Nasmil Valera Cuevas

February 12th, 2024

Foundations of Programming, Python

Assignment 05

<https://github.com/nasmilvc/IntroToProg-Python-Mod05>

**Advanced Collections and Error Handling**

**Introduction**

In this assignment, I will go over how I created a Python program that demonstrates the use of data processing using lists and dictionaries, understanding when to use JavaScript Object Notation (JSON) files and the importance of Structured Error Handling with the Try-Except approach. This work was completed with the source material provided by Professor Randal Root, helpful demonstrations by Anubhaw Arya and a handful of external videos linked throughout the module.

**Drafting the Code**

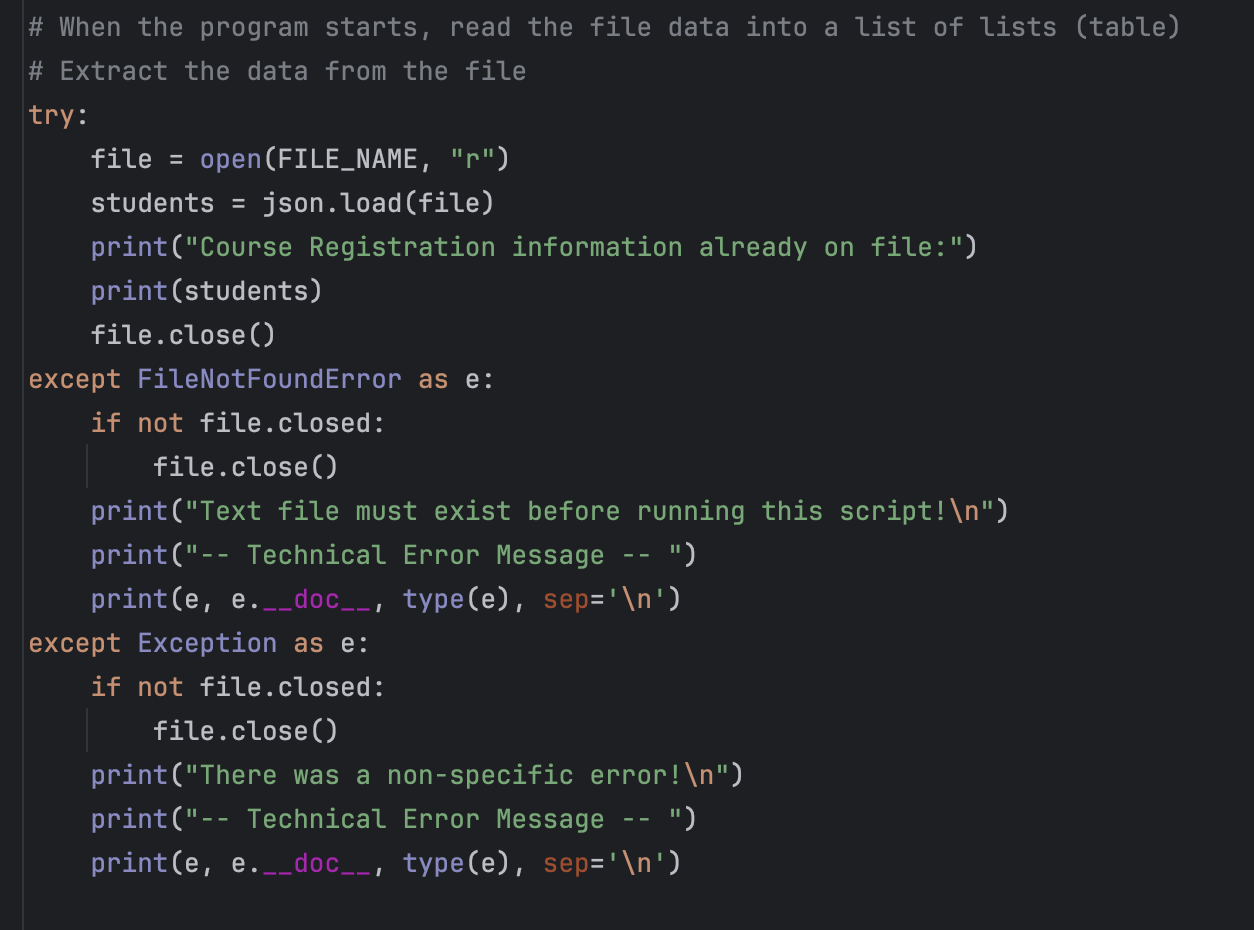
I began by opening and reviewing the starter file Assignment05-Starter.py, and since I knew I would be working with JSON data, I imported the built-in JSON module onto the script.

# Defining Data Items

I updated the variables and constants to match the assignment data types, defined them, and reviewed them to ensure they matched the assignment instructions.

# Performing and Printing Operations

I used a try-except construct and an open() function to open the file “Enrollments.json” on read mode. This resulted in all the lines of the file to be read and returned as a list of dictionaries (Figure 1). In order for the program to read the file, I had to put some starting data into the file. If I had not done this, it would have shown an error. Using the try-except construct in this instance allows for catching the error of opening the file on read mode without the file existing, and a non-specific error. Accounting for non-specific errors improves the readability of any errors that may occur. I also used a print() function to show the user the custom error of the exception (e), a brief description of the exception type (e.\_\_doc\_\_), and the type of error(type(e) in separated by a new line.

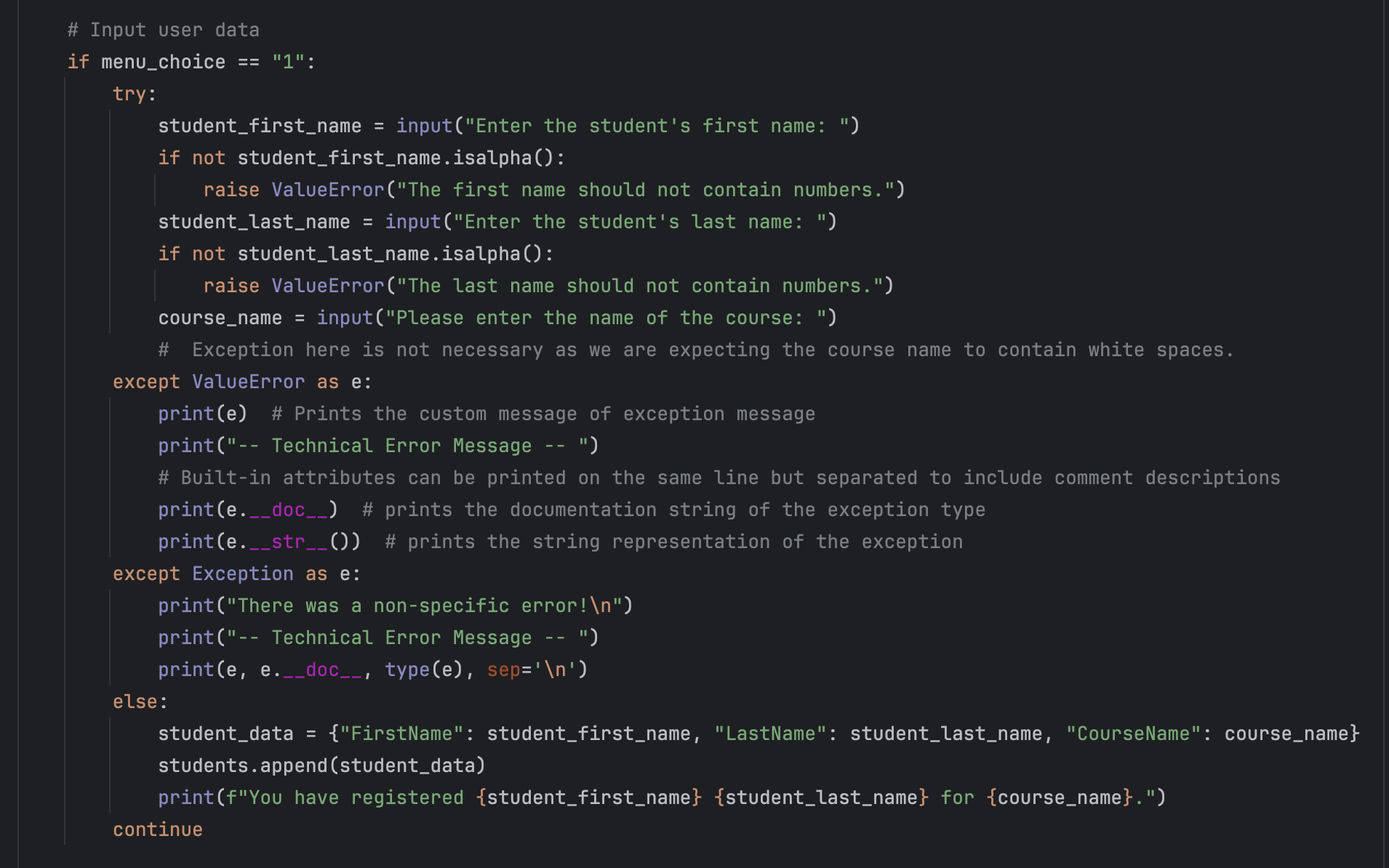


**Figure 1: Screenshot of data already in file in a table (list of lists) format.**

Presenting the menu of choices

I used a while loop, printed the menu, and prompted the user to choose a menu option. For menu choice 1, I used a structured error handling when the user enters a first name and a last name by adding the .isalpha() function, which ensure that all the entered values are alphabetical. For the course name input, I looked for a function what would ensure the input is limited to letter and numbers and I found the .isalnum() function. I ended up not using this function because it doesn’t allow for other symbols or, white spaces. I then complete the try-except construct with exception sections that insert e as a ValueError and as an Exception in case any other errors occur. Within the exception sections, there’s three key things I will be printing: e, as a custom message of the exception, e.\_\_doc\_\_, as a brief description of the exception type and e.\_\_str\_\_, as a string that provides details about the exception, including its error message. In this case, printing e and e.\_\_str\_\_ show the same result. Nevertheless, this will not always be the case as my code evolves, so I decided to leave the print(e) command in.

I wanted the user to return to the top of the while loop if they encounter an error, so I formatted the student\_data dictionary in the else statement so it only prints if there are no errors (Figure 2).



**Figure 2: Screenshot of menu\_choice 1, showing try-except constructs**

For menu choice 2, I used a for loop to display all the data that is currently in the list. The assignment did not specify the format it wanted the data, so I chose to present the data in a string format by printing all the values of the keys in the dictionary.

For menu choice 3, I opened the file in write mode in a try-except construct. I used the json.dump() function to add the information in my list of lists to the file and I proceeded to close the file. Then, I used a for loop to print the data that was saved in every row in a dictionary format. I added exceptions to tell the user to check the data is in a valid JSON format and a catch-all for any other error that may occur. I also added a last exception to make sure the file is closed so all the data is saved. Menu choice 4 breaks the user out of the loop and after the loop is closed, the program notifies the user that the program was ended.

A computer screen shot of a program code

Description automatically generated

**Figure 3. Screenshot of menu choice 3, showing the try-except constructs and the use of the json.dump() function**

# Running the Script in OS Command

I opened the Terminal console on my computer, navigated across directories and used the python3 command with file Assignment05.py to run my script. I followed the user prompts, and the script ran as expected (Figure 4).

A screen shot of a computer

Description automatically generated

**Figure 4. Screenshot of file "Enrollments.json” text file with data from the program.**

A screenshot of a computer program

Description automatically generated

**Figure 5: Screenshot showing a portion of the program Assignment05.py running currently in the MacOS Terminal**

**Summary**

I was able to successfully create a Python program that demonstrates the use of data processing using lists and dictionaries, understanding when to use JavaScript Object Notation (JSON) files and the importance of Structured Error Handling in the Try-Except approach. I also used an array of functions streamline the code and push errors to guide code users to follow prompts according to the format.