Ryan Seng

5/13/2024

Python 100

Assignment 05

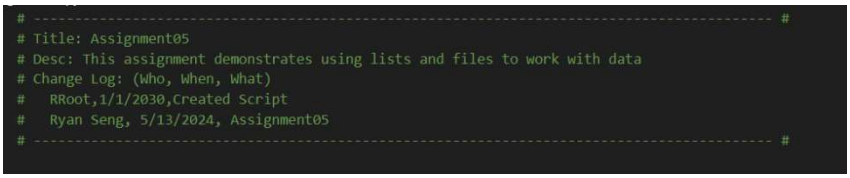
Assignment 05 – Advanced Collections & Error Handling

Introduction

This document will outline how I created the assignment05.py script to turn in for the assignment. To create this code, I referenced the lessons learned in the class on 5/8/2024. The methodology section below goes into further detail.

Methodology

The comments for the assignment were taken straight from my submission for Assignment 04 but I changed <Your Name Here>, <Date>, and Assignment to my name, Ryan Seng, 5/13/2024, and Assignment05 which was when I initially started working on the assignment. The snip below contains the text I was referring to:



Similarly, a lot of the code in this assignment was copied over from Assignment 04. For example, I set the constant, MENU, as a string using ": str" and equal to a set of text (see picture below) that is supposed to be a menu. I also set the constant, FILE_NAME, as a string with ": str" and equal to "Enrollments.csv". I set the variables, student_first_name, student_last_name, course_name, csv_data, and menu_choice as strings using ": str" and set them all to blank or "". For the variable, file, I did not set its type and I set it equal to "None". Lastly, I set the variables, student_data and students, as a dictionary and as a list respectively using ": dict" and ": list" and set them as blank with "[]". Below is a picture of all the constants and variables as described.

```
# This is the MENU users will be selecting from
MENU = """
---- 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
-----------------------------------------
"""

# This are the other constants
FILE_NAME: str = "Enrollments.csv"

# These are the variables
student_first_name: str = ""
student_last_name: str = ""
course_name: str = ""
csv_data: str = ""
file = None
menu_choice: str = ""
student_data: dict = []
students: list = []
```

For the next part of the assignment, I need to load the data in from Enrollments.csv, prompt the user to select an option on the menu, and have the program act accordingly to the program while having error handling for various situations.

First, I needed to make sure that the Python checked to make sure “Enrollments.csv” existed and only ran if it could load the file and data properly. If the file couldn’t be found, then it’d print “Error: File not found”. If there were issues with loading the data, it’d print “Error: There are corruption issues in {FILE_NAME}”. Lastly, if there was an error I missed, it would print “Error:” along with the details of error. The picture below is the code for these exceptions/errors.

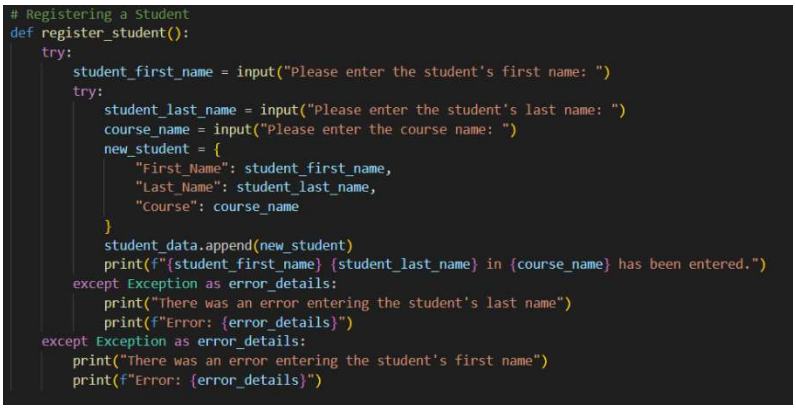
```
except FileNotFoundError:
    print("Error: File not found")
except ValueError:
    print(f"Error: There are corruption issues in {FILE_NAME}")
except Exception as error_details:
    print(f"Error: {error_details}")
```

If everything worked properly, then the data from the file would load. To load the data from Enrollments.csv, I opened the file in read mode with “open(FILE_NAME, ‘r’)” (quotes changed from the actual code to make it easier to read in Word). Then I used the students variable to store the lines of text as lists with the .readlines function. Lastly, I iterated through each list with a “for loop” to clean the data, and remove spaces. In each iteration of the loop, it would define the variable “student_dict” (intended to be a dictionary) with keys “First_Name”, “Last_Name”, and “Course” equal to indexes 0, 1, and 2 of the data. Then I would add the data to create a list of dictionaries to the student_data variable. Below is the code as described.

```
try:
    for user in students:
        cleaned_student_data = user.strip().split(",")
        student_dict = {
            "First_Name": cleaned_student_data[0],
            "Last_Name": cleaned_student_data[1],
            "Course": cleaned_student_data[2]
        }
        student_data.append(student_dict)
```

Again, if the program loaded “Enrollments.csv” properly, then it would run the rest of the program including the menu. To facilitate the menu, I contained all of the following code described in the next section with a “while loop” set to run while True.

For option 1, I ran the “register_student()” function. There, I prompted the user for the student’s first name with an input() function and the prompt, “Please enter the student’s first name: ”. I then prompted the user for the student’s last name and course similarly with the input() function but with the prompts “Please enter the student’s last name: “ and “Please enter the course name: “ respectively. If there were errors with the first name entered, the program printed that there was an error with the first name entered and the error details. There was a similar contingency in the case that there was an error with the last name too. If there were no errors, I then stored this information as a dictionary in the new_student variable which I then appended to the student data variable using the .append() function. Lastly, to confirm that the code worked properly, I added a print() function with an f-string containing the student_first_name, student_last_name, and course_name variables to show that the entered data was registered. The snip below contains the code as described.



```
# Registering a Student
def register_student():
    try:
        student_first_name = input("Please enter the student's first name: ")
        try:
            student_last_name = input("Please enter the student's last name: ")
            course_name = input("Please enter the course name: ")
            new_student = {
                "First_Name": student_first_name,
                "Last_Name": student_last_name,
                "Course": course_name
            }
            student_data.append(new_student)
            print(f"{student_first_name} {student_last_name} in {course_name} has been entered.")
        except Exception as error_details:
            print("There was an error entering the student's last name")
            print(f"Error: {error_details}")
    except Exception as error_details:
        print("There was an error entering the student's first name")
        print(f"Error: {error_details}")
```

For option 2, I ran the “print_data() function. In that function, I had the script print “The current list of students is:” using the print() function. Then I iterated through the student_data list and printed data for each student as one line of text per student. Below contains the code as described.

```
# Printing Current Student Data
def print_data():
    print("The current list of students is:")
    print("Name \t\tLast Name \tCourse")
    for student in student_data:
        print(f"{student["First_Name"]} \t\t {student["Last_Name"]} \t\t{student["Course"]}")
```

For option 3, I ran the “save_data()” function which was needed to save the data of the students into the “Enrollments.csv” file. To do this, I opened the file with “open(FILE_NAME, ‘w’)” and set it equal to file. Then I iterated through each entry on student_data and stored the student’s data in csv_data as an f-string. I then wrote the f-string into the file, added a new line and continued through the student_data. Once that was completed, I closed the file with the .close function. To show that the work was completed, I printed that the lists in student_data were saved in “Enrollments.csv” through FILE_NAME and then printed the actual list of student data saved into the file. In the case that there was

an error with saving the data, I had the program print that there was an error and print the details of the error. Below is the code as described.

```
case "3":
    file = open(FILE_NAME, "w")
    for student in student_data:
        csv_data = f"{student[0]},{student[1]},{student[2]}\n"
        file.write(csv_data)
    file.close()
    print(f"The following list was saved in {FILE_NAME}:")
    for student in student_data:
        print(student)
```

For option 4, the program needs to end. To do this, I added the code “exit()” to have the program end. In order to show that the program ended, I made sure to print “The program has ended” before the program actually ended.

```
case "4":
    print("The program has ended.")
    exit()
```

Lastly, In the case that the user enters an invalid option, I made the program print “Please select only 1, 2, 3, or 4”. The program would detect an invalid option through the “case _:” part of the code. Thanks to that part of the program being in the while loop, the program will loop back to the menu.

```
case _:
    print("Please select only 1, 2, 3, or 4.")
```

By doing all of this, I was able to complete Assignment 05. I then named the script “assignment05.py”.

Summary

Through lessons learned in class, I was able load in data from a csv file into a list to create a dictionary of lists. Then I prompted the user to follow a menu with a variety of options including registering data, displaying the current data, saving the data, and exiting the program. All of this included structured error handling detailing what went wrong if there was an input from Enrollment.csv or from the user that cause issues.

Summary

Link to Github: https://github.com/RyanS39/IntroToProg-Python