

Name: Julianne Tillmann

Date: November 15, 2024

Course: IT FDN 110 A Au 24: Foundations Of Programming: Python

Assignment 05

## **Creating a Python Script with Dictionaries, JSON Files, and Error Handling**

### **Introduction**

The fifth assignment in this course builds upon previous assignments by introducing dictionaries, JSON files, and structured error handling. The goal was to enhance the existing student registration program to use dictionaries for data storage, JSON files for persistence, and try-except blocks for error handling. This program registers students for courses, stores their data, displays current enrollments, saves data to a JSON file, and exits when prompted.

In creating this script, I worked from the provided starter file and made the necessary changes to meet the requirements.

### **Creating the Script and Adding the Header**

As with all assignments, I began by updating the script header with my name, date, and a brief description of the changes made. This ensures proper documentation and version control.

### **Creating the Main Body of the Script**

#### ***Defining Constants and Variables***

The script includes the following constants:

MENU: A string constant displaying the menu options for the user.

FILE\_NAME: A string constant holding the name of the JSON file used for storing enrollments.

Additionally, several variables are defined for storing student details (student\_first\_name, student\_last\_name, and course\_name), a dictionary (student\_data) for individual student records, and a list (students) for holding all student records.

#### ***Using JSON***

I updated the script to use a JSON file (Enrollments.json) for storing and retrieving data. At the start of the program, the file is opened in read mode, and its contents are loaded into the students list using the json.load() function. If the file does not exist, the program catches the FileNotFoundError exception, notifies the user, and creates a new file when data is saved.

This approach ensures that all data is stored in a structured format and persists between program executions.

#### ***Adding Error Handling***

Next, I added error handling using three types of error handling:

1. File Errors: Handles missing files and ensures the file is closed or saved properly after reading or writing.
2. Input Validation: Checks that first and last names are alphanumeric. If invalid input is detected, a ValueError is raised, and the user is prompted to try again.
3. Key Errors: Handles missing or incorrect data. This also applies to data loaded from a pre-existing file.

These enhancements ensure the program can handle unexpected errors and invalid input.

### ***Main Features of the Program***

1. Menu Option 1: Entering New Data - This option prompts the user for the student's first name, last name, and course name. The dictionary is then appended to the students list. Validation is performed to ensure that the first and last names are alphanumeric and extra spaces at the end of the name are removed.
2. Menu Option 2: Show Current Data - This option iterates over the students list and prints each student's details. The program checks for missing or incorrect data using a try-except block.
3. Menu Option 3: Save Data to a File - This option writes the students list to the JSON file using the json.dump() function. The program confirms that the data has been saved successfully by printing the saved data back to the console.
4. Menu Option 4: Exit the Program - This option ends the program by breaking the while loop. If the user enters an invalid menu option, the program displays an error message and re-prompts the user for input.

### **Testing the Script**

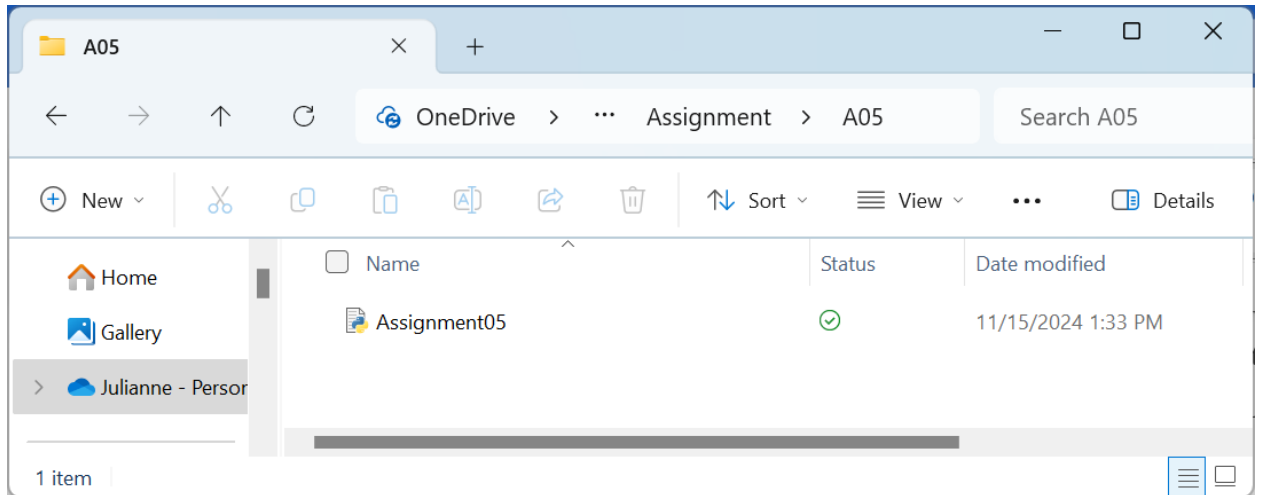
I tested the script by simulating various scenarios:

1. Starting with no pre-existing file and adding new students.
2. Verifying that data is saved correctly to Enrollments.json.
3. Reloading the program with pre-existing data and ensuring that all enrollments are displayed.
4. Handling invalid input for first and last names.
5. Testing the program's behavior with malformed or incomplete data in the JSON file.

Below is a summary of the test cases and results:

#### ***Test Case 1: Starting Without a Pre-Existing File***

- Input: No Enrollments.json file present (Figures 1 and 2).



*Figure 1.* The folder did not contain a pre-existing json file prior to running the script.

```
C:\Users\JTill\OneDrive\Documents\Professional development\UW Python Fall 2024\Module05\Assignment\A05>Assignment05.py
Enrollments.json was not found. A new file will be created.

---- 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: |
```

*Figure 2.* Running the script without a pre-existing json file showed the message that a new script will be created.

- Result: The program created a new file after adding data and saved it correctly (Figures 3 and 4)

```
Command Prompt - Assignm... x + v

What would you like to do: 1
Enter the student's first name: Juliann
Enter the student's last name: Tillmann
Please enter the name of the course: Python 100
You have registered Juliann Tillmann 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: Bob
Enter the student's last name: Ross
Please enter the name of the course: Art 410
You have registered Bob Ross for Art 410.

---- 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: |
```

Figure 3. Adding two rows of new data.

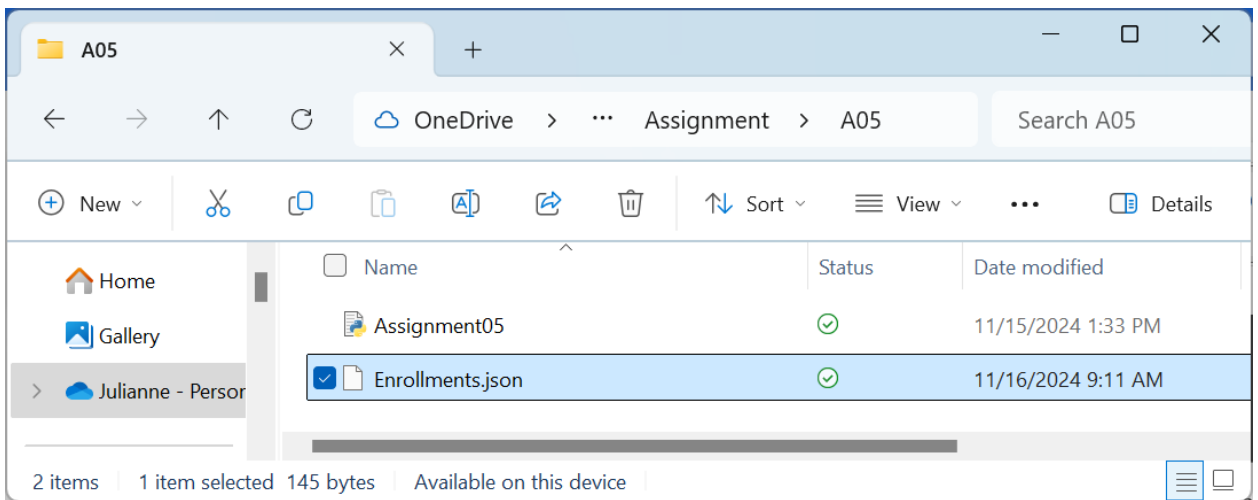
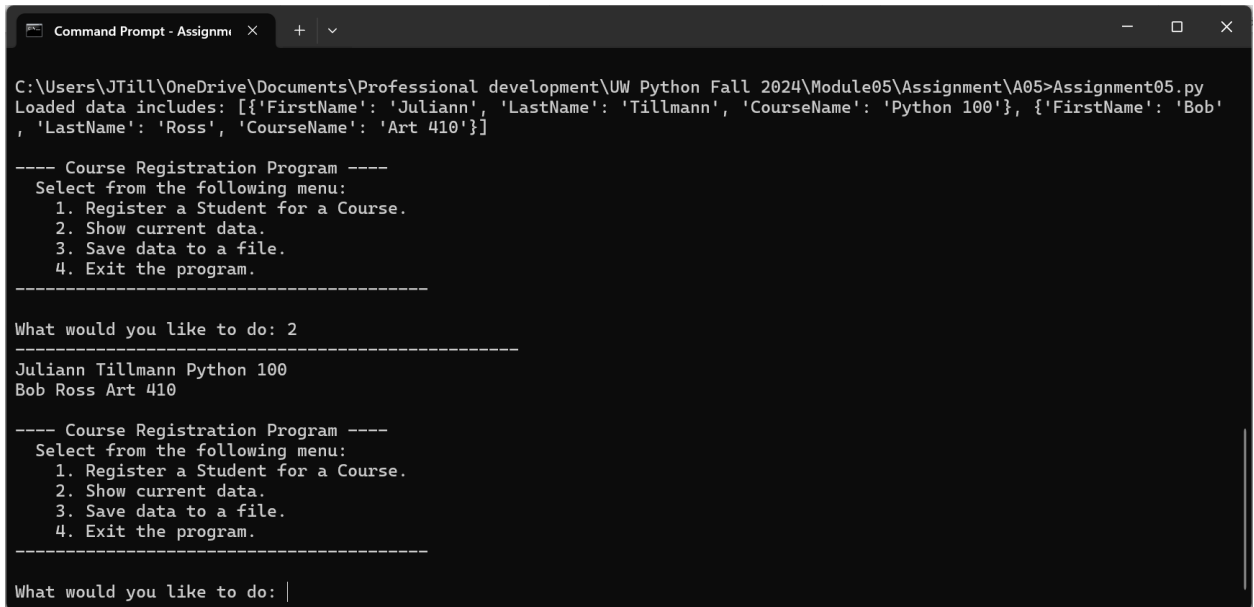


Figure 4. The JSON file has been created.

### Test Case 2: Adding New Students and Checking Data Accuracy

- Input: Entered multiple students using Menu Option 1.
- Input: Selected Menu Option 2.
- Result: All student data was displayed correctly (Figure 5).



```
C:\Users\JTill\OneDrive\Documents\Professional development\UW Python Fall 2024\Module05\Assignment\A05>Assignment05.py
Loaded data includes: [{'FirstName': 'Juliann', 'LastName': 'Tillmann', 'CourseName': 'Python 100'}, {'FirstName': 'Bob',
, 'LastName': 'Ross', 'CourseName': 'Art 410'}]

---- 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
-----
Juliann Tillmann Python 100
Bob Ross Art 410

---- 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: |
```

Figure 5. When selecting Option 2, the two new records showed up correctly.

- Opened the JSON file to check the data in the file (Figure 6).



```
[{"FirstName": "Juliann", "LastName": "Tillmann", "CourseName": "Python 100"},
{"FirstName": "Bob", "LastName": "Ross", "CourseName": "Art 410"}]
```

Figure 6. Contents of the JSON file when opened with Notepad.

#### ***Test Case 4: Reloading with Pre-Existing File***

- Input: Re-ran the program with Enrollments.json already containing data (Figure 7).
- Result: The program loaded the data correctly into the students list (Figure 7).
- Input: Added a couple more new students, saved the file, and exited. The JSON file contained the pre-existing and the new data (Figure 8).

```
Command Prompt - Assignm x + v
C:\Users\JTill>cd C:\Users\JTill\OneDrive\Documents\Professional development\UW Python Fall 2024\Module05\Assignment\A05
C:\Users\JTill\OneDrive\Documents\Professional development\UW Python Fall 2024\Module05\Assignment\A05>Assignment05.py
Loaded data includes: [{'FirstName': 'Juliann', 'LastName': 'Tillmann', 'CourseName': 'Python 100'}, {'FirstName': 'Bob',
, 'LastName': 'Ross', 'CourseName': 'Art 410'}]

---- 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
-----
Juliann Tillmann Python 100
Bob Ross Art 410

---- 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: |
```

Figure 7. When running the script with a pre-existing file, the initial message displays all of the pre-existing data. Using Option 2, shows the data correctly.

```
_Data.txt | _LabData.txt | Enrollments | Enroll x + - □ ×
File Edit View ⚙

[{"FirstName": "Juliann", "LastName": "Tillmann", "CourseName": "Python 100"},
{"FirstName": "Bob", "LastName": "Ross", "CourseName": "Art 410"}, {"FirstName":
"Rob", "LastName": "Mathews", "CourseName": "Music 300"}, {"FirstName":
"William", "LastName": "Bragg", "CourseName": "Physics 420"}]
```

Figure 8. The JSON file containing the old and the new data.

### ***Test Case 5: Handling Invalid Input***

- Input: Entered invalid first and last names.
- Result: The program raised an error, displayed an appropriate message, and prompted for re-entry (see Figure 9).

```
Command Prompt - Assignm... x + v
-----
What would you like to do: 1
Enter the student's first name: Chistopher Roger

Error: Student's first name must be alphanumeric. Spaces are not allowed. Please try again.

---- 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: Chirstopher
Enter the student's last name: Columbus 2

Error: Student's last name must be alphanumeric. Spaces are not allowed. Please try again.

---- 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: |
```

*Figure 9.* Entering first and last names using non alphanumeric characters prompts an error message.

- Input: Re-ran the program using a modified Enrollments.json (provided for the assignment) that contained invalid data (i.e., one of the entries was missing a last name and included an e-mail address instead).
- Result: Was informed of invalid data in the file (Figure 10). The file cannot be saved with invalid data, which also prints an error message (Figure 11).

```
Command Prompt - Assignm x + v
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\JTill>C:\Users\JTill\OneDrive\Documents\Professional development\UW Python Fall 2024\Module05\Assignment\A05
'C:\Users\JTill\OneDrive\Documents\Professional' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\JTill>cd C:\Users\JTill\OneDrive\Documents\Professional development\UW Python Fall 2024\Module05\Assignment\A05

C:\Users\JTill\OneDrive\Documents\Professional development\UW Python Fall 2024\Module05\Assignment\A05>Assignment05.py
Loaded data includes: [{'FirstName': 'Bob', 'LastName': 'Smith', 'CourseName': 'Python 100'}, {'FirstName': 'Sue', 'Email': 'Jones', 'CourseName': 'Python 100'}, {'FirstName': 'Craig', 'LastName': 'Rothburn', 'CourseName': 'Math 100'}]
-----
Loaded data includes: Bob Smith Python 100
Missing or incorrect data in student record: 'LastName'. Record: {'FirstName': 'Sue', 'Email': 'Jones', 'CourseName': 'Python 100'}
Loaded data includes: Craig Rothburn Math 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: |
```

Figure 10. An error message showing incorrect data in the pre-existing file.

```
Command Prompt - Assignm x + v
l': 'Jones', 'CourseName': 'Python 100'}, {'FirstName': 'Craig', 'LastName': 'Rothburn', 'CourseName': 'Math 100'}]
-----
Loaded data includes: Bob Smith Python 100
Missing or incorrect data in student record: 'LastName'. Record: {'FirstName': 'Sue', 'Email': 'Jones', 'CourseName': 'Python 100'}
Loaded data includes: Craig Rothburn Math 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 data was saved to file:
Bob Smith Python 100.
There was an error saving data to Enrollments.json: 'LastName'.

---- 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: |
```

Figure 11. An error message when saving incorrect data.

## Summary

This assignment enhanced my understanding of Python dictionaries, JSON handling, and structured error handling. By integrating these features into the student registration program, I improved its functionality and reliability. The use of JSON for data storage ensures that data remains structured and easy to work with across sessions. Testing the program thoroughly



allowed me to identify and fix issues. Error handling that I added to the script was very helpful in this process.