Ryan Choi

November 13, 2024

Foundations of Programming: Python

Assignment 05

https://github.com/megaryanc/IntroToProg-Python-Mod05

Advanced Collections and Error Handling

Introduction

Assignment 05 was built on the script previously written for assignment 04. However, it added a new technique of error handling with a more advanced method for data collections. PyCharm IDE was used for the script with basic outline provided as a starter file.

Define Data Constants and Variables

Data constants and variables were all the same from assignment 04 apart from student_data being defined as a dictionary and using json_data. Import json was used for various json functions. The dictionary was set to empty as shown on page 3 of Mod05 notes as shown in figure 1.

```
import json

# 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.
----------------------------------------
'''
# Define the Data Constants
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables and constants
student_first_name: str = ''  # Holds the first name of a student entered by the user.
student_last_name: str = ''  # Holds the last name of a student entered by the user.
course_name: str = ''  # Holds the name of a course entered by the user.
student_data: dict[str,str] = {} # one row of student data
students: list = []  # a table of student data
json_data: str = ''  # a text-based format for storing and exchanging data that is both human-readable and machine-parsable.
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
```

Figure 1. Application of list.

Processing

The first portion of the script starts with loading a json file. How to work with json file was found in page 17 of the notes. The following script accounts for the json file available and read into a two-dimensional list table. The following exceptions were made if the json file did not exist or if any other errors exist.

```
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)
    file.close()
except FileNotFoundError: # if file does not exist.
    print('File not found. Starting with empty list.') # notifies user of file not found.
except Exception as e: #for any other errors.
    print(f'Error code: {e}')
```

Figure 2. Read Json file with error handling.

Menu choice 1 was updated to alert users of an error if the input was left empty. The empty field made more sense rather than setting the script to detect numbers in the names. A ValueError creates a message of “missing” which forces users to input the name. The student data inputs were renamed for an easier recall for the print of alerting the user of the registration. Figure 3 shows the script for menu choice 1 with the error handlings.

```
# Present and Process the data
while (True):

    # Present the menu of choices
    print(MENU)
    menu_choice = input("What would you like to do: ")

    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        try:
            student_first_name = input("Enter the student's first name: ")
            if not student_first_name: # if left empty
                raise ValueError('First name missing.')
            student_last_name = input("Enter the student's last name: ")
            if not student_last_name: # if left empty
                raise ValueError('Last name missing.')
            course_name = input("Please enter the name of the course: ")
            student_data = {
                           'FirstName': student_first_name,     # rename for easier recall.
                           'LastName': student_last_name,
                           'CourseName': course_name
            }
            students.append(student_data)
            print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
        except ValueError as ve:
            print(f'Error code: {ve}')
        except Exception as e:
            print(f'Error code: {e}')
        continue
```

Figure 3. Menu_choice 1 with error handling.

The rest of the menu choices are shown in figure 4. Menu choice 2 still saves the data and displays a message of the entered inputs into a format specified in the print. Menu choice 3 added a json,dump so the output of the python could be recorded into the json. The last menu choice closed the program and displayed the message of “program ended”.

```
    # Present the current data
    elif menu_choice == "2":
        # Process the data to create and display a custom message
        print("-"*50)
        for student in students:
            print(f"Student {student['FirstName']} {student['LastName']} is enrolled in {student['CourseName']}")
        print("-"*50)
        continue
    # Save the data to a file
    elif menu_choice == "3":
        try:
            with open(FILE_NAME, "w") as file:
                json.dump(students, file)
            file.close()
        except ValueError as ve:
            print(f'Error code: {ve}')
        except Exception as e:
            print(f'Error code: {e}')
        file.close()
        print("The following data was saved to file!")
        continue

    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop
    else:
        print("Please only choose option 1, 2, or 3")
print("Program Ended")
```

Figure 3. Rest of menu choices with error handling

Testing

The program was tested in PyCharm and CMD with the results shown in figure 5 and 6. The proper error handlings were examined with if multiple registrations were taking place. The file output was to json with the result show in in figure 7.

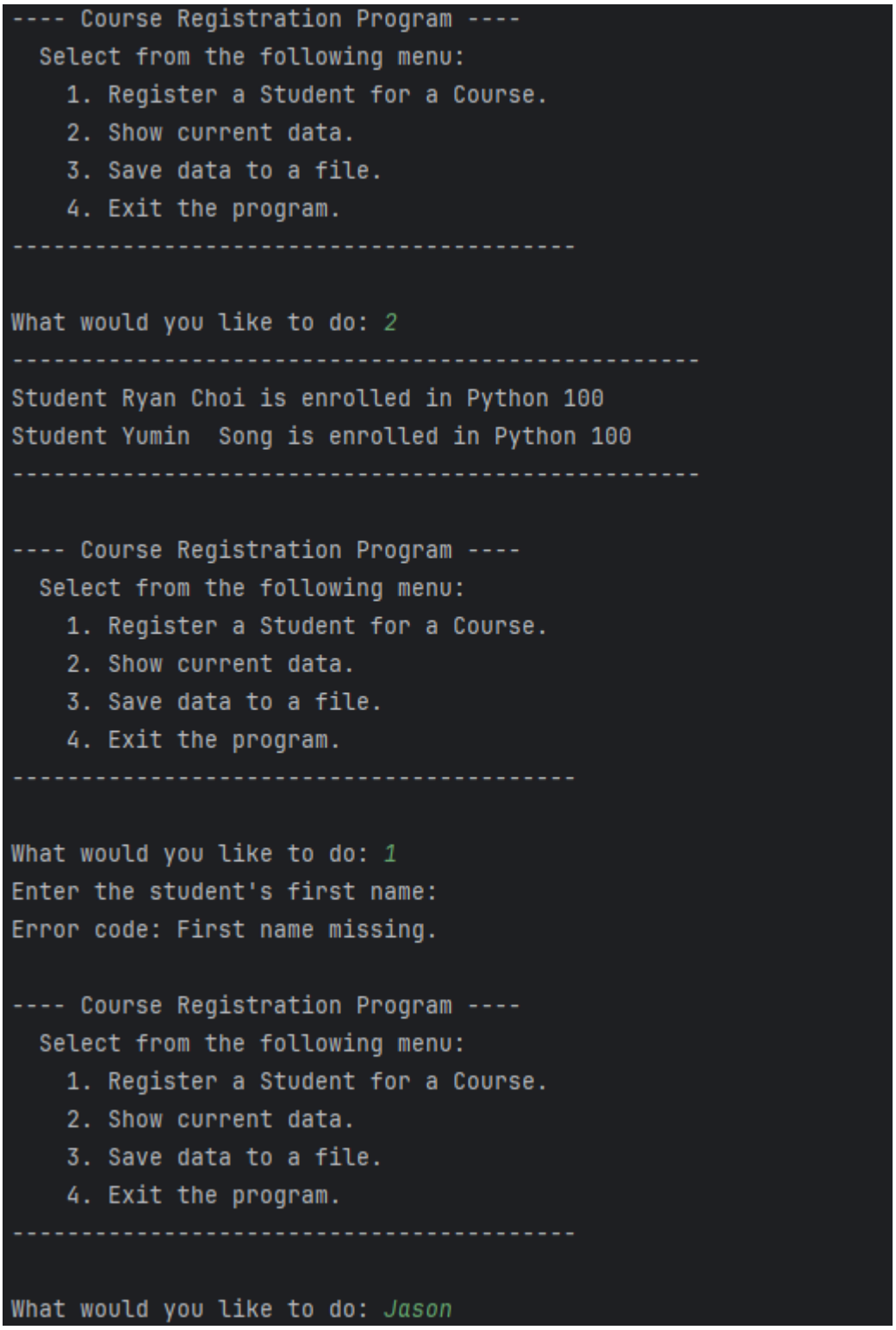

Figure 5. Demonstrates the requirements being met on PyCharm with error handling and multiple registrations.

```
What would you like to do: 1
Enter the student's first name: Ryan
Enter the student's last name:
Error code: Last name missing.

---- 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: Ryan
Enter the student's last name: Choi
Please enter the name of the course: Python 100
You have registered Ryan Choi for 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: 1
Enter the student's first name: Yumin
Enter the student's last name: Song
Please enter the name of the course: Python 100
You have registered Yumin Song for 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: 2
------------------------------------------------
Student Ryan Choi is enrolled in Python 100
Student Yumin Song is enrolled in Python 100
------------------------------------------------
```

Figure 5. Demonstrates the requirements being met on CMD.

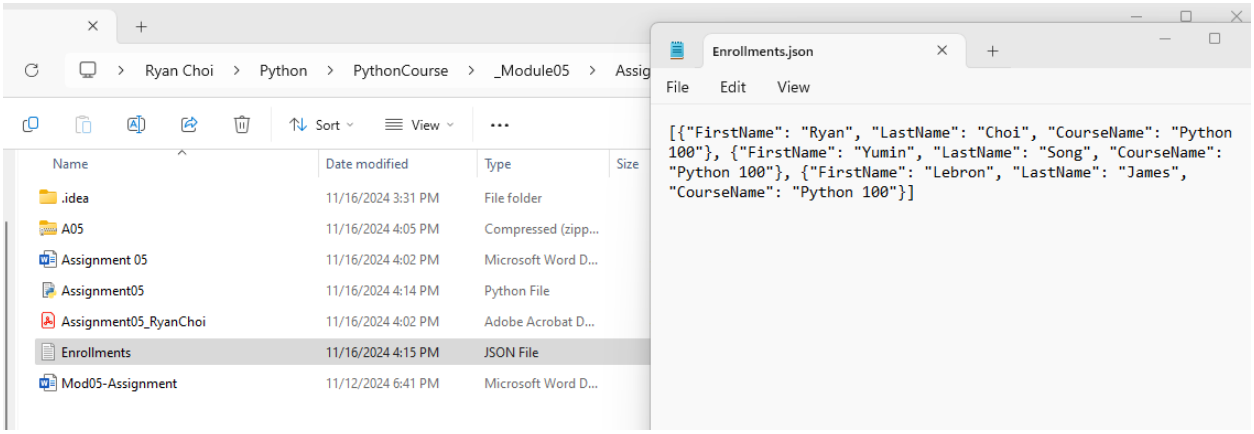

Figure 6. Data recorded in the json.

Summary

The assignment was like 4 but it involved how data could be collected while working with Json. It was new to be able to import functions for json while learning about error handling. The new skills seem useful to detect potential user errors before it is inputted into the database.