Jordan Sellers

2/7/2024

Foundations of Python: Programming

Assignment 04

Module 5: Advanced Collections and Error Handling

# Introduction

The student registration program has been upgraded to save data using JSON files. It now includes error responses to prevent the program from cashing and provide information to the user.

# Dictionaries

A dictionary is a pairing of a key and a value. The key is the name given to the value, and the value is the information that is stored. The key and value are separated by a colon: “key”: “Value”. Dictionaries can be strung together in rows (contained in brackets), and the rows added to a list (in brackets) to form a table [Figure 1]. The key is a string, and the value can be one of a number of different types: string, integer, float, boolean, etc.

*Figure 1: a table of dictionaries*

Dictionaries are useful because they can be arranged in a grid, similar to a spreadsheet. This makes it easier to work with they data. If a specific value needs to be referenced, it can be called up using the key instead of by indexing, as in a list. Data can be added or removed, or individual values can be changed.

# JSON Files

JSON (Java Script Object Notation) files are used for working with data. Despite being associated with Java Script, they can be used for other computer languages too. Applications for JSON files often involve manipulating and sending data, such as when communicating between servers and clients or preparing data for analysis. They are easier to read than other data file types because JSON more closely resembles spoken language.

# Extracting and Processing Data

In relation to dictionaries in Python, JSON files are convenient to work with because their format for storing data is the same as the format used for dictionaries. When the file data is read into the program it can be stored as a table without needing to be reformatted. The program opens the file in write mode, the file data is loaded into the program and assigned to a variable, and then the file is closed [Image 2]. The data is automatically formatted as a table.

This program allows the user to add data. The inputted user data is assigned to variables, and the variables are assigned to dictionaries, which are formatted in a row contained in brackets. This row is then appended to the file data [Figure 2].



*Figure 2: reading a JSON file*

The file data can also be displayed to the user. The dictionaries in a given row are formatted into a string, and the rows are printed to the screen one by one in a for-loop [insert image].

Saving data to the file is simpler too. The file data does not need be reformatted. The file is opened in write mode, the file data is then “dumped” directly into the file by referencing the variable, and the file is closed [Figure 3].



*Figure 3: saving to a JSON file*

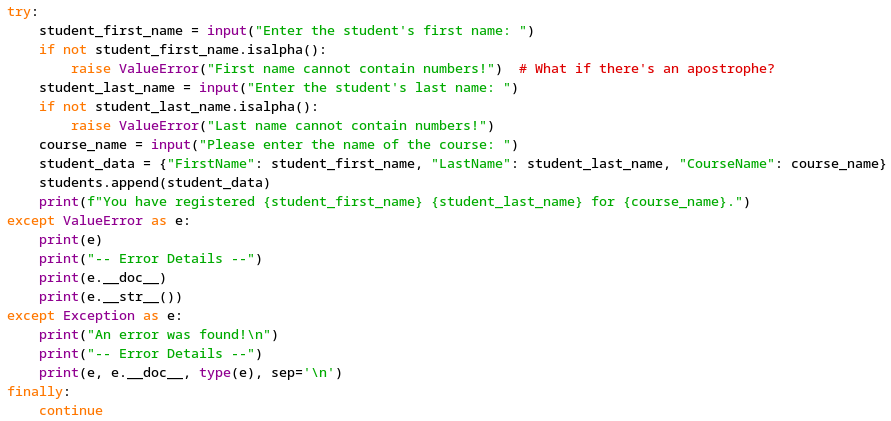
**Error Handling**

Another new feature of the program is error handling. Instead of the program crashing when there is an error, it handles errors using try-except. “Try” is inserted before a section of code that could result in an error, which is then indented. After that, certain scenarios are specified using “except.” These instruct the program what to do when there is an error. There are instructions for what to do with an anticipated error, as well as one that is new. This helps the programmer exercise more control over the error-handling process, instead of allowing python to operate the way it normally would.

Try-except is an opportunity to print a message to the screen explaining what happened in a way an average user could under stand. As well as a custom print statement, a statement giving details about the error is printed giving more details about the error itself. This often takes advantage of the error reporting Python already does. The exception class (type of error) is assigned a variable and printed to the screen. This causes a message about the error generated by Python to be displayed to the user.

The last situation to be specified is “finally.” This instructs the program what to do whether there is an error or not. This could be a message to the user, or continuing with the rest of the program.

Here is an example of a try-except block when collecting user data [Figure 4]:



*Figure 4: Handling exceptions while collecting user data*

**Summary**

Dictionaries pair values with keys as a way to easily store and reference information. JSON files store data in a way that is easy to read and convenient for accessing and manipulating information later. Error handling prevents errors from crashing the program, provides helpful information to the user, and allows more control over how errors are handled.

Link to my GitHub: <https://github.com/jordan-aloysius>