Python 100

Assignment 05

Module 05 – Advanced Collections and Error Handling

Introduction

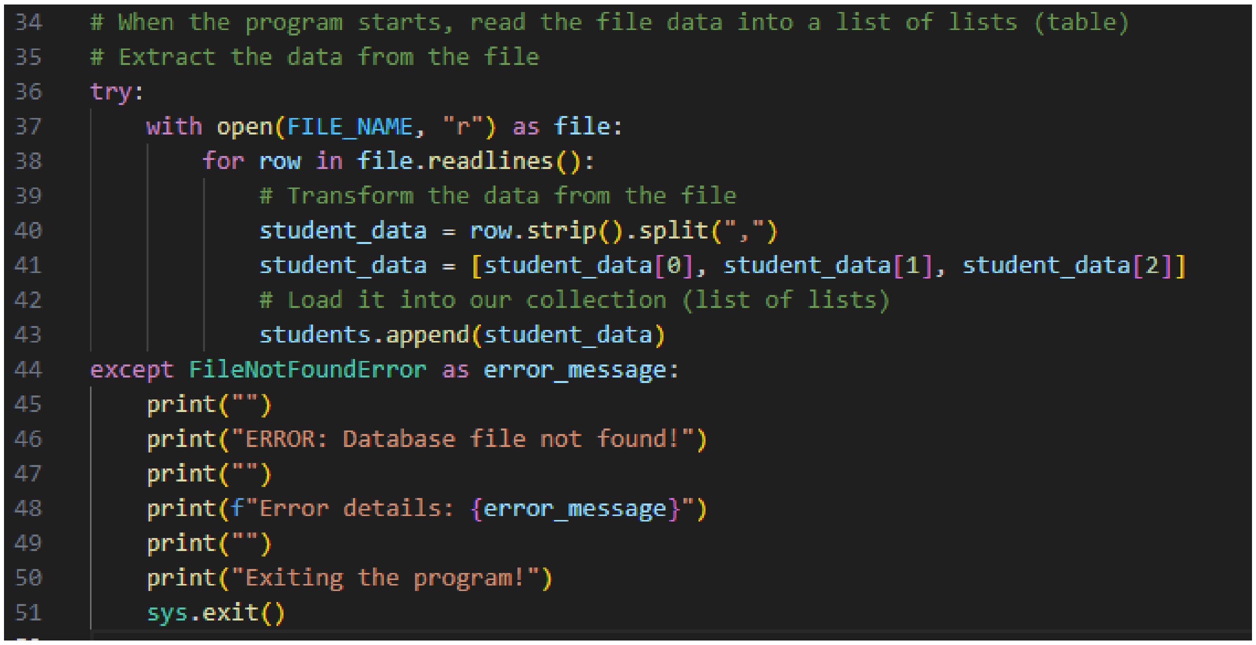
In this document I will describe the steps I took to create a Python script that presents the user with menu choices, takes the user's input and performs a task based on that input. The user provided information is then stored in a dict, presented to the user and saved to a .csv file. This process loops until the user selects to exit the program. This assignment expands on the assignment from Module 04, the student_data list is now student_data dict. We also used structured error handling during the initial read of the csv file, while the user inputs first and last name information and then when the data is written to the csv file.

Creating the script

I began by opening Visual Studio Code and creating a file starting with my code from Assignment04. I then followed the instructions in the Mod05-Assignment file. This assignment focuses on using the dict data type to store sets of data input by the user, as well as handling errors that may occur due to missing files, incorrect data entered and improperly formatted information for the csv.

- Under "# Define the Data Constants" I created a "MENU" constant that is used to present the user with their options, I also created a "FILE_NAME" constant that is used for the .csv file.
- Under "# Define the Data Variables" I created Variables for first and last name, course name, csv_data, file_obj for the writing of data to the .csv file, a variable for the user's menu choice, student_data "dict" and students as a "list" for storing user input.
- Before defining the variables I used "import sys" so that I would be able to use "sys.exit()" to end the loop that I was going to create.
- Before entering the user input and data processing area I used "try:" to open and read the csv file information to students. I used the FileNotFound exception to present an error to the user if there was no data file present, I then presented the user with this error and exited the program.

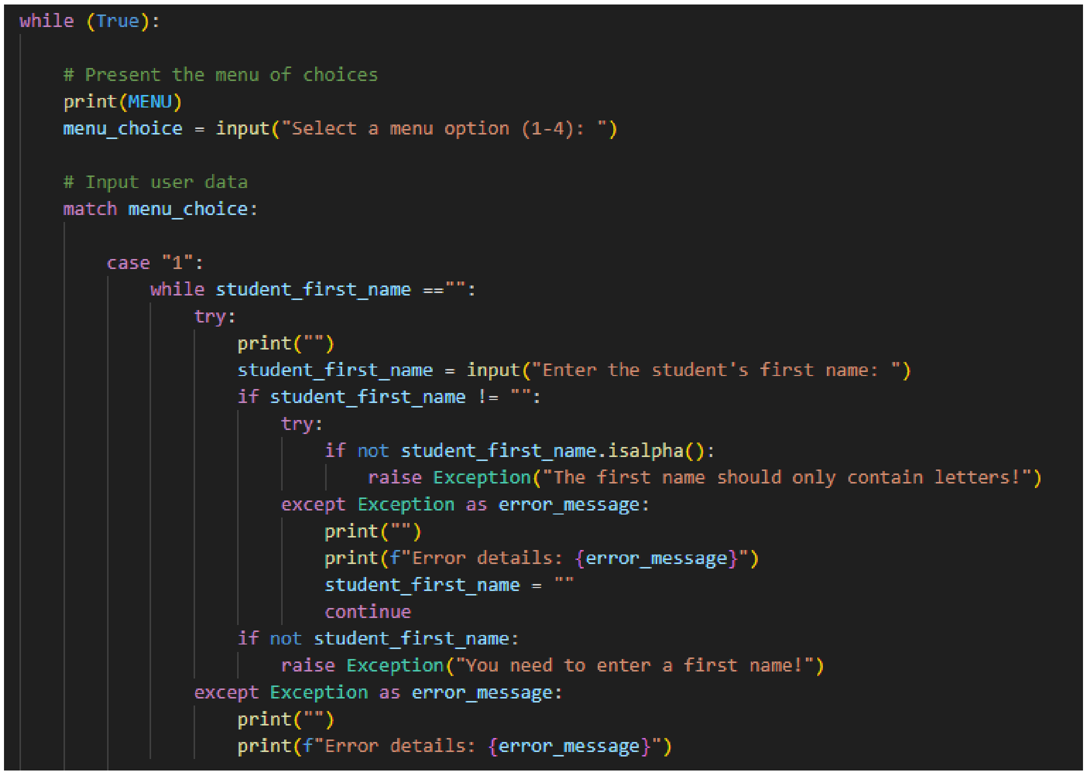
This can be seen in the screenshot below.



```python
# When the program starts, read the file data into a list of lists (table)
# Extract the data from the file
try:
    with open(FILE_NAME, "r") as file:
        for row in file.readlines():
            # Transform the data from the file
            student_data = row.strip().split(",")
            student_data = [student_data[0], student_data[1], student_data[2]]
            # Load it into our collection (list of lists)
            students.append(student_data)
except FileNotFoundError as error_message:
    print("")
    print("ERROR: Database file not found!")
    print("")
    print(f"Error details: {error_message}")
    print("")
    print("Exiting the program!")
    sys.exit()
```

- I then created a “while” loop to present the user with choices.
    - Under ”#Present the menu of choices” I used the print function to present the user with the menu.
    - Under “#Input user data” I set the menu choice variable to the user’s selection with an input function. I then used match with the menu choice variable to perform different actions based off the user input.

- For case 1 the user is presented with requests for input to be stored in variables for first name, last name and course name. I then append this information to the students variable. I also added error handling to the case “1” inputs for first and last name. I was looking for empty strings and numbers entered instead of letters. This can be seen in the screenshot below.



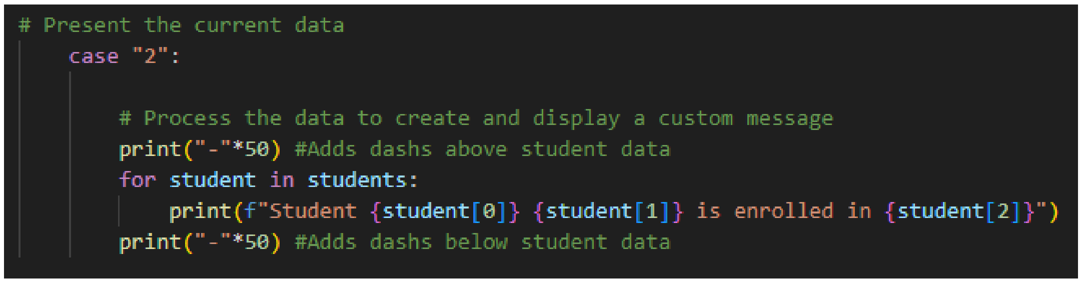
```python
while (True):

    # Present the menu of choices
    print(MENU)
    menu_choice = input("Select a menu option (1-4): ")

    # Input user data
    match menu_choice:

        case "1":
            while student_first_name =="":
                try:
                    print("")
                    student_first_name = input("Enter the student's first name: ")
                    if student_first_name != "":
                        try:
                            if not student_first_name.isalpha():
                                raise Exception("The first name should only contain letters!")
                        except Exception as error_message:
                            print("")
                            print(f"Error details: {error_message}")
                            student_first_name = ""
                            continue
                    if not student_first_name:
                        raise Exception("You need to enter a first name!")
                except Exception as error_message:
                    print("")
                    print(f"Error details: {error_message}")
```

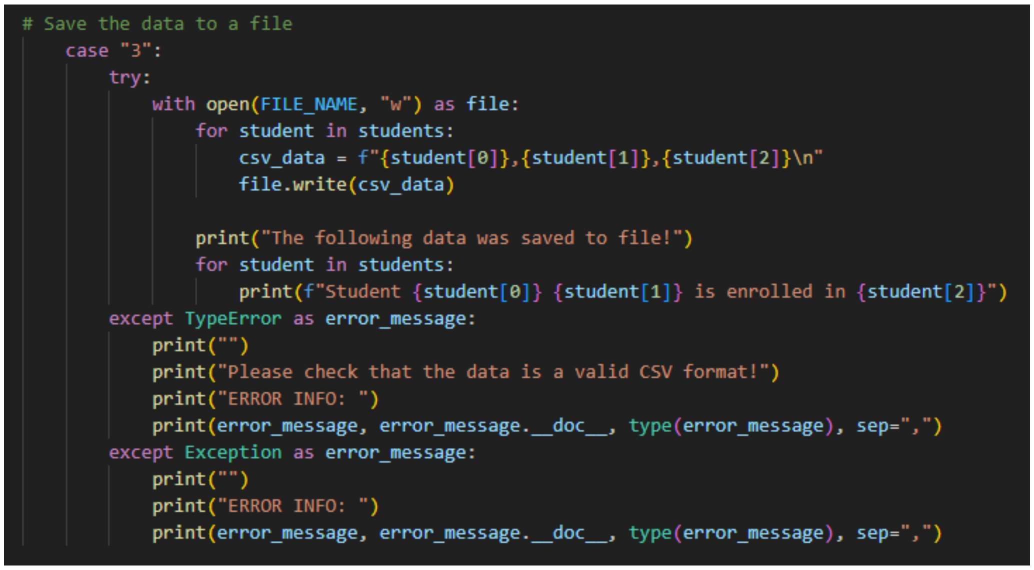
- For case 2 the data stored in students is presented to the user with a for loop and print command. In a for loop I extract the first name, last name and course name and then use a formatted print to display this information. This is seen in the image below.



```python
# Present the current data
    case "2":

        # Process the data to create and display a custom message
        print("-"*50) #Adds dashs above student data
        for student in students:
            print(f"Student {student[0]} {student[1]} is enrolled in {student[2]}")
        print("-"*50) #Adds dashs below student data
```

- Under “#Save the data to a file” For case 3, I use “try” again to execute some error handling to check csv formatting. I also wrote the data stored in the variables to a csv file. Here I also use a for loop to extract the data, format it and then write it to the csv file. Each entry is also presented to the user with the print statement. Shown in the image below.



```
# Save the data to a file
    case "3":
        try:
            with open(FILE_NAME, "w") as file:
                for student in students:
                    csv_data = f"{student[0]},{student[1]},{student[2]}\n"
                    file.write(csv_data)

                print("The following data was saved to file!")
                for student in students:
                    print(f"Student {student[0]} {student[1]} is enrolled in {student[2]}")
        except TypeError as error_message:
            print("")
            print("Please check that the data is a valid CSV format!")
            print("ERROR INFO: ")
            print(error_message, error_message.__doc__, type(error_message), sep=",")
        except Exception as error_message:
            print("")
            print("ERROR INFO: ")
            print(error_message, error_message.__doc__, type(error_message), sep=",")
```

- Under “#Stop the loop” For case 4, I used print to inform the user the program was exiting and then used sys.exit() to end the loop/program.
- Under “#In the event of something entered other than (1-4)” I used “case _:” to display a message that the user made an invalid selection. The loop starts over from here.

Testing the script

I tested the script using the following methods.

- In Visual Studio Code I ran the script in the Terminal and checked the output of the print statement Vs. what had been entered.
- I checked the folder I was working in for the .csv file that was created. I opened this file in Notepad and in Excel.

Summary

Using the class demonstrations, my assignment from module 4 and documents provided for module 5 I was able to create a script that takes user input, presents it to the user and saves it into a .csv file. I expanded my ability to create a loop, take a user’s input, save it to file and add to that file as the user inputs more data. I also learned how to check for certain error conditions and present the error information to the user.