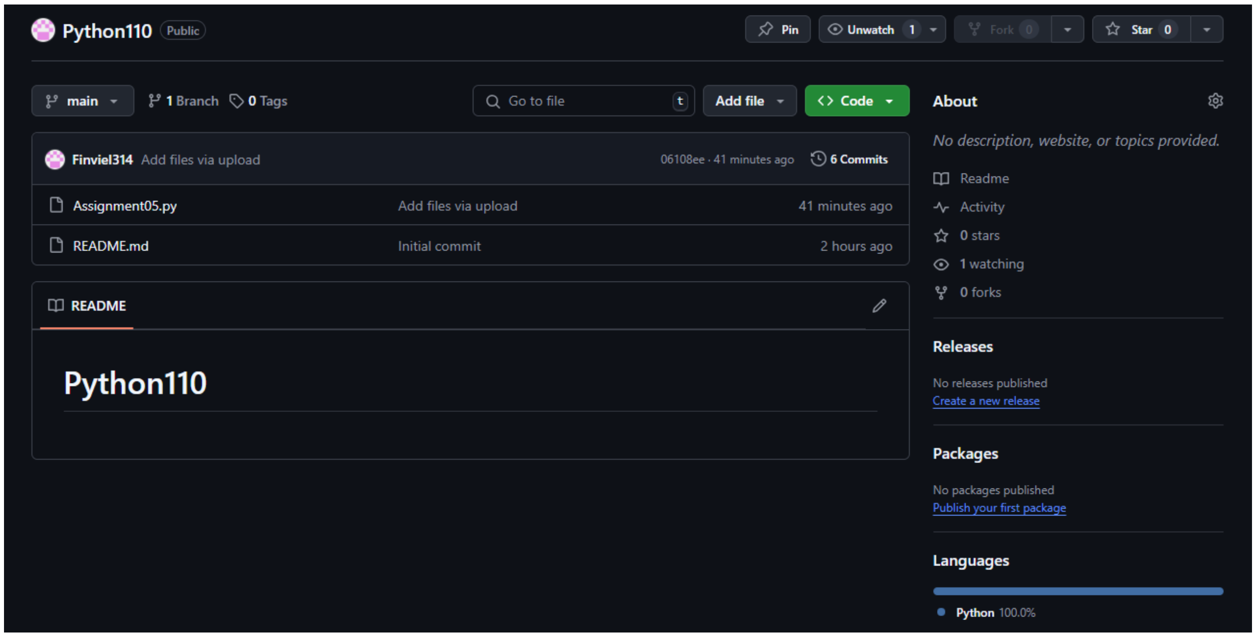


```
if menu_choice == '1':
    while True:
        try:
            student_first_name = input('Input your first name: ')
            if not student_first_name.isalpha():
                raise ValueError("Name must be alphabetic")
            break
        except ValueError as e:
                print(e)
    while True:
        try:
            student_last_name = input('Input your last name: ')
            if not student_last_name.isalpha():
                raise ValueError("Name must be alphabetic")
            break
        except ValueError as e:
                print(e)
    course_name = input('Input course: ')
    student_data = {'first_name': student_first_name,
                    'last_name': student_last_name,
                    'course_name':course_name}
    students.append(student_data)  # this will add the user information to the list of lists
```

Figure 6: User input error handling

GitHub

I created a Git account and started a repository for Python 110. GitHub is a website that allows you to post your code but more importantly, to revision control your information.



Finviel314 Add files via upload 06108ee · 41 minutes ago 6 Commits

Assignment05.py	Add files via upload	41 minutes ago
README.md	Initial commit	2 hours ago

Python110

About

No description, website, or topics provided.

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

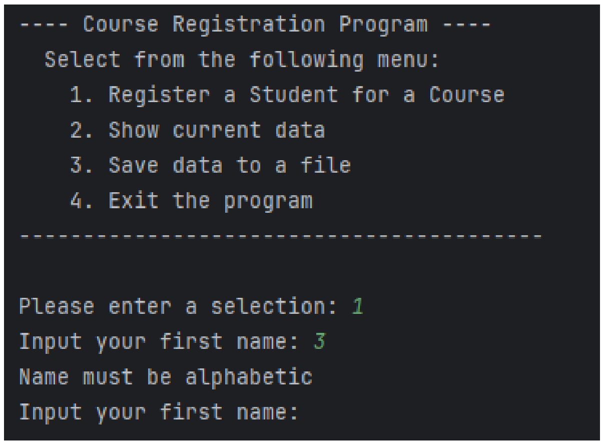
Languages

Python 100.0%

Figure 7: Git Repository for Python 110

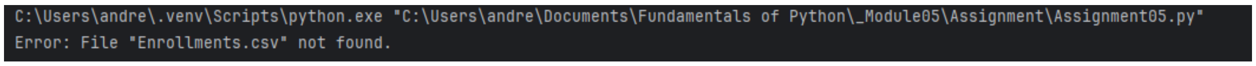
Testing the Program

I was able to successfully open and run the program in both PyCharm and CMD. Error handling works as can be seen in Figures 8 and 9 respectively.



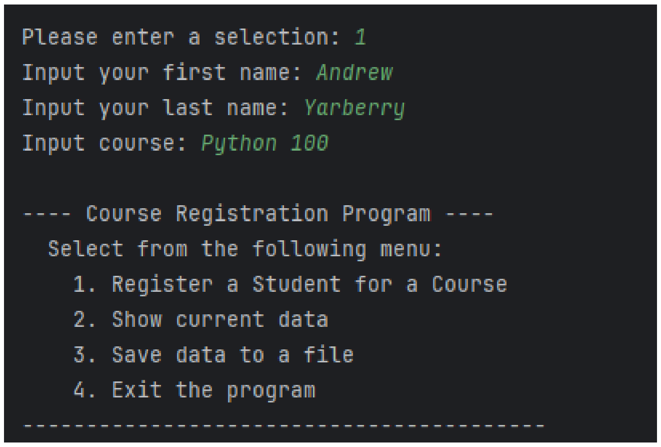
```
---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course
    2. Show current data
    3. Save data to a file
    4. Exit the program
-----------------------------------------


Please enter a selection: 1
Input your first name: 3
Name must be alphabetic
Input your first name:
```

Figure 8: Error Handling Example



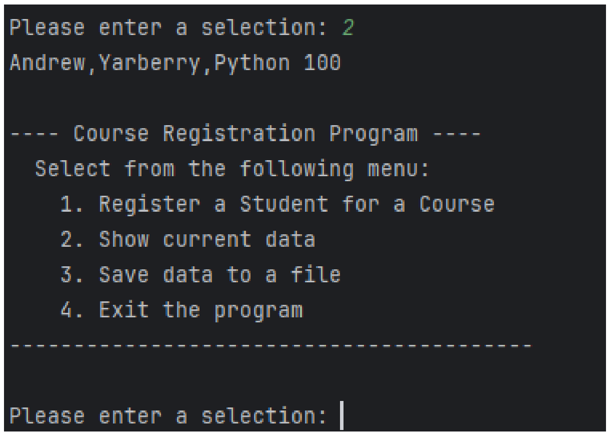
```
C:\Users\andre\.venv\Scripts\python.exe "C:\Users\andre\Documents\Fundamentals of Python\_Module05\Assignment\Assignment05.py"
Error: File "Enrollments.csv" not found.
```

Figure 9: File Not Found Error Handling



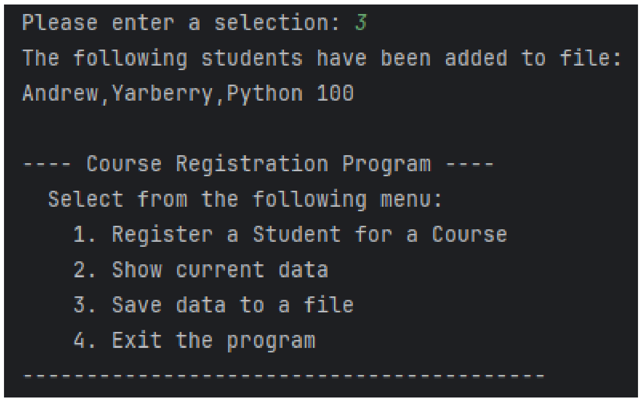
```
Please enter a selection: 1
Input your first name: Andrew
Input your last name: Yarberry
Input course: Python 100

---- 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: Selection 1 Output PyCharm



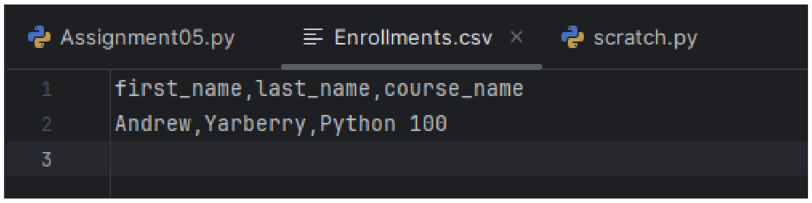
```
Please enter a selection: 2
Andrew,Yarberry,Python 100

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

Please enter a selection: 
```

Figure 11: Selection 3 Output PyCharm



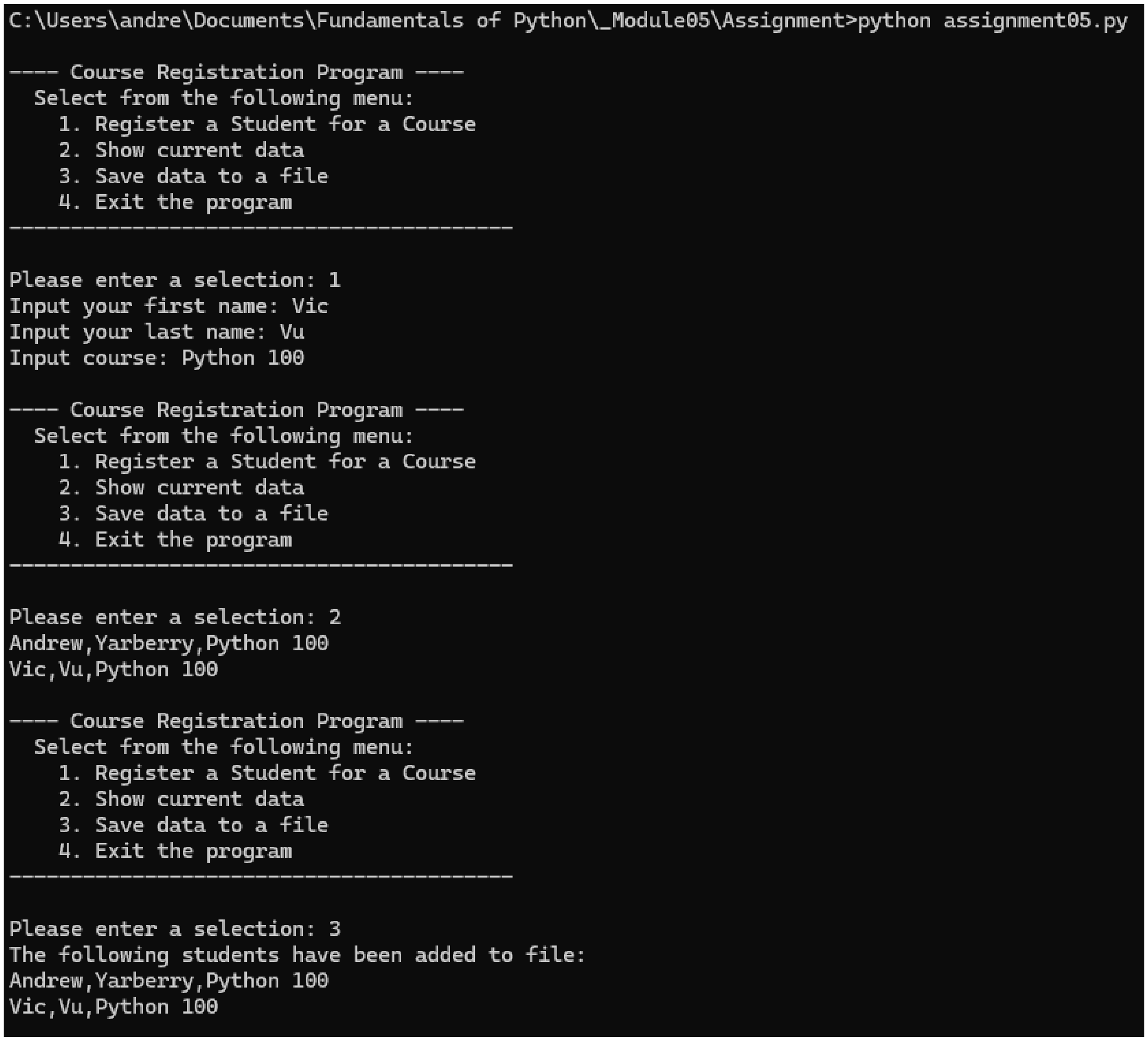
```
Please enter a selection: 3
The following students have been added to file:
Andrew,Yarberry,Python 100

---- 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: Selection 3 Output PyCharm



```
first_name,last_name,course_name
Andrew,Yarberry,Python 100
```

Figure 13: Selection 3 Output PyCharm



```
C:\Users\andre\Documents\Fundamentals of Python\_Module05\Assignment>python 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
-----------------------------------------

Please enter a selection: 1
Input your first name: Vic
Input your last name: Vu
Input course: Python 100

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

Please enter a selection: 2
Andrew,Yarberry,Python 100
Vic,Vu,Python 100

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

Please enter a selection: 3
The following students have been added to file:
Andrew,Yarberry,Python 100
Vic,Vu,Python 100
```

Figure 14: CMD Output

Summary

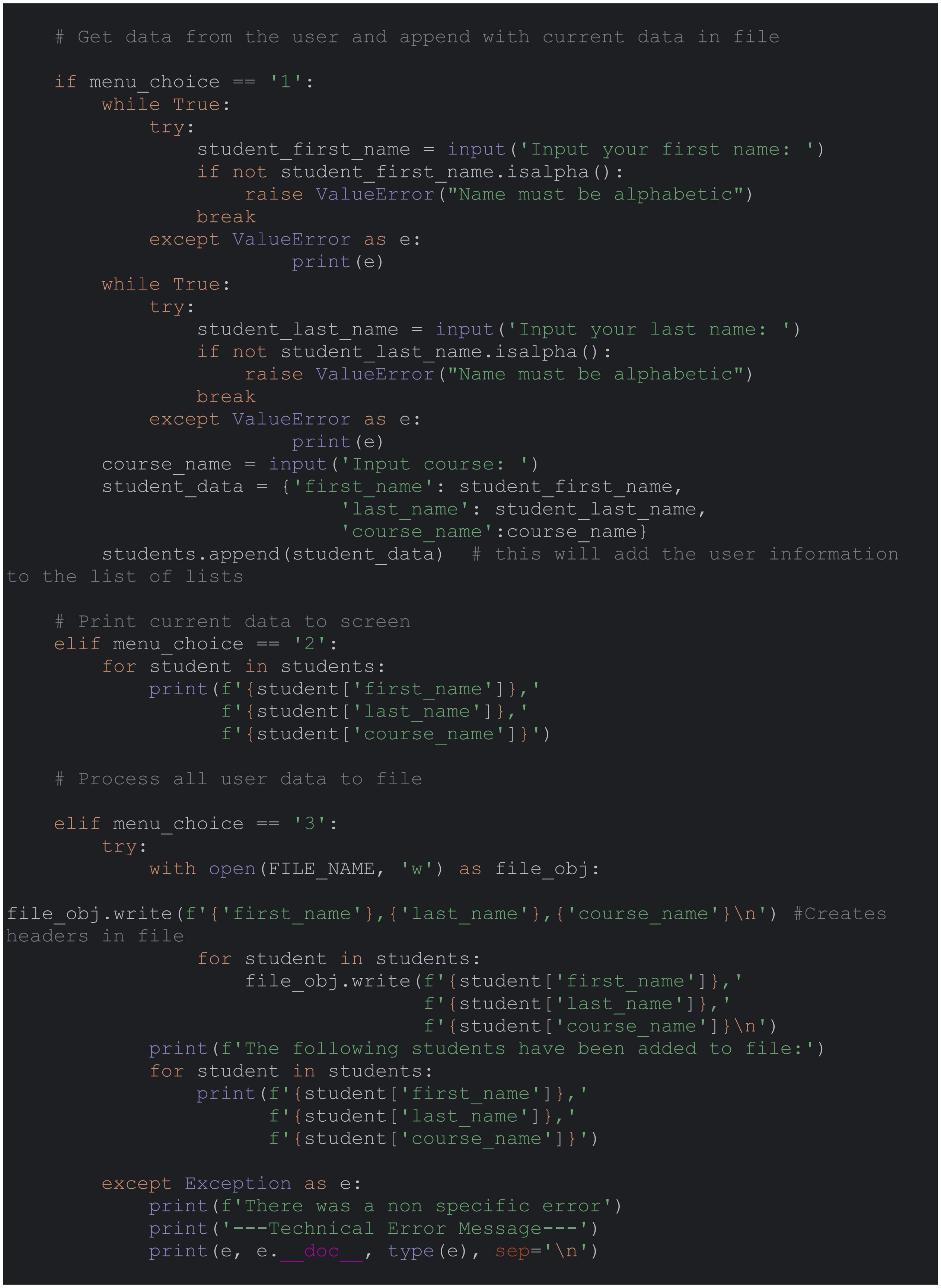
In summary, this program builds on previously discussed concepts and improves our ability to handle and pass data. We learned how to open and read data from a file and pass that data into a dictionary. This further increases the ease of handling data withed keyed entries. We also included error handling to help us or the user understand what exceptions are being thrown.

References

1. *W3 Schools, https://www.w3schools.com/python/python_lists.asp, 2024) (External Site)*
2. *W3 Schools, https://www.w3schools.com/python/python_dictionaries.asp) (External Site)*
3. *W3 Schools, https://www.w3schools.com/python/python_try_except.asp) (External Site)*
4. *github, https://github.com/Finviel314/Python110 (External Site)*

Appendix

```
# ---------------------------------------------------------------------------
--------------- #
# Title: Assignment05
# Desc: This assignment demonstrates using a dictionary instead of lists and
adds error handling
# Change Log: (Who, When, What)
#   #  Andrew Yarberry, 11/07/2024, Initial Release

# ---------------------------------------------------------------------------
--------------- #

# 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
-----------------------------------------
'''
FILE_NAME: str = 'Enrollments.csv'

# Define the Data Variables
menu_choice: str = ''
student_first_name: str = ''
student_last_name: str = ''
course_name: str = ''
student_data: dict[str,str] = {}
students: list = []
csv_data: str = ''

try:
    with open(FILE_NAME, 'r') as file_obj:  # With open will auto close the
file
        for line in file_obj.readlines():  # steps through file and reads
each line appending data to students
            student = line.strip().split(',')
            student_data = {'first_name':student[0],
'last_name':student[1],'course_name':student[2]}
            students.append(student_data)
        del students[0] #Deletes headers in file
except FileNotFoundError as e:
        print(f'Error: File "{FILE_NAME}" not found.')
except Exception as e:
    print(f'There was a non specific error')
    print('---Technical Error Message---')
    print(e, e.__doc__, type(e), sep='\n')

# Present Menu of Choices
while True:
    print(MENU)
    menu_choice = input('Please enter a selection: ')
```



```
    # Get data from the user and append with current data in file

    if menu_choice == '1':
        while True:
            try:
                student_first_name = input('Input your first name: ')
                if not student_first_name.isalpha():
                    raise ValueError("Name must be alphabetic")
                break
            except ValueError as e:
                        print(e)
        while True:
            try:
                student_last_name = input('Input your last name: ')
                if not student_last_name.isalpha():
                    raise ValueError("Name must be alphabetic")
                break
            except ValueError as e:
                        print(e)
        course_name = input('Input course: ')
        student_data = {'first_name': student_first_name,
                            'last_name': student_last_name,
                            'course_name':course_name}
        students.append(student_data)  # this will add the user information
to the list of lists

    # Print current data to screen
    elif menu_choice == '2':
        for student in students:
            print(f'{student['first_name']},'
                  f'{student['last_name']},'
                  f'{student['course_name']}')

    # Process all user data to file

    elif menu_choice == '3':
        try:
            with open(FILE_NAME, 'w') as file_obj:

file_obj.write(f'{'first_name'},{'last_name'},{'course_name'}\n') #Creates
headers in file
                for student in students:
                    file_obj.write(f'{student['first_name']},'
                                   f'{student['last_name']},'
                                   f'{student['course_name']}\n')
            print(f'The following students have been added to file:')
            for student in students:
                print(f'{student['first_name']},'
                      f'{student['last_name']},'
                      f'{student['course_name']}')

        except Exception as e:
            print(f'There was a non specific error')
            print('---Technical Error Message---')
            print(e, e.__doc__, type(e), sep='\n')
```

```python
    # Exit program
    elif menu_choice == '4':
        print('Goodbye!')
        exit()

    # handles all other inputs
    else:
        print('Does not compute')
```

Figure 9: Full Python Code