Ben Stafford

14 February 2024

IT FDN 110 A

Assignment 05

https://github.com/bstaff99/IntroToProg-Python-Mod05

Assignment 5: Dictionaries and JSONs

Introduction

Assignment 5 built on the work of assignment 4 but with the introduction of Dictionaries and JSON files to streamline data input collection. We also began the process of learning error handling using Try-Except blocks.

Re-writing Script for JSON Files

This assignment begins with code that is set up to use a CSV file, so the first step was converting it to use the JSON format. I added "import json" to the top of the document and replaced the beginning code that imports the file and added the import json code. Moving on to Option 1, I reformatted the student_data variable into a dictionary that is appended to "students". Initial runs gave me errors and I made sure to go into the JSON sample file and unify my keys with the ones in the file. Option 2 was basically the same, but again with the print statement adjusted to reflect the key names rather than the numbered rows of a list. For option 3, I replaced the code with the json.dump command, and again printed the confirmation for each student. Option 4 remained unchanged. All options worked correctly in PyCharm and CMD.

Error Handling

For better or worse, I decided to rewrite the full script first and then incorporate the try-except structured error handling. For the initial opening of the file, I put the code under a "try" block, and added the FileNotFoundError and Exception error blocks beneath. For the user inputs in Option 1, I again put the whole block of existing code under "try" and added the ValueError and Exception blocks beneath. I added "If" statements beneath the name entry options that raise the ValueError if non-letters are used. **(Figure 1)** After incorporating these blocks, the program worked correctly in both PyCharm and CMD, and properly saved the JSON file.

```
try:
    student_first_name = input("Enter the student's first name: ")
    if not student_first_name.isalpha():
        raise ValueError
    student_last_name = input("Enter the student's last name: ")
    if not student_first_name.isalpha():
        raise ValueError
    course_name = input("Please enter the name of the course: ")
    student_data = {"FirstName": student_first_name, "LastName": student_last_name, "CourseName": 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("Please make sure to use only letters.\n")
    print("Technical Explanation: ")
    print(e, e.__doc__, type(e), sep='\n')
except Exception as e:
    print("An unexpected error occurred.\n")
    print("Technical Explanation: ")
    print(e, e.__doc__, type(e), sep='\n')
```

Figure 1: User Input with Try-Except Blocks added

Summary

I found that the things we learned in this week's assignment really helped to streamline the coding process and helped to make our code a lot cleaner. I find the Try-Except blocks exciting, because I see future implications using these to execute different code to make the program still useable if run conditions aren't met (ie. Trying to read a file, and, if a FileNotFoundError is triggered, the code beneath the error *creates* the file for you).