Jocelyn Munoz

July 29, 2024

Foundations of Programming: Python

Assignment 05

<https://github.com/Joce9322/IntroToProg-Python>

Error Handling and JSON Files

**Introduction**

This document reviews the steps for incorporating error handling to a program created in Assignment 04 for processing student registration data. The program also transitions to using JSON files instead of the previously used csv files. This knowledge document will cover the differences in code for the different processing files.

The program continues to build off Assignments 3 and 4, and the defined variables and constants are still the same. Please reference those documents for a review of the program variables, constants, and inputs/outputs.

**JSON Files**

JavaScript Object Notation (JSON) files are commonly used by web developers to store key-value pairs and arrays. JSON files are capable of storing various and complex data types in a readable format. JSON files are mainly used for data interchange between different forms of technology, and JSON files are compatible with many different languages including Python.

To load JSON files in Python, we can use the built-in module that makes it easy to work with JSON files. First, we want to import code from Python’s JSON module. Then, to read data existing in a file, we can use the json.load() method to parse the data in the file into a Python list of dictionaries (students). Similar to working with csv files in Python, we want to make sure to use the open() function in read “r” mode before using the json.load() method.

To write data in a JSON file, we still want to open() the file in the write “w” mode. Then, we use the json.dump() method to write our data directly to the JSON file (*Figure 1*).

A screenshot of a computer

Description automatically generated

A computer screen with text and numbers

Description automatically generated

*Figure 1*. Loading and dumping data from and to a JSON file using the built-in Python JSON module.

**Error Handling**

In this program, we introduced structured error handling using a few try-except constructs. The error handling methods were introduced to manage errors that are likely to be introduced by the people using this program. The advantages to integrating structured error handling to our program include providing user-friendly error messages to the program’s user while also allowing for an organized management of error statements to the developer. The try-except block is specifically helpful in this program because it allows us, the developers, to customize the error message. We introduced try-except blocks in three areas:

* When a JSON file is read into the program
* When the user inputs the first and last name of the registered student
* When the dictionary rows (our data) are written into a JSON file

Loading Files

In our first try-except section, we used a FileNotFoundError when the JSON file is being read into the program. In this except block, we catch the FileNotFoundError exception if the file doesn’t already exist before running the script. We also use a second except block for the generic exception which acts as a catch-all for any other exceptions that may occur. It includes a general error message along with the documentation string and the type of the exception. If the first except block catches an error, the second except block won’t run. Lastly, we used a “if file.closed ==False” statement to check that the file is closed. If the file is still open, the program then closes it to prevent any issues that may come up later in the program from having an open file (*Figure 2*).

A computer screen shot of a program code

Description automatically generated

*Figure 2*. Try-except blocks used during file loading.

User Input Errors

In our second try-except section, our registration program asks the user to input student name data where we only want alphabetical characters. Therefore, we introduced structured error handling when the user enters the student’s first and last name. We used a ValueError block that catches and handles a ValueError exception that is raised if the input contains numbers in the first and last name. We included print statements for the custom error message “(e)” and the technical error message with the docstring and the string representation. We also included a second except block for the generic exception with a custom message, the docstring, and the string representation (*Figure 3*).

A computer screen shot of text

Description automatically generated

*Figure 3*. Try-except blocks used when asking the user for input first and last name data.

Writing in Files

Finally, in our last try-except section, we use a TypeError when data from our program is being written into a JSON file. In the first except block, we catch the TypeError if the data is not in a valid JSON format. We also include a second except block for the generic exception and a “if file.closed ==False” statement (*Figure 4*).

**A computer screen shot of a program

Description automatically generated**

*Figure 4*. Try-except block used when writing data to a JSON file.

**Summary**

This program utilizes structured error handling to catch errors that occur during the program execution. Error handling allows us to prevent the program from unexpectedly crashing when an error occurs and provides clear and meaningful feedback to the program’s user about the error. In this program, we mainly focused on using try-except blocks in combination with specific types of exceptions. The try block is used to place code that might raise an exception and the except block is where the program handles the exception if it occurs. The specific types of excpetions we used were: FileNotFoundError; ValueError; and TypeError.