Jason Johanneck

2021-Nov-28

IT FDN Programming 110

FDN Programming 110 Assignment 07

# Introduction

In assignment seven we cloned the code from the last assignment and changed the data storage type in the output file from text to binary. Exception handling was added for user input whenever prompting for a CD ID.

# Writing the script:

The main issue I ran into when working on the script was how to read pickle data and determine when the end-of-file was reached. To do this, I used a While Loop to read the binary data from the CD Inventory file and then setup an exception to catch the end-of-file error.  
  
When testing the script, I used short cd titles and artists such as c4 and a4 to speed up testing.

Running the scripts.

Given the instructions to use binary data, my screen prints show the load data and save data operations. I also included screen prints to show the use of error handling when prompting for a CD ID during the Add CD and Delete CD user selection options.

Running the script from Spyder

### Loading CD Inventory into Memory:

My starting point was a file with three CD’s already in inventory:

Graphical user interface, text, application

Description automatically generated

Figure - Spyder - CD Inventory file pre-loaded with binary data.

Here I am displaying CD Inventory to show the data was loaded from binary and then converted for display purposes.  
  
Text

Description automatically generated

Figure -Spyder - Displaying CD data after initial load from file.

### Exception Handling when prompting for a CD ID to add

When prompting to add a CD ID, exception handling was added to confirm the ID was an integer.

Text

Description automatically generated

Figure - Spyder - Throwing an exception when adding a CD ID that is not an int.

### Exception Handling when prompting for a CD ID to delete

When prompting to delete a CD ID, exception handling was added to confirm the ID was an integer.

### Text Description automatically generated

Figure - Spyder- Throwing an exception when prompting for a CD ID to delete when it is not an integer.

### Saving CD Inventory from memory into file as binary data:

Here is the menu selection to save inventory after I have added a couple of new CDs to the in-memory table:

### A screenshot of a computer Description automatically generated with medium confidence

Figure - Spyder - Saving Inventory to Disk after adding two CD's to in-memory table.

Here are the contents of the CD Inventory file after saving the two new CD’s:

Graphical user interface, application

Description automatically generated

Figure - Spyder- Contents of Inventory File after Saving the two new CD's.

### 

Running the script from the Console:

### Loading CD Inventory into Memory:

My starting point was a file with CD data populated during interactive Spyder sessions.

Here I am displaying CD Inventory to show the data was loaded from binary and then converted for display purposes.  
  
Text

Description automatically generated

Figure - Console - Loading and Displaying Binary Data.

### Exception Handling when prompting for a CD ID to add or delete:

When prompting to add or delete a CD ID, exception handling was added to confirm the ID was an integer.

Text

Description automatically generated

Figure - Console - Exception Handling when prompting for a CD ID to add or Delete.

### Saving CD Inventory from memory into file as binary data:

Here I am adding a new CD and then saving my current inventory in memory to disk in binary format.

Text

Description automatically generated

Figure - Console - Saving CD Inventory from memory to Disk.

Here is the CD inventory file being displayed in NotePad++ after CD 10 was added:

# Graphical user interface, text, application, email Description automatically generated

Figure - Viewing CD Inventory file in NotePad++ to show data was saved in binary format.

# GitHub Repository Link:

<https://github.com/jjohanne514/Assignment07>

## Summary

For this assignment we learned the following:

* How to load and save data in binary format using the imported Pickle module.
* Basic use of try/except blocks.