Jeremy Peters
November 5, 2024
IT FDN 110 B
Assignment 05
Files in GitHub

Dictionaries, JSON files, Exceptions & GitHub

Introduction

This lesson taught me that dealing with multiple nested *while* and *if* statements with custom error handling can quickly become complex! We also learned about nesting *dictionaries* inside *lists*, which we used for this assignment when building the list written to CSV. Interestingly, while the lesson discussed working with JSON files, the assignment did not leverage any of that. Finally, the assignment concludes with us uploading our assignment files to a GitHub repository.

Variables and Declarations

As this assignment was built on the previous one, the constants and variables defined at the beginning of the file mostly remained the same. I added some variables for my custom messages, which were used at input prompts and for custom error messages.

```
# Custom message variables
prompt_firstname: str = f"Please enter the student's first name: "
prompt_lastname: str = f"Please enter the student's last name: "
no_data: str = f"You have not entered any data.\n" \
               f"Try starting with starting option 1."
alpha_only: str = f"Student name should only contain alphabetic characters."
ascii_only: str = f"Course name should only contain ascii characters."
```

Wrapping code in try:

I started my program execution code with a *try:* statement to check for an existing CSV file. If the file isn't present, I create it with an empty value so that later code does not fail when attempting to read from it.

```
# Open and reset the CSV file back to a base value for this scenario
# These two lines can be commented out if there's an existing file with data
try:
    print(f"Checking for existing file {FILE_NAME}...")
    if os.path.exists(FILE_NAME) and os.path.getsize(FILE_NAME) > 0:
        raise FileExistsError(warning_color.format(file_exists))
    with open(FILE_NAME, "w") as file:
        print(f"No existing file {FILE_NAME} found. File will be created.")
        file.write("")
except FileExistsError as e:
    # print(error_color.format("File already exists and has content.\n"))
    # print(error_color.format("-- Error Message -- "))
    print(error_color.format( *args: e, e.__doc__, type(e), sep="\n"))
finally:
    if file in locals() and not None and not file.closed:
        file.close()
```

When data is found, the program first reads the contents of the file and, using a *for* loop, adds the data to a list, then formats it as a dictionary before adding the data values to the **students** variable via an *.append()* method.

Exception Handling

Due to the way I wanted the program to run, I nested the try and except statements deep in the first section of code. This is because I wanted the program to retain the data input if the first name was entered properly, then fail and retry on the last name if invalid characters were found. I used the *.isalpha()* method to ensure that only alphabetic characters were entered for both the first and last names.

```
        # Input user data
        if menu_choice == "1":
            try:
                while True:
                    try:
                        student_first_name = input(husky_purple.format(
                            prompt_firstname))
                        if not student_first_name.isalpha():
                            raise ValueError(
                                warning_color.format(f"{alpha_only}"))
                        student_first_name = student_first_name.title().strip()
                        break
                    except ValueError as e:
                        print(error_color.format("-- Error Message -- "))
                        print(error_color.format(e.__doc__))
                        print(error_color.format(e))
```

If invalid characters are entered for either the student_first_name or student_last_name variables, I catch it as an exception and provide an error to the user. The code then retries the last field until valid characters are entered. I wanted to do something similar for the course_name variable, but the following methods all had odd behaviors that didn't meet my needs.
.isascii() = includes special characters that I wanted to consider invalid
.isalnum() = did not allow for "Python 100" as it has both alpha and numeric characters
.isprintable() = did not allow for any spaces in the text entered

Once the data is entered with the appropriate characters, I write the data collected to the **student_data** dictionary before appending it to the **students** *list*, printing the data collected, and returning the program to the menu.

```
                student_data = {"FirstName": student_first_name,
                                "LastName": student_last_name,
                                "CourseName": course_name}
                students.append(student_data)
```

I added exception handler code to menu choices 2 and 3, raising a *ValueError* if the user proceeds to either of these options before entering data in menu choice 1.

```
        # Present the current data
        elif menu_choice == "2":
            try:
                # print(students)
                if student_first_name == str():
                    raise ValueError(warning_color.format(no_data))
                else:
                    print(husky_gold.format(
                        f"The following students are registered:"))
                    for student in students:
                        print(husky_purple.format(
                            f"{student["FirstName"]},{student["LastName"]} is "
                            f"enrolled in {student["CourseName"]}"))
            except ValueError as e:
                print(error_color.format("-- Error -- "))
                # print(error_color.format(e.__doc__))
                print(error_color.format(e))
```

The added color coding to makes the errors more visible in the console.

```
E:\OneDrive\Documents\UW_IT-FDN-110\_Module05\Assignment>python .\Assignment05_JeremyPeters.py
Checking for existing file Enrollments.csv...
File Enrollments.csv already exists. Skipping file creation.

---- 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
-- Error --
You have not entered any data.
Try starting with starting option 1.

---- 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
-- Error --
You have not entered any data.
Try starting with starting option 1.

---- 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: 
```

Similar to the last assignment, when menu option 3 is chosen, we write the data to a CSV file. However, this time we're iterating through the **students** *list* of *dictionaries*, collecting that as a *list* type in the **csv_data** variable before writing that to the CSV file.

```
        # Save the data to a file
        elif menu_choice == "3":
            try:
                if student_first_name == str():
                    raise ValueError(warning_color.format(no_data))
                else:
                    print(
                        husky_gold.format(f"The following was saved to file:"))
                    with open(FILE_NAME, "w") as file:
                        for student in students:
                            csv_data += (
                                f"{student["FirstName"]},{student["LastName"]}"
                                f",{student["CourseName"]}\n")
                            print(husky_purple.format(f"{student["FirstName"]}\
{student["LastName"]} is enrolled in {student["CourseName"]}"))
                        file.write(csv_data)
            except ValueError as e:
                print(error_color.format("-- Error -- "))
                # print(error_color.format(e.__doc__))
                print(error_color.format(e))
            except Exception as e:
                print(error_color.format("-- Error Message -- "))
                print(error_color.format( *args: e, e.__doc__, type(e), sep="\n"))
                print(error_color.format("There was a non-specific error!\n"))
```

Summary

I liked this assignment because it forced me outside of my comfort zone. I realize that we're learning, which is why this code looks so "messy" with a lot of code duplication, but it made it very hard for me to wrap my head around the concept. I can see why error handling is usually coded into a separate class or function that you can then call with a single statement. Doing so would make this otherwise simple program look less busy.

```
Command Prompt

E:\OneDrive\Documents\UW_IT-FDN-110\_Module05\Assignment>python .\Assignment05_JeremyPeters.py
Checking for existing file Enrollments.csv...
File Enrollments.csv already exists. Skipping file creation.

---- 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
Please enter the student's first name: bill
Please enter the student's last name: 3gates
-- Error --
Student name should only contain alphabetic characters.
Please enter the student's last name: gates
Please enter the course name: python 100
You have added Bill Gates for course Python 100 to the registration list.

---- 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
The following students are registered:
Vic,Vu is enrolled in Python 100
Bill,Gates is enrolled in 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
----------------------------------------

What would you like to do: 3
The following was saved to file:
Vic Vu is enrolled in Python 100
Bill Gates is enrolled in 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
----------------------------------------

What would you like to do: 4


E:\OneDrive\Documents\UW_IT-FDN-110\_Module05\Assignment>
```

```
Enrollments.csv
File   Edit   View

Vic,Vu,Python 100
Bill,Gates,Python 100

Ln 1, Col 1   40 characters   100%   Windows (CRLF)
```