Jonathan Den Haan

11/26/2022

Foundations of Programming: Python

Assignment 07

**Module 7**

**Introduction:**

The two main ideas looked at in this module are the use of binary data, and structured error handling. Binary data is a form of data storage that is only readable by computers, and is useful for data that will only ever be accessed by a script. Structured error handling is a way to ensure that scripts keep running even if invalid entries are used because it gives the script an option as to what to do in those cases, instead of just terminating. This module also looks at reading text files in python using built in functions that can make it easy to bring data in and save data to file.

**Labs:**

Lab07-A

Text

Description automatically generated

This snip shows the two functions I wrote to allow the script to read from MathIn and MathOut.txt documents. The first function, read\_file opens the file and uses an index to assign the two values to the variables needed in the math operations functions. These variables are returned by the function, and when it is called they are assigned as intNumA and intNumB, which are the arguments that the math functions take.

The second function takes the results argument which is a list, and converts it to a string, iterating through the list and adding a comma. When I just had the program write the str(list), it removed the commas, so I found the need to add them in the write function. This leaves a trailing comma which I think I’d have to write another block of code to remove, and since I’m unsure how I’ll leave it at that.

Lab07-B

Text

Description automatically generated

This script works by substituting the use of a carefully worded function that writes each element in a list to file with a binary data dump that simplifies the process. The functions instead call on these .dat files to load and save data. The other difference from LABA is that we use program arguments to run the script. If the user enters “IO” as an argument, they get the functions called so that they are able to enter the data and save it to file. If the user enters “calc” as an argument they prompt the script to use the saved data to perform the calculations and save to the output file. This is very useful for calling a script that may have a lot of functionality but you only need it to do a specific task at that moment. Much like how you would open MS Word to write an essay, but you’d want to open MS PowerPoint to make a presentation instead of opening the full office suite every time.

Lab07 – C

Text

Description automatically generated

This script works the same way as the script in Lab07 B except that it includes some error handling functionality. This allows the script to identify errors without ending the program. If an invalid character is entered, or an unknown file is opened, (among other errors) the script will tell the user what the error is and prompt them to enter a different value. In this way the script will not just end when an error is found, it is flexible.

Structured Error Handling research:

[Handling Errors in Python. Best practices for gracefully handling… | by Julie Elise | Better Programming](https://betterprogramming.pub/handling-errors-in-python-9f1b32952423)

A good overview of error handling with some nice concise explanations of what will happen with certain uses cases, as well as do’s and don’ts of error handling.

[Exception Handling in Python: Catch and Handle Errors with valueerror, try except, typeerror, & raise in Python | DataCamp](https://www.datacamp.com/tutorial/exception-handling-python?irclickid=TOTzpQyxoxyNWz-3FVWSXQYFUkA0bo3xpy1zUo0&irgwc=1&utm_medium=affiliate&utm_source=impact&utm_campaign=000000_1-2003851_2-mix_3-all_4-na_5-na_6-na_7-mp_8-affl-ip_9-na_10-bau_11-Bing%20Rebates%20by%20Microsoft&utm_content=BANNER)

Good site with diagrams of the try, except, finally clause structure and how it works. Clarifies the difference between errors and exceptions. Errors are within the script and will always cause it to run poorly, exceptions are typically user generated, or otherwise caused by factors outside the scope of the code in the script. Covers all the built in exceptions/standard errors in Python.

Pickling Research:

<https://www.synopsys.com/blogs/software-security/python-pickling/>

Interesting website that mostly deals with the security risks of loading in pickle data from an untrusted source, and how to mitigate those risks. Does not explain the concept very well because the examples are too generic and there aren’t enough of them.

<https://www.afternerd.com/blog/python-pickle/>

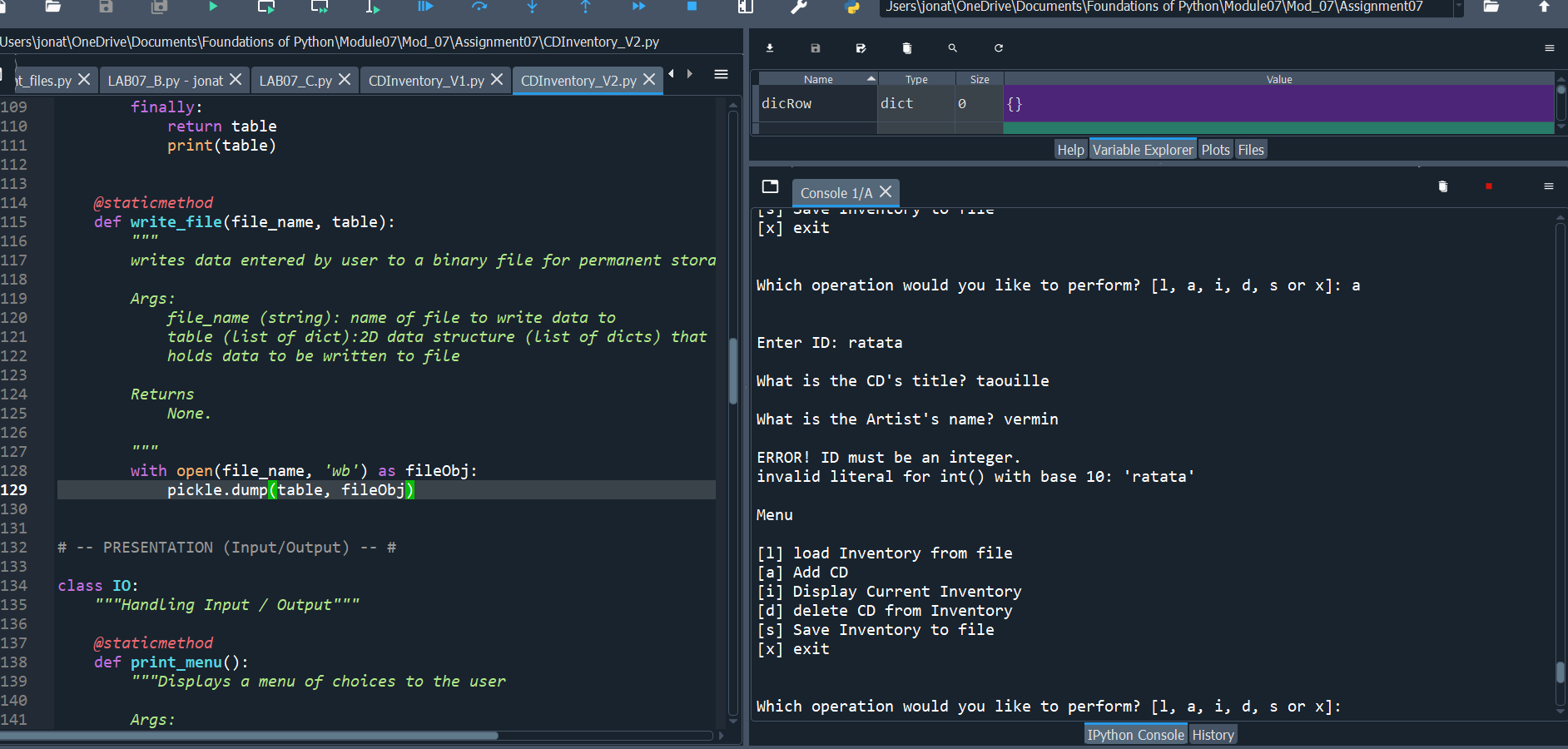
A great blog post describing pickling with lots of examples. Gives a good rational for why you might choose pickling over text files. Number one, text files take more space which can cause network or memory congestion. Number two, since text files often have a specific format, it means that only a script that is built to unpack that format can use the data. Not so with pickling since any script can pickle/un pickle data and use it. Also compares pickling in python with other ways of serializing data.

<https://realpython.com/python-pickle-module/>

Good general overview of the pickling technique.

Assignment07

Running in Spyder:



In terminal Window:

Text

Description automatically generated

The most difficult part of this script was understanding that the pickle.load is not as straightforward as in the examples in module 07. Since the table was still formatted as a list of dictionary items, the pickled table needed to be unpacked back into the dictionary format. If I had changed this at the root level, ie, not having a list of dictionaries, this might have been easier.

Other than that, this script works as intended, with excellent error handling and the use of the binary data types.

The steps I took were 1) add error handling to any spots were I anticipated that an error could happen 2) test error handling thoroughly 3) Change the data file structure from text files to binary file types. 4)test again until it works!

**Summary:**

This module looked at using binary files in addition to text files. We discovered that binary files do not need to be formatted or unpacked when saving or loading from file. This make them more of a data dump that is quick to use, especially when you don’t actually need to open the file outside of running the script. We also continued to create functions to organize our code, and we learned to use Structured Error Handling to make our scripts handle exceptions in the case of user error.