Jake Jake Velasco

Feb. 13th, 2024

IT FDN 110 A Wi 24

Assignment 05

Advanced Collections & Error Handling

Introduction

In this assignment the goal is to demonstrate my understanding of data structures and file handling in Python. Specifically showing the differences between List and Dictionaries, the use of indexes & keys within my code and utilize the to read & write function to add data onto a file from a dictionary or pull data from a JSON file. Displaying skills on how to create Structured Error Handling by the Try-Expect construct method. Lastly, I will dive into the use of Github and how it is used within collaborative team environments.

Redefining the Variables and JSON File Processing

Very similar to the previous assignment, the initial steps were to define/set the data variables and constants to have the capability to read/write JSON files. Added import JSON at the very beginning, eliminated the previous data to read JSON file.

```
# 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.
json_data: str = "" #Holds the data for a json file
file = None  # Holds a reference to an opened file.
menu_choice: str  # Hold the choice made by the user.
student_data: dict = {}  # one row of student data
students: list = []  # a table of student data

# When the program starts, read the file data into a list of lists (table)
# Extr the data from the file
file = open(FILE_NAME, "r")
students = json.load(file)
file.close()
```

Then targeted the student_data variables within in the read segment, input user segment, and throughout the rest of the code and formatted them as a dictionary and identified the keys within them. I leveraged the “Mod05-Lab01-WorkingWithDictionariesAndFiles” & the “Mod05-Lab02-WorkingWithJSONFile”. I also noticed that the for loop and csv data variable was no longer needed when saving the data to a file.

```
# Present and Process the data
while (True):

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

    # Input user data
    if menu_choice == "1":  # This will not work if it is an integer!
        student_first_name = input("Enter the student's first name: ")
        student_last_name = input("Enter the student's last name: ")
        course_name = input("Please enter the name of the course: ")
        student_data =
{"firstname":student_first_name,"lastname":student_last_name,"course":course_
name}
        students.append(student_data)
        print(f"You have registered {student_first_name} {student_last_name}
for {course_name}.")
        continue

    # Present the current data
    elif menu_choice == "2":

        # Process the data to create and display a custom message
        print("-"*50)
        for student in students:
            print(f"Student {student['firstname']} {student['lastname']} is
enrolled in {student['course']}")
        print("-"*50)
        continue

    # Save the data to a file
    elif menu_choice == "3":
        file = open(FILE_NAME, "w")
        json.dump(students,file)
        file.close()
        print("The following data was saved to file!")
        print(json_data)
        continue

    # Stop the loop
    elif menu_choice == "4":
        break  # out of the loop
    else:
        print("Please only choose option 1, 2, or 3")

print("Program Ended")
```

Error Handling 101

To ensure that the program that the program provides a structured error handling when the file is read into the list of dictionary rows, I leveraged Demo03-UsingExceptionHandling and the Mod05-Lab03-WorkingWithExceptions. After hours of turmoil, arrived at the following code and found success.

```
# When the program starts, read the file data into a list of lists (table)
# Extr the data from the file
try:
    file = open(FILE_NAME, "r")
    students = json.load(file)
except FileNotFoundError:
    print("Unfortunately, the file does not exist")
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:
    print("Closing File")
    file.close()
```

The program provides structured error handling when the user enters a first name & last name and force the user to input letters and not numbers first but using the isalpha function in combination with an if not clause. I also used the raise Value error function to provide a more human response versus Python's automated built in error message.

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

# Input user data
if menu_choice == "1":  # This will not work if it is an integer!
    try:
        student_first_name = input("Enter the student's first name: ")
        if not student_first_name.isalpha():
            raise ValueError("Student first name must only contain alphabetic
letters.")
        student_last_name = input("Enter the student's last name: ")
        if not student_last_name.isalpha():
            raise ValueError("Student last name must only contain alphabetic
letters.")
        course_name = input("Please enter the name of the course: ")
        student_data =
{"firstname":student_first_name,"lastname":student_last_name,"course":course_
name}
        students.append(student_data)
        print(f"You have registered {student_first_name} {student_last_name}
for {course_name}.")
        continue
    except ValueError as e:
        print("User entered incorrect info, try again please.")
        continue
```

Next up was error handling the save the data to a file and ensure it provides structured error handling when the dictionary rows are written to the file. It started to come naturally at this point when using the try/exception/finally functions.

```python
# Save the data to a file
elif menu_choice == "3":
    try:
        file = open(FILE_NAME, "w")
        json.dump(students,file)
    except Exception as e:
        print("Error dectected when writing to the file.")
        print(e, e.__doc__)
    finally:
        print("The following data was saved to file!")
        file.close()
        continue
```

Summary

Overall, the most difficult hurdle of this assignment was formatting the try/exception method when error handling. I hope in the lecture modules to come that this will come more naturally.