

Jessica Boardman

July 29, 2024

IT FND 110

Assignment 05

<GitHubURL>

Dictionaries, JSON and Error Handling

Introduction

This week I learned how to store data in Dictionaries, read/write to JSON files, and build error handling with validations. The script I created reads from an existing JSON file, collects inputs and stores the inputs in a Dictionary with keys, which is added to a list, before writing back to the JSON file. Included in the script are error messages when exceptions occur or when specific user inputs are validated. This Python script started from an existing base script, from which I needed to interpret the existing script and make adjustments to meet the acceptance criteria. Upon completion of authoring the program, I was able to execute the script in both PyCharm and the MacOS terminal. The steps I took to complete Assignment 05 are expanded below.

Creating the Program

Before attempting to create the program, I read through the entire acceptance criteria in the assignment to gain a holistic view of what the program needed to do from beginning to end. I encountered some confusion with the Acceptance Criteria in the 'Test' section, but decided to proceed with the JSON format as that is what was taught this week. I authored the Python

program Assignment05.py in PyCharm. First I set up the appropriate header (Figure 1.1). In the header, my references are on a new line as I was modifying an existing script.

```
Assignment05.py x
1  # -----
2  # Title: Assignment05
3  # Desc: This assignment demonstrates using dictionaries, files, and exception handling
4  # Change Log: (Who, When, What)
5  #   RRoot,1/1/2030,Created Script
6  #   JBoardman, 7/30/2024, updated script with error handling
7  # ----- #
```

Figure 1.1: Header Setup

Based on the requirements, I created constants and variables to be used in the script (Figure 1.2). The student_data in this script is set to be an empty dictionary, as indicated by the curly brackets {}. In addition to the constant and variable set up, I also imported code from Python's JSON module using the import function (Figure 1.2). This was placed above the constants and variables because of the PyCharm info/warning messages suggesting I use the import function at the beginning of the script. One last thing I added in this section was the Show boolean. This variable is used to only show the menu choice if loading the JSON is successful.

```
8  # (jb 7.30.24) import code from python's json module into my script
9  import json
10
11  # Define the Data Constants
12  MENU: str = '''
13  ---- Course Registration Program ----
14  Select from the following menu:
15  1. Register a Student for a Course.
16  2. Show current data.
17  3. Save data to a file.
18  4. Exit the program.
19  -----
20  '''
21  # Define the Data Constants
22  FILE_NAME: str = "Enrollments.json"
23
24  # Define the Data Variables and constants
25  student_first_name: str = '' # Holds the first name of a student entered by the user.
26  student_last_name: str = '' # Holds the last name of a student entered by the user.
27  course_name: str = '' # Holds the name of a course entered by the user.
28  student_data: dict = {} # one row of student data (jb 7.30.24) made dict
29  students: list = [] # a table of student data
30  json_data: str = '' # (jb 7.30.24) string for json data
31  # csv_data: str = '' # Holds combined string data separated by a comma. (jb 7.30.24) removed
32  file = None # Holds a reference to an opened file.
33  menu_choice: str # Hold the choice made by the user.
34  Show: bool = True # (jb 7.30.24) added to hide the menu if the file doesn't load correctly
35
```

Figure 1.2: Constant and Variable Setup, Import JSON

Read and Load a JSON File

This week's assignment introduced the use .json files. JSON files are useful for complex and nested data structures and use key-value pairs to organize the data in a row. Python has a module specifically for JSON which was imported into the script. The modules .load() method was called when reading the file (Figure 1.3). Instead of a for loop to go through each row (dictionary) and parse the key-values into variables, the json.load(file) extracted all data into the students list.

```
36 # When the program starts, read the file data into a list of dictionaries
37 # Extract the data from the file
38 try: # (jb 7.30.2024) added try for exception error handling
39     file = open(FILE_NAME, "r")
40     # (jb 7.30.2024) loading the students from the json
41     students = json.load(file) # (jb 7.30.2024) fixed starter file json with Email as key for las
42     file.close()
43 # (jb 7.30.24) removed the parsing format from csv & multi row comment errors inside try
44 except Exception as e: # (jb 7.30.2024) adding exceptions
45     print("-" * 80)
46     print("Error loading file: Please verify the file exists and includes valid JSON data.")
47     print("-" * 80)
48     print("-- Technical Error Message -- ")
49     print(e, e.__doc__, type(e), sep='\n')
50     Show = False # (jb 7.30.24) set to false hide the menu prompt for the user
51 finally:
52     if file.closed == False:
53         file.close()
54
```

Figure 1.3: Read and Load JSON file

Dictionaries

Dictionaries are one of the prebuilt data types in Python and are useful when storing multiple items in a single variable (row). Different from Lists, Dictionaries use specific keys to map to values within the row, making it intuitive and easy to understand. A significant advantage to using Dictionaries is the ability to transform data by accessing the keys where lists may require

additional processing. For this assignment, the program collects input data from menu choice 1 that is stored in variables for the students first name, last name and course name in a Dictionary. The keys are enclosed in double quotes, followed by a colon. The dictionary student_data is then appended to the students list. (Figure 1.4).

```
74 # (jb 7.30.24) Keys are defined as "FirstName", "LastName", "CourseName"
75 student_data = {"FirstName": student_first_name, "LastName": student_last_name, "CourseName": student_course_name}
76 students.append(student_data)
```

Figure 1.4: Dictionary and Keys

Error Handling

In this Python script, error handling was introduced to catch general exceptions and to validate specific user inputs. Before loading the JSON file, the script has a try block. It was important to indent everything that I wanted to be caught by the error handling. The except block is the error handling if there is an issue reading the file or loading the data to the students list. The script is printing the error message, any documentation and the type of error (Figure 1.5). On line 50 is the Show variable, which I am setting to false so the menu doesn't appear if there is an error in initializing the JSON file.

```
36 # When the program starts, read the file data into a list of dictionaries
37 # Extract the data from the file
38 try: # (jb 7.30.2024) added try for exception error handling
39     file = open(FILE_NAME, "r")
40     # (jb 7.30.2024) loading the students from the json
41     students = json.load(file) # (jb 7.30.2024) fixed starter file json with Email as key for last
42     file.close()
43 # (jb 7.30.24) removed the parsing format from csv & multi row comment errors inside try
44 except Exception as e: # (jb 7.30.2024) adding exceptions
45     print("-" * 80)
46     print("Error loading file: Please verify the file exists and includes valid JSON data.")
47     print("-" * 80)
48     print("-- Technical Error Message -- ")
49     print(e, e.__doc__, type(e), sep='\n')
50     Show = False # (jb 7.30.24) set to false hide the menu prompt for the user
51 finally:
52     if file.closed == False:
53         file.close()
```

Figure 1.5 Error Handling

In addition to general error handling, there are requirements to validate specific user inputs. The student first and last name inputs should only allow alpha characters to be entered. If non-alpha characters are entered, a custom error message should be displayed. `ValueError` is being set with the custom message to inform the user how to correct their inputs. The `ValueError` is then displayed using the `print` function in the `except` block (Figure 1.6).

```
64         # Input user data
65         student_first_name = input("Enter the student's first name: ")
66         # (jb 7.30.24) specific error handling for non alpha
67         if not student_first_name.isalpha():
68             raise ValueError("The first name only accepts letters.")
69         student_last_name = input("Enter the student's last name: ")
70         # (jb 7.30.24) specific error handling for non alpha
71         if not student_last_name.isalpha():
72             raise ValueError("The last name only accepts letters.")
73         course_name = input("Please enter the name of the course: ")
74         # (jb 7.30.24) Keys are defined as "FirstName", "LastName", "CourseName"
75         student_data = {"FirstName": student_first_name, "LastName": student_last_name, "CourseName": course_name}
76         students.append(student_data)
77         print() # (jb 7.30.24) blank line
78         print(f"You have registered {student_first_name} {student_last_name} for {course_name}")
79     # (jb 7.30.24) specific error handling
80     except ValueError as e:
81         # (jb 7.30.24) for value error the message for each field would be displayed
82         print() # blank line
83         print(e) # specific error message
84         print("-- Technical Error Message -- ")
85         print(type(e))
86         print(e.__doc__)
87         # (jb 7.30.24) not including e.__str__() as this double prints the exception text
```

Figure 1.6 Validations

Outputs to JSON File

Similar to Assignment04 this Python script also writes an output but instead of outputs to `.csv`, the output is to the `.json` file. Writing to the JSON file was simple compared to writing to the `csv`. The `json.dump()` on line 115 added the students list to the file (Figure 1.7). Error handling was added to this section to ensure writing to the JSON file was successful and also checking to ensure the file is closed.

```

# Save the data to a file
elif menu_choice == "3":
    try: # (jb 7.30.2024) added try for exception error handling
        file = open(FILE_NAME, "w")
        # (jb 7.30.24) removed write for csv
        # (jb 7.30.24) added write for json with dump function from json module
        json.dump(students, file)
        file.close()
        print("-" * 50) # (jb 7.30.24) added to be consistent with other print functions
        print("The following data was saved to file!")
        for student in students:
            # (jb 7.30.24) adjusted to print from dict using keys
            print(f'Student {student["FirstName"]} '
                  f'{student["LastName"]} is enrolled in {student["CourseName"]}')
        print("-" * 50) # (jb 7.30.24) added to be consistent with other print functions
    except Exception as e: # (jb 7.30.2024) added exception error handling
        print("Error: An error occurred while writing to the file.")
        print("-- Technical Error Message -- ")
        print(e.__doc__)
        print(e.__str__())
    finally:
        if file.closed == False:
            file.close()
        continue

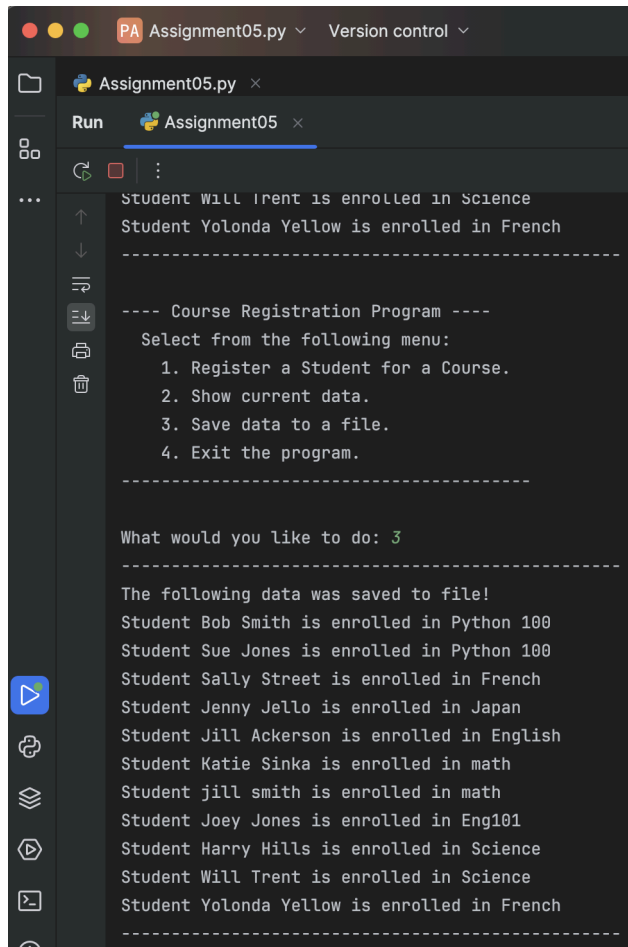
```

Figure 1.7: Output to .json and Error Handling

Testing

The final step in completing Assignment05 was to test the script in both PyCharm and the terminal. To run the program in PyCharm I selected the 'Run' button and was prompted to select from the MENU. Going through each input value, I was able to test the entire script and

with the final option '4' I exited the program (Figure 1.8).



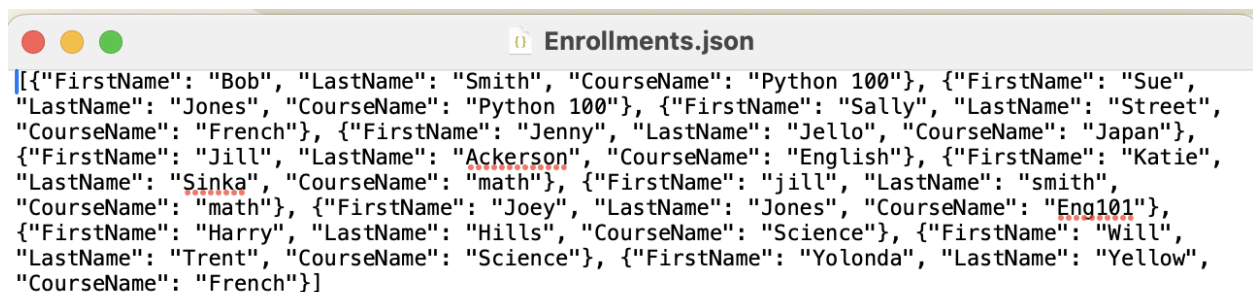
```
Assignment05.py x
Run Assignment05 x
Student Will Trent is enrolled in Science
Student Yolonda Yellow is enrolled in French
-----
---- Course Registration Program ----
Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----

What would you like to do: 3
-----

The following data was saved to file!
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Sally Street is enrolled in French
Student Jenny Jello is enrolled in Japan
Student Jill Ackerson is enrolled in English
Student Katie Sinka is enrolled in math
Student jill smith is enrolled in math
Student Joey Jones is enrolled in Eng101
Student Harry Hills is enrolled in Science
Student Will Trent is enrolled in Science
Student Yolonda Yellow is enrolled in French
-----
```

Figure 1.8: PyCharm Testing

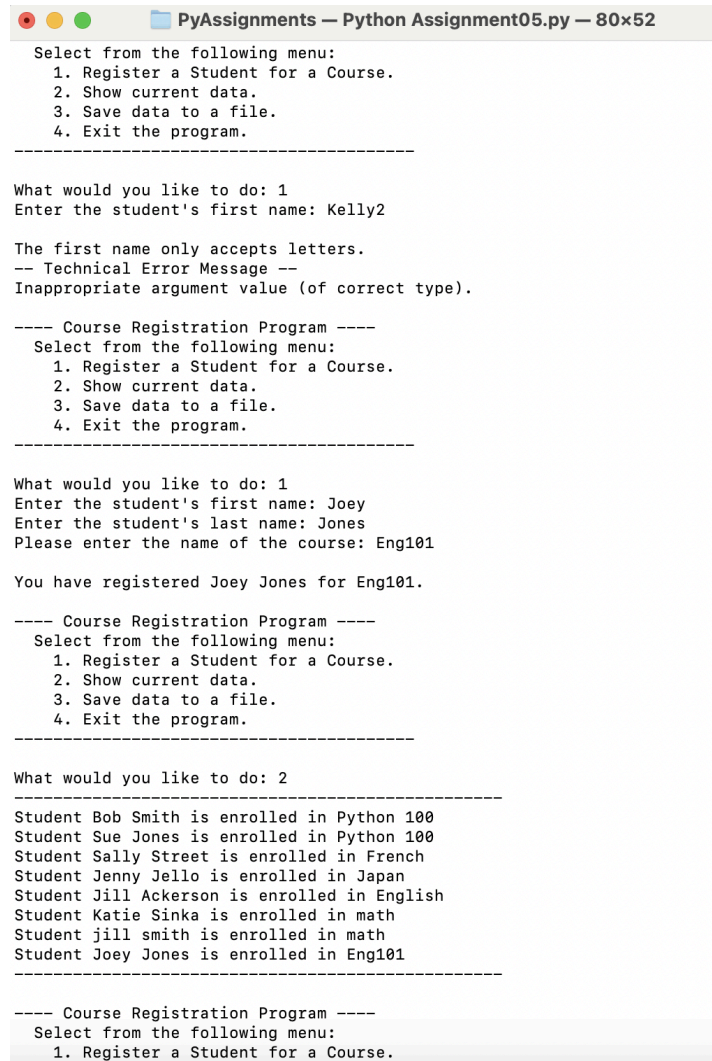
To ensure the JSON file was updated successfully, I opened the Enrollments.json file and verified the contents (Figure 1.9).



```
Enrollments.json
[{"FirstName": "Bob", "LastName": "Smith", "CourseName": "Python 100"}, {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"}, {"FirstName": "Sally", "LastName": "Street", "CourseName": "French"}, {"FirstName": "Jenny", "LastName": "Jello", "CourseName": "Japan"}, {"FirstName": "Jill", "LastName": "Ackerson", "CourseName": "English"}, {"FirstName": "Katie", "LastName": "Sinka", "CourseName": "math"}, {"FirstName": "jill", "LastName": "smith", "CourseName": "math"}, {"FirstName": "Joey", "LastName": "Jones", "CourseName": "Eng101"}, {"FirstName": "Harry", "LastName": "Hills", "CourseName": "Science"}, {"FirstName": "Will", "LastName": "Trent", "CourseName": "Science"}, {"FirstName": "Yolonda", "LastName": "Yellow", "CourseName": "French"}]
```

Figure 1.9: Verify the .json

The final testing was completed in the terminal to simulate End User Testing. After opening the terminal, I used the cd command to navigate to the correct directory. Once I was in the correct directory, I executed the program by entering python3 Assignment05.py in the terminal window. This ran the program and I was able to read the existing file, provide multiple student inputs, display the data and write to the .csv (Figure 1.10).



```
PyAssignments — Python Assignment05.py — 80x52
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: Kelly2

The first name only accepts letters.
-- Technical Error Message --
Inappropriate argument value (of correct type).

---- 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: Joey
Enter the student's last name: Jones
Please enter the name of the course: Eng101

You have registered Joey Jones for Eng101.

---- 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 Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Sally Street is enrolled in French
Student Jenny Jello is enrolled in Japan
Student Jill Ackerson is enrolled in English
Student Katie Sinka is enrolled in math
Student jill smith is enrolled in math
Student Joey Jones is enrolled in Eng101
-----

---- Course Registration Program ----
Select from the following menu:
  1. Register a Student for a Course.
```

Figure 1.10: Terminal Output

Summary

This week I completed Assignment05 and was able to create a Python script in PyCharm that

reads an existing JSON file, collects inputs, creates dictionaries and lists, handles errors, and writes data to a JSON file after displaying the data to users on the screen. The script was tested in both PyCharm and the terminal to verify it worked as expected and met all documented acceptance criteria. The final script was uploaded to Github for peer review by my classmates.