Josh Huang

12/13/2023

IT FDN 110 A

Assignment 5

Assignment 5 – Advanced Collections and Error Handling

Introduction

This assignment demonstrates the usage of using dictionaries, reading .json files, and error exception handling. To accomplish this, the 'json' module was imported and modifications to the reading and writing of the file was done to accommodate the change in file type. Exceptions were also added to handle potential errors the program might experience during execution.

Dictionary

A key difference between assignment 4 and assignment 5 is the difference in data structures. Previously, a 'list of lists', or a table, was used to store information. A dictionary, however, is composed of key-value pairs. The keys can be thought of as like the column name in a spreadsheet.

JSON

JSON, or JavaScript Object Notation, is a commonly used format that is used for configuration files and data storage. Just like Python dictionaries, JSON files consist of key-value pairs. By importing the 'json' module, the methods for importing and exporting data between the file and the program is simplified.

Data Variables

The first section of this script defines all of the required constants and variables needed for this assignment. The FILE_NAME has been changed from a .csv to a .json, and the table that will hold all of the student information has been changed to a dictionary, in anticipation of working with the .json file.

Data Processing

To start, the file "Enrollments.json" needs to be opened. To read data from the existing .json file, the method *json.load* (from the imported *json* module) is used to read into a two-dimensional list. This will populate the *students* table with the *student_data* dictionaries from the file.

```
try:
    file = open(FILE_NAME, "r")
    students=json.load(file)
    file.close()
```

```
students = {list: 4} [{'course_name': 'P23', 'first_name': 'Yu', 'last_name': 'Lee'}, {'course_name': 'P23', 'first_name': 'Yue', 'last_name': 'Lie'},
  0 = {dict: 3} {'course_name': 'P23', 'first_name': 'Yu', 'last_name': 'Lee'}
  1 = {dict: 3} {'course_name': 'P23', 'first_name': 'Yue', 'last_name': 'Lie'}
  2 = {dict: 3} {'course_name': 'P324', 'first_name': 'Boe', 'last_name': 'Joe'}
  3 = {dict: 3} {'course_name': 'PY142', 'first_name': 'Joe', 'last_name': 'Ma'}
  __len__ = {int} 4
```

Figure 1: Loading Existing data from the file

In the case that the file does not exist, an exception is used to warn the user and to create the required file. To test this exception, the Enrollments.json file was deleted and the program was run, resulting in the error message correctly displaying (Figure 2.1), in addition to the file being created to prevent this error from occurring again in future program runs. A catch-all exception is also used to catch any unforeseen errors. In the event that the file remains open after an unknown exception, the file is then closed. This completes the exception handling block. To test this exception, an exception was raised during the file opening process, resulting in the error message correctly printing the Exception as 'e' (Figure 2.2).

```
#Present an error if 'Enrollments.json' does not exist
except FileNotFoundError:
    print('File not found. Creating file.')
    open(FILE_NAME, 'w')
except Exception as e:
    print('Unknown exception. Resetting.')
    students = []
    print(type(e), e, sep='\n')
finally:
    file = open(FILE_NAME, "r")
    if file and not file.closed:
        file.close()
```

Figure 2: File Read- Error Handling

```
Run  Assignment05
C:\Users\ua703f\AppData\Local\Microsoft\
File not found. Creating file.
```

Figure 2.1: File not found Error

```
C:\Users\ua703f\AppData\Local\Microsoft\
Unknown exception. Resetting.
<class 'Exception'>
```

Figure 2.2: Catch-all

Data Processing

The first menu option registers the students for a course. A new requirement from the assignment is for structured error handling for both the user input of a first and last name. In this case, both name inputs must be alphabetic, and so ValueError exception handling was implemented for this case (Figure 3.1).

```
student_first_name = input("Enter the student's first name: ")
if not student_first_name.isalpha():
    raise ValueError ('Error: First name must be alphabetic')
student_last_name = input("Enter the student's last name: ")
if not student_last_name.isalpha():
    raise ValueError('Error: Last name must be alphabetic')
```

Figure 3: User Input – Error Handling

```
Run  Assignment05
---- 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: 21
Error: First name must be alphabetic
```

Figure 3.1: Name must be Alphabetic Error

The user must also enter at least one value for the course name. In the case that the course name is blank, a ValueError exception was implemented for this case, which would inform the user that they did not enter any information.

```
course_name = input("Please enter the name of the course: ")
if not course_name:
    raise ValueError('Course name cannot be blank')
```

Figure 4: Blank Input – Error Handling

Next, the user input data is loaded into a dictionary. This dictionary is then appended to the entire student list, '*students*'. After this, the exception error set as 'e' is a catch-all exception for any unexpected errors.

The third menu option saves the data into the file. The json module provides a method, '*dump*', which converts the Python object into a quoted-string which contains key-value mapping.

```
# Save the data to a file
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w")
        json.dump(students,file)
        file.close()
```

Figure 5: Saving the data to the file

Similar to the exceptions before, this section is completed with an Exception error. This serves as a catch-all exception for any errors related to the data saving, and will print the Exception. In the case that the file is not closed after the Exception, the '*finally*' clause is used to close it.

```
    except Exception as e:
        print('Error saving data to file.')
        print(e)
    finally:
        if file and not file.closed:
            file.close()
```

Figure 6: Data save Exception error

Source Control

The script file and the knowledge document are hosted on the following GitHub repository.

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

This link has been added to the GitHub links forum.

Summary

The script created for Assignment 5 runs from both the console and PyCharm. This script is able to read and process any data from the existing file, receive additional user input and append it to the list of lists, display the data, and save the data to the file. The inclusion of dictionaries, .json files, exception handling, and importing modules allowed for a more robust program that helps anticipate unexpected and potential errors, to result in a better user experience.