

Sabrina Fechtner

10 November 2023

IT FDN 110 B Au 23

Assignment 05

<https://github.com/sfechtner42/IntroToProg-Python>

# Interactive Course Registration Part 3

## Introduction

This week I was tasked modify my program from last week and incorporate dictionaries, JSON files, and structured error handling. Like before, the program should be able to allow the user to submit multiple inputs, but this week all the data is written into a JSON file.

## Constant Variables

The major modification to the constant variables is now the Enrollment file is a .json file rather than a .csv (Figure 1).

```
import json
from typing import TextIO

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.JSON"
```

**Figure 1: Defining the constant variables**

## Data Variables

Again, the data variables remain mostly unchanged from last week's assignment. This week in lecture we learned about dictionaries. Therefore, this week a student's individual data is now stored as a dictionary and the collection of student data is now defined as a list of dictionary entries (Figure 2).

```
# Define the Data Variables
student_first_name: str = ""
student_last_name: str = ""
course_name: str = ""
student_data: dict = None
file: TextIO = None
menu_choice: str
students: list[dict] = []
```

**Figure 2: Data variables**

## Exception Handling 1: None/Preexisting Data

Previously, I pre-loaded data into an .csv file in order to overcome an error. This week, I used exception handling to overcome the error. I assumed the file did exist, so I began with opening the file in read mode and loading the file. Because there is a possibility of no file existing prior, this code is under a 'try' command to hint at a possible error. To catch the other possibility that the file didn't exist previously, I wrote another except block opening and writing initial data (which is blank see figure 2). When running my code, I came across a `JSONDecodeError` with extra lines or characters. Reading through this blog post suggested there was an issue with the array: <https://bobbyhadz.com/blog/python-jsondecodeerror-extra-data>, therefore, I wrote in a file reset to overcome this. If any other error came up that I missed, I wrote a generic catch all exception that I decided to resolve with resetting as well. And to close out the try/except statements, I finished up with closing the files. (Figure 3).

```
# Try reading existing data first
try:
    with open(FILE_NAME, "r") as file:
        students = json.load(file)
        print("Data successfully loaded from the file.")
except FileNotFoundError as e:
    print(f"File not found, creating it...")
    students = []
    with open(FILE_NAME, "w") as file:
        json.dump(obj=students, fp=file)
except json.JSONDecodeError as e:
    print(f"Invalid JSON file: {e}. Resetting it...")
    students = []
except Exception as e:
    print(f"An unexpected error occurred while loading data: {e}")
    students = []
finally:
    if file and not file.closed:
        file.close()
```

**Figure 3: Exception Handling 1**

## Data Input and Exception Handling 2

This remained unchanged from last week (Figure 4). From there on, all the menu options are coded as if/elseif statements to execute different commands within the loop.

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

**Figure 4: User selection from menu**

Option 1 requires first name, last name, and course name which are stored as values to their respective keys as a dictionary. Here, there is more exception handling where the first and last names must not be alphanumeric. Again, using try the user is assumed to enter in a value with no numbers/symbols. However, if they don't the program will stop and re-ask for the information again under a while loop. Once corrected, the program is continues forward and the data is appended into the array (Figure 5).

```
if menu_choice == "1":
    while True:
        try:
            student_first_name = input("Enter the student's first name: ")
            if not student_first_name.isalpha():
                raise ValueError("The first name cannot be alphanumeric. Please re-enter the first name.")
            break
        except ValueError as e:
            print(e)
    while True:
        try:
            student_last_name = input("Enter the student's last name: ")
            if not student_last_name.isalpha():
                raise ValueError("The last name cannot be alphanumeric. Please re-enter the last name.")
            break
        except ValueError as e:
            print(e)
    course_name = input("Please enter the name of the course: ")
    student_data = {"student_first_name": student_first_name, "student_last_name": student_last_name,
                    "course": course_name}
    students.append(student_data)
    continue
```

**Figure 5: Option 1 Data Input and Exception Handling**

## Data Presentation

Option 2 remains mostly unchanged from last week where it allows the user to see what data has been entered thus far. The only major update this week is that the data is not in a f-sting itemized list anymore, it is in dictionary/array (Figure 6).

```
# Present the current data
elif menu_choice == "2":
    print("\nThe current data is:")
    for item in students:
        print(item)
    continue
```

**Figure 6: Option 2 data input check**

## Saving Data to JSON and Exception Handling 3

Option 3 will save/write the user input information onto the .JSON file. Once again, there is a chance that the program errors out here too. Thus, these are written as try/except/finally commands. Under try, I assume that file does exist (upon the success of the code in Figure 3) thus the program will first attempt to write to .JSON under 'w' mode with each input as a new line. The exception is that the file isn't found to which I wrote in another spot to open the file and write input. Like before, I wrote a generic catch-all error message and closed the file using finally.

```
# Save the data to a file
elif menu_choice == "3":
    try:
        with open(FILE_NAME, 'w') as file:
            json.dump(obj: students, fp: file)
            file.write('\n')
        print("Data successfully written to the file.")
    except (FileNotFoundError, IOError) as e:
        with open(FILE_NAME, "w") as file:
            json.dump(obj: students, fp: file)
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
    finally:
        if file and not file.closed:
            file.close()
    continue
```

**Figure 7: Option 3 saving to .JSON and Error Handling 3**

## Closing the Program and the Loop

Finally, option 4 closes out the program. I retained my code from last week without modifications (Figure 8).

```
# 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 8: Closing/Exiting**

## Final Result

I ran my program successfully in PyCharm and in the terminal (Figure 9) where I registered myself for Python 100 and my cat for Python 200. Both the methods of running the code produced a .JSON file showing Cinnamon and I registered for our respective Python courses (Figure 10).

```

File not found, creating it...

---- 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: S@brina
The first name cannot be alphanumeric. Please re-enter the first name.
Enter the student's first name: Sabrina
Enter the student's last name: Fechtner
Please enter the name of the 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.

What would you like to do?: 1
Enter the student's first name: Cinnamon2
The first name cannot be alphanumeric. Please re-enter the first name.
Enter the student's first name: Cinnamon
Enter the student's last name: TheCat
Please enter the name of the course: Python 200

---- 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 current data is:
{'student_first_name': 'Sabrina', 'student_last_name': 'Fechtner', 'course': 'Python 100'}
{'student_first_name': 'Cinnamon', 'student_last_name': 'TheCat', 'course': 'Python 200'}

---- 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
Data successfully written to the file.

---- 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
Program Ended

```

```

File not found, creating it...

---- 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: Sabrina
Enter the student's last name: Fechtner
The last name cannot be alphanumeric. Please re-enter the last name.
Enter the student's last name: Fechtner
Please enter the name of the 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.
-----

What would you like to do?: 1
Enter the student's first name: Cinnamon
Enter the student's last name: TheCat
Please enter the name of the course: Python 200

---- 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 current data is:
{'student_first_name': 'Sabrina', 'student_last_name': 'Fechtner', 'course': 'Python 100'}
{'student_first_name': 'Cinnamon', 'student_last_name': 'TheCat', 'course': 'Python 200'}

---- 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.
-----

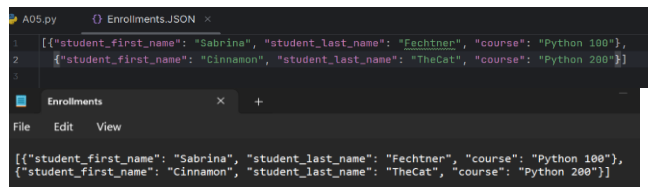
What would you like to do?: 3
Data successfully written to the file.

---- 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

```

**Figure 9: Program output in PyCharm.**



The screenshot shows the PyCharm IDE. At the top, a file named 'Enrollments.JSON' is open, displaying a JSON array of two student enrollment records. Below the file editor, the 'Enrollments' variable is shown in the console, containing the same JSON array.

```

1 [{"student_first_name": "Sabrina", "student_last_name": "Fechtner", "course": "Python 100"},
2   {"student_first_name": "Cinnamon", "student_last_name": "TheCat", "course": "Python 200"}]
3

```

```

Enrollments
[{"student_first_name": "Sabrina", "student_last_name": "Fechtner", "course": "Python 100"},
{"student_first_name": "Cinnamon", "student_last_name": "TheCat", "course": "Python 200"}]

```

**Figure 10: Generated JSON files.**

## Summary

This week, I modified my interactive course registration program from last week where I that took user input, combined it into a dictionary, included error handling, and wrote it onto a .JSON file. This program successfully runs, but there are still some revisions I would like to make. For example, hyphenated names should be accepted input, but my current program will reject them. Therefore, I'm curious how one could accomplish that in the future.