Deric Fetters

5/27/2024

IT FDN 110 A

Assignment 05

Advanced Collections and Error Handling

Introduction

Module 5 most notably introduces working with dictionaries, JSON files, error handling and the GitHub website. All of these topics improve the functionality, usability and shareability of our code.

Dictionaries and working with JSON files

Importing json Module

In order to use the load() and dump () functions, the json module must be imported by the program. From experience, this step is easy to forget. See Figure 1.

```
7    import io
8    import json
```

Figure 1: Assignment05.py import of io and json modules

Declaring and Initializing Variables

Use of dictionaries and the json module required some changes to the variables compared to the starter file. “Student_data” was initialized with {} instead of [] to indicate it is a dictionary. “Json_data” was initialized as a string, but never used in the code. See Figure 1.

```
20   # Define the Data Constants
21   FILE_NAME: str = "Enrollments.json"
22   FILE_NAME_CSV: str = "Enrollments_csv.csv"
23
24   # Define file variables
25   file = io.TextIOWrapper
26   file_csv = None
27
28   # Define the Data Variables and constants
29   student_first_name: str = ''  # Holds the first name of a student entered by the user.
30   student_last_name: str = ''  # Holds the last name of a student entered by the user.
31   course_name: str = ''  # Holds the name of a course entered by the user.
32   json_data: str = ''
33   student_data: dict = {}  # one row of student data in a dictionary
34   students: list = []  # a table of student data
35   csv_data: str = ''  # Holds combined string data separated by a comma.
36    # Holds a reference to an opened file.
37   menu_choice: str  # Hold the choice made by the user.
```

Figure 1: Assignment05.py declaration and initialization of variables

Reading data from and writing data to .json file

The load() and dump() functions were used to read from and write to the file “Enrollments.json”. This simplified the syntax for both actions to one line of code each instead of the For loops used for reading and writing to .csv files. See figures 2 and 3.

```
42      file = open(FILE_NAME, "r")
43      students = json.load(file)
44      file.close()
```

Figure 2: Assignment05.py Reading data from json file

```
102             # Writes data to .json file
103             file = open(FILE_NAME, "w")
104             json.dump(students, file)
105             file.close()
```

Figure 3: Assignment05.py Writing data to a json file

Using dictionaries in conjunction with Lists

Using dictionaries with lists instead of lists with lists provided some advantages and disadvantages. One of the biggest advantages was the improved readability of the code. Named keys are used instead of referencing a position in a list. This advantage becomes more apparent as more elements are added to a list. When printing, the code structure for dictionaries was very similar as well. For loops were used to print each element of the list containing the dictionary. See Figure 4. One disadvantage is that all keys must match throughout the code and the files that are uploaded. I see it as a minor trade off as PyCharm will find these issues for you. The “Enrollments.json” file had an incorrect key that caused my code to return errors. Since this error wasn’t mention in the assignment criteria, I corrected the issue in “Enrollments.json”.

```
63      # Input user data
64      if menu_choice == "1":  # This will not work if it is an integer!
65          try:
66              student_first_name = input("Enter the student's first name: ")
67              if not student_first_name.isalpha():
68                  raise ValueError("The first name should not contain numbers.")
69              student_last_name = input("Enter the student's last name: ")
70              if not student_last_name.isalpha():
71                  raise ValueError("The last name should not contain numbers.")
72              course_name = input("Please enter the name of the course: ")
73              student_data = {"FirstName": student_first_name, "LastName": student_last_name, "CourseName": course_name}
74              students.append(student_data)
75              print(f"You have registered {student_data["FirstName"]} {student_data["LastName"]} "
76                f"for {student_data["CourseName"]}.")
```

Figure 4: Assignment05.py Using dictionaries with lists

Error handling

Importing the io Module

In order to use the Try-Except error handling, the io module had to be imported by the program. This also allowed the use of the "Exception" class to create customized error messages that are more meaningful to the user. See Figure 1.

Declaring and Initializing file variable

The only change to declaring and initializing variables was that the "file" variable was initialized to "io.TextIOWrapper" instead of "None". See Figure 2.

Adding error handling

Both Try-Except and customized error handling were used. Try-Except error handling allows multiple types of errors to be checked for. "FileNotFoundError" was used to detect if "Enrollments.json" was present. If not, a specific error message about "Enrollments.json" was printed to the screen. If "Enrollments.json" was found, but another issue occurred while reading the file, a more generic error message was displayed. The "finally" section verified that "file" was closed. If not, it was closed to prevent future errors of opening a file that was already open from occurring. See Figure 5. "ValueError" was used in conjunction with the isalpha() function to display a customized error message after checking if any numerical characters were used in the first and last name. See Figure 4.

```
41  try:
42      file = open(FILE_NAME, "r")
43      students = json.load(file)
44      file.close()
45  except FileNotFoundError as e:
46      print("\nEnrollments.json file must exist before running this script.\n")
47      print("File not found.")
48      print(e, e.__doc__, type(e), sep='\n')
49  except Exception as e:
50      print("There was a non-specific error.\n")
51      print("-- Technical Error Message --")
52      print(e, e.__doc__, type(e), sep='\n')
53  finally:
54      if not file.closed:
55          file.close()
```

Figure 5: Assignment05.py Using dictionaries with lists

Printing to .csv file

One item on the homework requirements indicated to save a student's first and last name and course name to a coma-separated file. I thought this might have been a mistake, but decided it was good practice and completed the task. First, I initialized "FILE_NAME_CSV" to "Enrollments_csv.csv" and "file_csv" to "None". See Figure 1. Due to the changes described, "csv_data" was no longer being used. I added a line of code to the print loop used for Option 3 that created a text string on multiple lines from elements of Students[] and set it equal to "csv_data". "file_csv" was opened, written to and then closed. Try-Except error handling was added as well. See Figure 6.

```
            print("The following data was saved to file!")
            for student in students:
                print(f"{student["FirstName"]} {student["LastName"]} is enrolled in {student["CourseName"]}")

                # creates text string for .csv file
                csv_data += f"{student["FirstName"]}, {student["LastName"]}, {student["CourseName"]}\n"

        except Exception as e:
            print("There was a non-specific error.\n")
            print("-- Technical Error Message --")
            print(e, e.__doc__, type(e), sep='\n')
        finally:
            if not file.closed:
                file.close()

        # Assignment sheet indicated program must save date to a comma separated string file.
        try:
            file_csv = open(FILE_NAME_CSV, "w")
            file_csv.write(csv_data)
            file_csv.close()

        except Exception as e:
            print("There was a non-specific error.\n")
            print("-- Technical Error Message --")
            print(e, e.__doc__, type(e), sep='\n')
        finally:
            if not file_csv.closed:
                file_csv.close()
        continue

```

Figure 6: Assignment05.py writing to .csv file

GitHub

A GitHub account with a repository for classwork was created. It can be found at:

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

Summary

In Module 5, I learned about working with dictionaries, JSON files, error handling and the GitHub website. All of these topics improve the functionality, usability and shareability of our code.