Miles Devine

7/31/2024

IT FDN 110

Assignment 05- Advanced Collections and Error Handling

Introduction

This week's module took us through a look at the fundamental concepts of data structures, file handling, and error handling. We learned how to write data to a JSON file which demonstrated how knowing what file format you need to use in a specific situation can cut down on how many lines of code your program has. Finally, we learned about GitHub and its significance in collaborative software development.

The Assignment

The task we were given for this assignment was to create a Python program that uses constants, variables, print statements, flow control, conditional logic, dictionaries, and lists to display user entered information and save that information to a JSON file.

As usual, we started our code with a header that contained important information for the user including the title, description of the program, and a change log. Next, we imported the json and JSONDecodeError modules (Fig. 1).

```
# ------------------------------------------------------------------------------------------ #
# Title: Assignment05
# Desc: This assignment demonstrates using dictionaries, files, and exception handling
#   Miles Devine, 7/31/2024, Created script
# ------------------------------------------------------------------------------------------ #
import json
from json import JSONDecodeError
```

Fig. 1- (Header with important information and imported modules)

Then we went on to define the constants and variables (including type hints) that are used throughout the program. This makes the users aware of what types of values are expected to be stored in them and provides some organization. This in turn helps the user understand what is happening as they move through the program (Fig. 2).

```
# Define the Data Constants
MENU: str = '''
---- 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.
----------------------------------------
'''
# Define the Data Constants
FILE_NAME: str = "Enrollments.json"

# Define the Data Variables and constants
student_first_name: str = ''  # Holds the first name of a student entered by the user.
student_last_name: str = ''  # Holds the last name of a student entered by the user.
course_name: str = ''  # Holds the name of a course entered by the user.
student_data: dict = {}  # one row of student data
students = []  #Holds a two-dimensional table of student data
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
```

Fig. 2-(Constant and variable definition)

Per the requirements, we had to read the data in the Enrollments.json file when the program opens. We utilized the load function in conjunction with the open function to open the file we needed to read from and to load the data into the students variable. We then closed the file using the close function (Fig 3). This required much less code than the csv method we used for last week's assignment (Fig. 4).

```
    file = open(FILE_NAME, "r")
    students = json.load(file) #Puts the contents of the file into the students variable
    file.close()
```

Fig. 3-(Reading the data from a JSON file and adding it to a two-dimensional list)

```
#Opens the Enrollments.csv file and reads the data into a 2 dimensional list (table)
file_obj = open(FILE_NAME, 'r')
for row in file_obj.readlines():
    row_data = row.strip().split(',')
    student_first_name = row_data[0]
    student_last_name = row_data[1]
    course_name = row_data[2]
    student_row = [student_first_name, student_last_name, course_name]
    students.append(student_row)
```

Fig. 4-(Reading the data from a csv file and adding it to a two-dimensional list)

Then we printed the menu for the user to select from and used a while loop to give the user as many opportunities to enter data as they want. Based on the user's input we used if statements to perform certain functions like allowing the user to enter data, display the data, save the data, and exit the program (Fig. 4). We added error handling throughout the menu choices that made for a more user friendly experience and ensured that the inputs we got from the user were what we expected.

```
while (True):

    # Present the menu of choices
    print(MENU)
    menu_choice = input("What would you like to do: \n")

    # Input user data
    if menu_choice == '1':
        try:
            student_first_name = input("What is the student's first name? ") #Gathers student first name from user
            if not student_first_name.isalpha():
                raise ValueError("\nThe first name should only contain letters.\n") #Displays custom error message
        except ValueError as e:
            print(e)
            print('----------Technical Error----------')
            print(e.__doc__, type(e), sep='\n') #Displays the technical error message
            if not student_first_name.isalpha():
                student_first_name = input("\nWhat is the student's first name? ") #Prompts the user to enter their input again
        try:
            student_last_name = input("What is the student's last name? ")
            if not student_last_name.isalpha():
                raise ValueError("\nThe last name should only contain letters.\n") #Displays custom error message
        except ValueError as e:
            print(e)
            print('---------Technical Error---------')
            print(e.__doc__, type(e), sep='\n') #Displays the technical error message
            if not student_last_name.isalpha():
                student_last_name = input("What is the student's last name? ") #Prompts the user to enter their input again

        course_name = input("What is the course name? ") #Gathers the course name from the user

        student_data = {"firstname": student_first_name,
                        "lastname": student_last_name,
                        "coursename": course_name}
        students.append(student_data)
        continue
```

Fig. 4-(Code block for menu choice 1)

I then tested the program in PyCharm as well as the command console to make sure that the program will work for a user regardless of what program they are using to run it.

Summary

In summary, this assignment reinforced some of the lessons learned in previous weeks while giving us practice with the new lessons from this module. Our code has started to become more complex, and the little things are really starting to matter. Using the debugger has helped a lot with troubleshooting as you can walk down through each line of your code and see what is happening. This assignment also demonstrated the importance of choosing the appropriate file type for each situation you're working in. I'm looking forward to next week's assignment.