Name: Alex Kover
Date: 05/15/24
Course: Introduction to Programming with Python

Assignment 05: Advanced Collections and Error Handling

GitHub link: https://github.com/AKover-UW/IntroToProg-Python-Mod05

Introduction

This document describes the topics and steps taken to complete this assignment. This assignment is very similar to Assignment 04, with the major changes being the use of JSON files instead of csv files, importing modules, and handling errors.

JSON Files

In this module, the JSON file type was introduced. JSON is a data format that is easy for people to read and write to as well as easy for machines to browse and create. It consists of key: value pairs just like dictionary files in python, but they are not restricted to being strings only. This gives JSON files a lot of versatility and usefulness and is simple for people to understand. JSON files and CSV files each have their advantages and disadvantages depending on how they are used. JSON files can be complex and are best suited for data interchange between systems while CVS files are text only but because of their simplicity are ideal for exporting and importing large databases.

```
file = open(FILE_NAME, "r")  # reads file
students = json.load(file)   # loads json file data into dict file
file.close()
```

Figure 1: Reading data from a JSON file into the script through the json.load() function

Importing modules

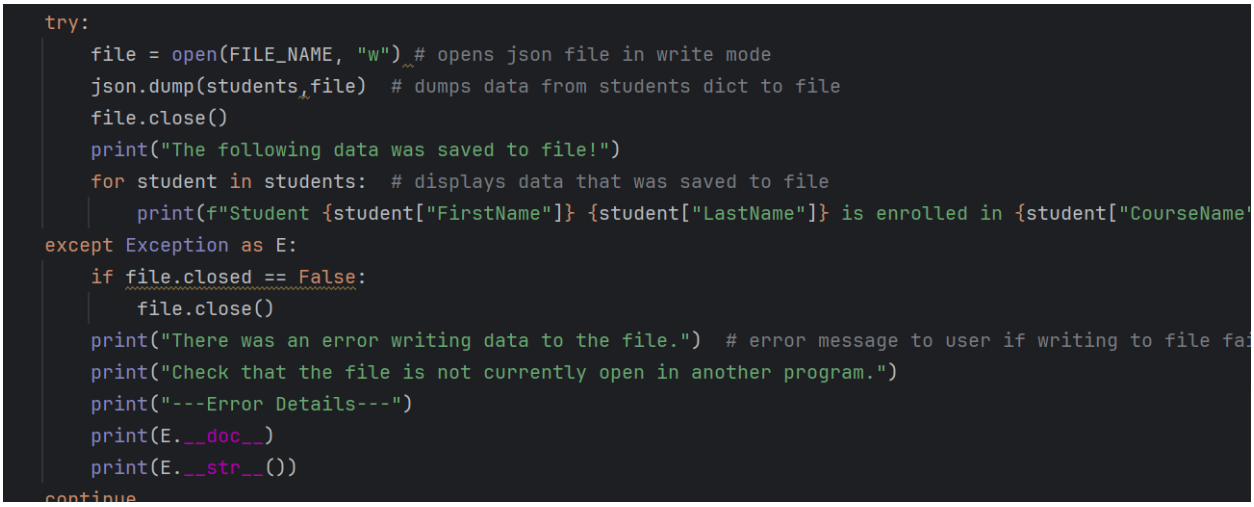
Another topic covering in this assignment is using the import function to bring in functionality that is not already part of the current script without having to create it in the script. Importing allows a script to use previously written code and functions to make writing the script more efficient and simpler. In this case, the import function is used to import the JSON module, which allows the script to use a large variety of JSON specific functions. Many of these functions are very similar to functions used in previous assignments but are designed to specifically operate with JSON files and file structures.

```
file = open(FILE_NAME, "w") # opens json file in write mode
json.dump(students,file)  # dumps data from students dict to file
file.close()
```

Figure 2: JSON specific function json.dump() works similarly to the write() function, but is designed to efficiently write to JSON files

Error Handling

Another topic covered in this module is structured error handling. This provides a way to manage errors that may be out of the programmer's control such as user errors. A user may enter information incorrectly and possibly cause the script to not function. Structuring error handling in the Try-Except format can allow the programmer to catch user errors when they happen and provide more user-friendly feedback than the technical error reports that various programs may display. This style is also a simple and organized way to handle error reports. The Try-Except error handling style can also direct python to display an error message in a way that is more useful to you and the user. Typically, the Try-Except format consists of a section called Try, where the code that is supposed to normally run is held, and Except sections where errors encountered in the Try section can be addressed. Custom errors can also be created using the raise exception where input that may not cause python to generate an error but would harm the functionality of the script can be addressed. An example would be a user entering a number in a name field. This wouldn't break python but would result in incorrect data being used.



```
try:
    file = open(FILE_NAME, "w") # opens json file in write mode
    json.dump(students,file)  # dumps data from students dict to file
    file.close()
    print("The following data was saved to file!")
    for student in students:  # displays data that was saved to file
        print(f"Student {student["FirstName"]} {student["LastName"]} is enrolled in {student["CourseName"
except Exception as E:
    if file.closed == False:
        file.close()
    print("There was an error writing data to the file.")  # error message to user if writing to file fai
    print("Check that the file is not currently open in another program.")
    print("---Error Details---")
    print(E.__doc__)
    print(E.__str__())
continue
```

Figure 3: Error handling with Try-Except format. A more detailed, user-friendly explanation for the error is provided to the user when an error occurs.

Summary

While this assignment is very similar to the previous assignment, the introduction of JSON files, importing modules, and Try-Except error handling enhances and broadens the capabilities and usefulness of the script.