Vernon Jackson

Aug 22, 2021

IT FDN 110

Assignment 07

# Modify Program to Store CD Inventory Data

# Introduction

In this document I will be explaining how I added code to the CD Inventory Starter program for assignment 06, that modifies the script as required to use binary data as the permanent data store and adds exception handling. Areas of focus for this document include 1) briefly covering the fundamentals that I learned in this module 2) the steps I took to organize and create the script and 3) a summary of my experience in creating this program.

# Coding Fundamentals

Within this assignment, I learned several new fundamentals of coding, including structured error handling (exceptions) and pickling. I will briefly summarize my own definitions of each used in the assignment in order to substantiate my learning.

**Exceptions:** This is a type of error that occurs when code results in an error when trying to execute the code. Python will show you which type of exception error is occurring, such as a Value Error, Type Error, or File Not Found Error. Handling exceptions allows your coding to be more robust and allows you to catch errors during runtime that can be controlled. The resource that was most useful for me was the *Datacamp: Exception and Error Handling in Python*[[1]](#footnote-1). I found this useful because it explained why exception handling was useful for coding, provided real examples and how to reproduce, and explained the differences between general errors and exceptions in Python. The other resources that I utilized, such as Stack Overflow, provided examples that would be useful for specific scenarios and did not provide details on general exception handling.

**Pickling:** Pickling is where a Python object is converted into binary data. The reverse process of unpickling is when you are converting binary data back into a Python object. The resource that was most useful for me again from Datacamp, called *Pickle in Python: Object Serialization* [[2]](#footnote-2). I found this useful as it provides a summary on what pickling is, when you should and should not use it, how to pickle objects/files, and explained the difference between JSON and Pickle.

# Updating the Program

In order to create the script, I first began with the *CDInventory.py* script from Assignment 06 and updated this to use binary data for permanent data store and added structured error handling. The main coding fundamentals utilized in this assignment are pickling for using binary data and exception handling to handle errors during user interaction for type casting and file handling. The following functionalities were modified:

**Binary Data Usage for Read/Write Functions:** In order to use binary data for the permanent data storage I modified the *read\_file* and *write\_file* functions in the *FileProcessor* class. In the *read\_file* function, I used pickling to load the file and append the file values to the *lstTble* list. In the *write\_file* function, I used pickling to dump or write the *lstTbl* list into the the *CDInventory.dat* file. I used the *wb+* file mode when opening the data file, as this is used in the start of the program to both write and also create a file if one does not exist.

**Exception Handling for Adding Inventory:** In order to add exception handling for Value Errors within this script for the *add\_choice* function, I needed to first create a while loop within the function, then added the integer input for the ‘ID’ as the try clause which gets user input for new ID. I then added the Except statement for ValueError to print “Please enter a number!”

**Exception Handling for Deleting Inventory:** In order to add exception handling for Value Errors within this script for deleting inventory, I needed to first create a while loop within the main program which handles input/output under process 3.5. I then added the integer input for the ‘ID’ to delete as the try clause, then added the Except statement for ValueError to print “Please enter an ID number!”

**Exception Handling for File Access Operations**: In order to add exception handling for File Not Found Errors within this script for process 1 which reads currently saved Inventory, I added a try statement to read an existing binary file with a clause that executes the *read\_file* function in the *FileProcessor* class. If the file is not found, I added the except clause for *FileNotFound* errors, which executes the *write\_file* function in the *FileProcessor* class which creates and writes to a file if one does not exist.

To test this script, I ran this as a Python script through Spyder IDE and Anaconda Prompt. Executing the script through both, showed that it had the intended functionality of using binary data for permanent data storage and structured error handling. Follow this link to view the script on GitHub: <https://github.com/Veejster/Assignment_07.git>.

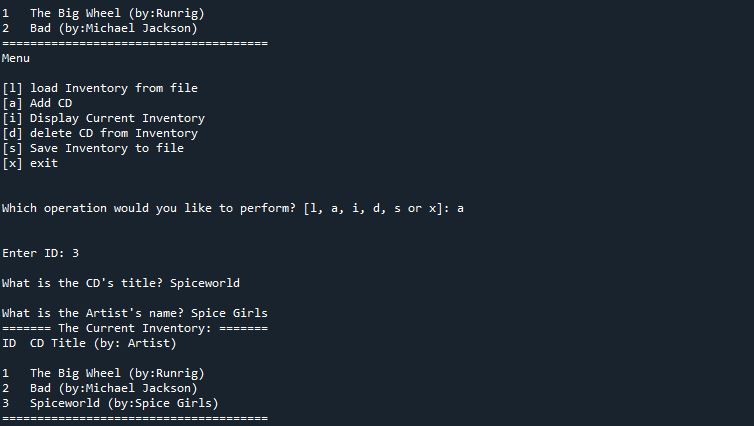


Figure 1 - Showing the program being executed through Spyder.

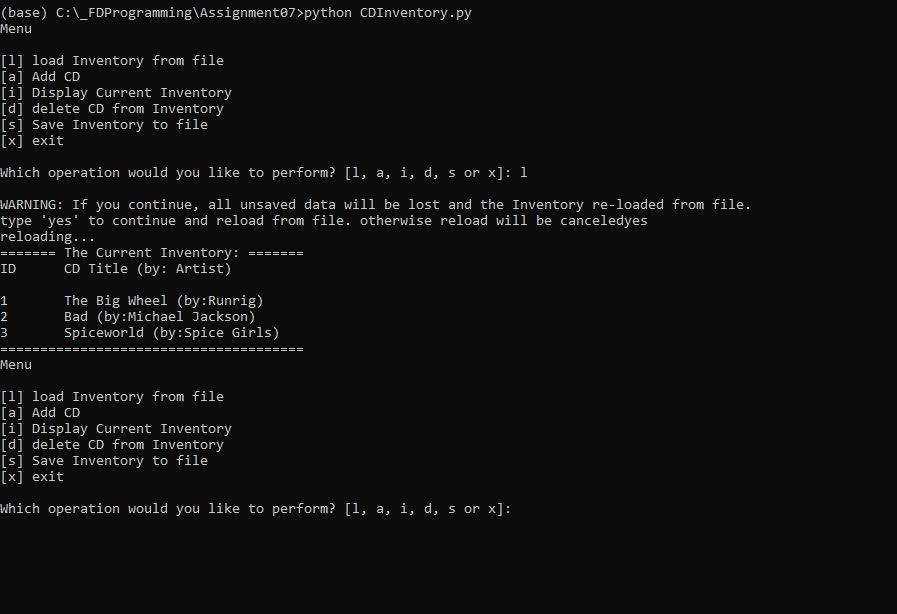


Figure 2 - Shows script being executed through Anaconda prompt.

# Summary

In this assignment we learned about structured error handling (exceptions) and pickling. I didn’t have many challenges with this assignment as converting to binary data was relatively easy, as was the pickling. One thing that I had challenges with was the getting the data file values to append to the *lstTbl*. In trying to source the issue, I found that I excluded appending the values to the actual *lstTbl* in *read\_file* function which was an easy fix once I found the culprit. Also, in the previous assignments it was indicated that I should move the print functions out of the processing functions, so I moved these to the actual main program, either using a return value of true or simply moved the show inventory functions to the main program.

1. <https://www.datacamp.com/community/tutorials/exception-handling-python> [↑](#footnote-ref-1)
2. <https://www.datacamp.com/community/tutorials/pickle-python-tutorial> [↑](#footnote-ref-2)