

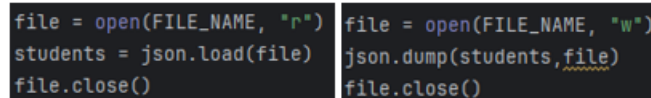
Assignment 5 – json, Dictionaries, and Error Handling

Introduction

Building off of the script from assignment 4 this week focused on incorporating dictionaries, working with json files, and error handling messages. Most of the list objects from assignment 4 were converted to dictionaries for this assignment. The csv formatted starter file was switched to a json formatted file. Loading of the file was given error handling logic, as was menu options 1 and 4. The goals for this weeks script in regards to processing data and the inputs/outputs are the same as week four's with the addition of the error handling outputs.

Json Files and Dictionaries

The starter file for this assignment was a json file (JavaScript Object Notation). Json files store lists of dictionaries. This means that instead of having contextless data separated by commas, such as in csv files, the data is paired with a descriptive key. These key-value pairs make the data more readable for people and it allows the user to parse data via the keys and not the index values like with lists. To more easily work with the json file the json module is imported prior to any handling of the json file. Having the json module allows the programmer to easily load data from and write data to the json file, figure 1.



```
-----  
file = open(FILE_NAME, "r")  
students = json.load(file)  
file.close()  
  
file = open(FILE_NAME, "w")  
json.dump(students, file)  
file.close()
```

Figure 1. The json.load and json.dump functions automate the formatting of data from and into the json file. This saves the programmer time and helps to condense the script.

The script first reads the json file using the json.load() function and saves its content to a list of dictionaries object "students". Students is later amended as the user inputs additional student information. Working with the dictionary class is similar to lists with the noticeable exceptions being the values are bracketed by curly braces {}, key names are case-sensitive, and key names are in double-quotes and must accompany the value. Once the user is finished entering new data option 3 is selected and the data stored in students is saved to the json file using the json.dump() function. For this step the json file is opened in write mode and all of it's content is replaced with the new data. Existing data is not lost as this was read and saved to students at the beginning of the script.

Structured Error Handling

This script employs the Try-Except statements to display information related to an encountered error to the user. Throughout the script the exceptions will print an informal statement describing the error and a technical error message report, figure 2.

```
if not student_last_name.isalpha():
    raise ValueError("Error: Student name should only contain letters.")

except ValueError as e:
    print(e)
    print("-- Technical Error Message --")
    print(e.__doc__, type(e), sep="\n")
    print("")

except Exception as e:
    print("Error: There was a non-specific error.")
    print("-- Technical Error Message --")
    print(e, e.__doc__, type(e), sep="\n")
    print("")
```

Figure 2. The "e" prints the exception object and error message. The "e.__doc__" prints a documentation string of the exception type. The "type" function prints the type of exception object.

For the file load portion of script, the structured error content covered the FileNotFoundError and the general Exception error. For both user input prompts in menu option 1 for the students first and last name the script is structured to only accept letter characters for the names. If a non-letter character is entered a ValueError exception will occur and the message will instruct the user to only use letter characters. There is also a generic Exception object to catch everything else. For both names the try-except logic is nested within while loops which run indefinitely until the user inputs a valid name. The final set of structured error logic supports menu option 3. These section of script includes both a PermissionError and general Exception error. The PermissionError will be triggered if the program attempts to write to the file and it is in a read only configuration.

Summary

Now that this script uses dictionaries in place of lists it will be more accessible to any programmer who wishes to use the script in the future. The json file format is necessary to support this transition to dictionaries. With the addition of error handling structure the script is now more complete and will provide a better experience for future users.