Cheryl Montejo

August 14, 2022

Foundations of Programming, Python

Assignment 6

CD Inventory FUNctions

# Introduction

For this assignment, I will explain the steps I took to update an existing Python script for a CD inventory system to use a list of dictionaries, allow for the loading of data, and allow the user to delete rows in the data as well.

# Learning Through the Module

For this assignment, I skimmed the module prior to trying the assignment. For this assignment, I really played around the started script and completed the assignment through trial and error.

# Drafting the Script

For this assignment, we were asked to modify an existing script that was a solution to Assignment 5. When reviewing the start script provided, I first updated the header to indicate that I copied the script. I then started reading through the script to add spaces or indents to the comments to make it easier to read. I chose not to rearrange this one like I did in the previous assignment because the sections were correctly numbered. I also noted where the TODOs were listed so I would know what I needed to create as part of the assignment. From the comments, I determined I needed to create functions for saving data to a text file and for processing data which, includes adding and deleting CDs.

When I tried to run the starter script to make sure everything worked, the first problem I ran into was that there was no error handling around loading the inventory. If a text file didn’t exist to load the data from, the script would error and kick you out. To remedy this, I added a try-except construct, which I know we’re going to cover more in module 7 (StackOverflow, <https://stackoverflow.com/questions/574730/python-how-to-ignore-an-exception-and-proceed>) (external website)[[1]](#footnote-1). Another smaller item I remedied was the extra break in section of the script to delete a CD, which I removed.

For creating functions for parts of the script that were marked as TODO, what took the most thought was determining what the arguments and returns should be. I figured that part out by carefully reading through each section and also through trial and error.

# Running the Script in Spyder and Anaconda Prompt

For running the script in Spyder, I launched Spyder and opened CDInventory.py (File -> Open…->CDInventory.py). Through the menu, I tested out the options. I started by loading multiple CDs into the inventory and then saving it to a text file. I then deleted all the CDs from the script, and then reloaded the inventory from the text file before exiting. I ran the script again and loaded the inventory from the text file to make sure the inventory would load at the start. I then deleted the text file so I could start from scratch while testing the script in the Terminal console.

For running the script in Anaconda Prompt, I opened the Terminal console and navigated to the correct folder using the cd command. I then used the python command with the file name, CDInventory.py, to run the script. Using the menu, I repeated the steps from when I tested the script in Spyder. I did add in the additional step of entering something not in the menu when prompted to make sure the script would prompt me for another operation again.

Text

Description automatically generated

Figure 1 - Screenshot showing CDInventory.py in Spyder being used to view the inventory and add a CD

Text

Description automatically generated

Figure 2 - Screenshot showing CDInventory.py in Spyder being used to add a CD and save the inventory to a text file

Text

Description automatically generated

Figure 3 - Screenshot showing CDInventory.py in Spyder being used to save the inventory to a text file and then delete a CD

Text

Description automatically generated

Figure 4 - Screenshot showing CDInventory.py in Spyder being used to delete a CD and then load the data back in from the text file

Text

Description automatically generated

Figure 5 - Screenshot showing CDInventory.py in Spyder being used to load the data from the text file and then exit the script

Text

Description automatically generated

Figure 6 - Screenshot showing CDInventory.py in Spyder loading the data automatically from the text file and then exiting

Application

Description automatically generated with medium confidence

Figure 7 - Screenshot showing the data in the text file

Text

Description automatically generated

Figure 8 - Screenshot showing CDInventory.py in Spyder being used to display the current inventory and then delete a CD

Text

Description automatically generated

Figure 9 - Screenshot showing CDInventory.py in Spyder being used to delete a CD and then save the inventory to a text file

Graphical user interface, application

Description automatically generated

Figure 10 - Screenshot showing the data in the text file after deleting a CD and saving it to the text file

Text

Description automatically generated

Figure 11 - Screenshot showing CDInventory.py in the terminal being used to display the empty inventory and then add a CD

Text

Description automatically generated

Figure 12 - Screenshot showing CDInventory.py in the terminal being used to add a CD and then save the data to a text file

Text

Description automatically generated

Figure 13 - Screenshot showing CDInventory.py in the terminal being used to delete a CD and then save the data to a text file

Text

Description automatically generated

Figure 14 - Screenshot showing CDInventory.py in the terminal being used to show what happens if you enter an item that isn't on the menu and then exiting the script

# Summary

In this assignment, we explored using functions and how that impacted the existing script we had from assignment 5. For this script, I played around with using try-except for error handling even though I know it will be covered in the next module. I’ve been looking forward to learning more about error handling and data validation since we started this class because I recognize its importance in creating functional and usable code.

# Appendix

## Link to GitHub Repo

## Listing AddressBook.py (pasted from Notepad++)

#------------------------------------------#

# Title: CDInventory.py

# Desc: Working with classes and functions.

# Change Log: (Who, When, What)

# DBiesinger, 2030-Jan-01, Created File

# CMontejo, 2022-Aug-13, Copied File

# CMontejo, 2022-Aug-14, Cleaned up formatting and comments, moved TODOs to functions, added error handling to read\_file function

#------------------------------------------#

# -- DATA -- #

strChoice **=** '' # User input

lstTbl **=** **[]** # list of lists to hold data

dicRow **=** **{}** # list of data row

strFileName **=** 'CDInventory.txt' # data storage file

objFile **=** **None** # file object

# -- PROCESSING -- #

**class** **DataProcessor:**

"""Processing the data within the script"""

*@staticmethod*

**def** add\_cd**(**lstTbl**,** strID**,** strTitle**,** strArtist**):**

"""Adds new CD to the table

Args:

lstTbl (list of lists): 2D table to hold CD Inventory data

strID (string): ID number for the CD

strTitle (string): Title of the CD

strArtist (string): Artist for the CD

Returns:

None.

"""

intID **=** **int(float(**strID**))**

dicRow **=** **{**'ID'**:** intID**,** 'Title'**:** strTitle**,** 'Artist'**:** strArtist**}**

lstTbl**.**append**(**dicRow**)**

IO**.**show\_inventory**(**lstTbl**)**

*@staticmethod*

**def** del\_cd**(**lstTbl**):**

"""Deletes a CD from the table

Args:

lstTbl (list of lists): 2D table to hold CD Inventory data

Returns:

lstTbl (list of lists): 2D table to hold CD Inventory data

"""

intRowNr **=** **-**1

blnCDRemoved **=** **False**

**for** row **in** lstTbl**:**

intRowNr **+=** 1

**if** row**[**'ID'**]** **==** intIDDel**:**

**del** lstTbl**[**intRowNr**]**

blnCDRemoved **=** **True**

**if** blnCDRemoved**:**

**print(**'The CD was removed\n'**)**

**return** lstTbl

**else:**

**print(**'Could not find this CD!\n'**)**

**class** **FileProcessor:**

"""Processing the data to and from text file"""

*@staticmethod*

**def** read\_file**(**file\_name**,** table**):**

"""Function to manage data ingestion from file to a list of dictionaries

Reads the data from file identified by file\_name into a 2D table

(list of dicts) table one line in the file represents one dictionary row in table.

Args:

file\_name (string): name of file used to read the data from

table (list of dict): 2D data structure (list of dicts) that holds the data during runtime

Returns:

None.

"""

**try:**

table**.**clear**()** # this clears existing data and allows to load data from file

objFile **=** **open(**file\_name**,** 'r'**)**

**for** line **in** objFile**:**

data **=** line**.**strip**().**split**(**','**)**

dicRow **=** **{**'ID'**:** **int(**data**[**0**]),** 'Title'**:** data**[**1**],** 'Artist'**:** data**[**2**]}**

table**.**append**(**dicRow**)**

objFile**.**close**()**

**except:**

**pass**

*@staticmethod*

**def** write\_file**(**file\_name**,** table**):**

"""Function to manage data ingestion from a list of dictionaries to a file

Reads the data from a 2D table (list of dictionaries) identified as lstTbl into a file.

Args:

file\_name (string): name of file used to read the data from

table (list of dict): 2D data structure (list of dicts) that holds the data during runtime

Returns:

None.

"""

objFile **=** **open(**strFileName**,** 'w'**)**

**for** row **in** lstTbl**:**

lstValues **=** **list(**row**.**values**())**

lstValues**[**0**]** **=** **str(**lstValues**[**0**])**

objFile**.**write**(**','**.**join**(**lstValues**)** **+** '\n'**)**

objFile**.**close**()**

# -- PRESENTATION (Input/Output) -- #

**class** **IO:**

"""Handling Input / Output"""

*@staticmethod*

**def** print\_menu**():**

"""Displays a menu of choices to the user

Args:

None.

Returns:

None.

"""

**print(**'Menu\n\n[l] Load Inventory from File\n[a] Add CD\n[i] Display Current Inventory'**)**

**print(**'[d] Delete CD from Inventory\n[s] Save Inventory to File\n[x] Exit\n'**)**

*@staticmethod*

**def** menu\_choice**():**

"""Gets user input for menu selection

Args:

None.

Returns:

choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x

"""

choice **=** ' '

**while** choice **not** **in** **[**'l'**,** 'a'**,** 'i'**,** 'd'**,** 's'**,** 'x'**]:**

choice **=** **input(**'Which operation would you like to perform? [l, a, i, d, s or x]: '**).**lower**().**strip**()**

**print()** # Add extra space for layout

**return** choice

*@staticmethod*

**def** show\_inventory**(**table**):**

"""Displays current inventory table

Args:

table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.

Returns:

None.

"""

**print()** # Add extra space for layout

**print(**'======= The Current Inventory: ======='**)**

**print(**'ID\tCD Title (by: Artist)\n'**)**

**for** row **in** table**:**

**print(**'{}\t{} (by:{})'**.format(\***row**.**values**()))**

**print(**'======================================'**)**

**print()** # Add extra space for layout

*@staticmethod*

**def** new\_cd**():**

"""Get user input for a new CD

Args:

None.

Returns:

None.

"""

strID **=** **input(**'Enter ID: '**).**strip**()**

strTitle **=** **input(**'What is the CD\'s title? '**).**strip**()**

strArtist **=** **input(**'What is the Artist\'s name? '**).**strip**()**

DataProcessor**.**add\_cd**(**lstTbl**,** strID**,** strTitle**,** strArtist**)**

# -- INTERFACE -- #

# 1. When program starts, read in the currently saved Inventory

FileProcessor**.**read\_file**(**strFileName**,** lstTbl**)**

# 2. Start main loop

**while** **True:**

# 2.1 Display Menu to user and get choice

IO**.**print\_menu**()**

strChoice **=** IO**.**menu\_choice**()**

# 3. Process menu selection

# 3.1 Process exit first

**if** strChoice **==** 'x'**:**

**break**

# 3.2 Process to load inventory

**if** strChoice **==** 'l'**:**

**print(**'WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.'**)**

strYesNo **=** **input(**'Type \'yes\' to continue and reload from file. Otherwise reload will be canceled\n'**)**

**if** strYesNo**.**lower**()** **==** 'yes'**:**

**print(**'reloading...'**)**

FileProcessor**.**read\_file**(**strFileName**,** lstTbl**)**

IO**.**show\_inventory**(**lstTbl**)**

**else:**

**input(**'canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.'**)**

IO**.**show\_inventory**(**lstTbl**)**

**continue** # Start loop back at top

# 3.3 Process to add a CD

**elif** strChoice **==** 'a'**:**

# 3.3.1 Ask user for new ID, CD Title and Artist and add item to the table (Was a TODO)

IO**.**new\_cd**()**

**continue** # Start loop back at top

# 3.4 Process to display current inventory

**elif** strChoice **==** 'i'**:**

IO**.**show\_inventory**(**lstTbl**)**

**continue** # Start loop back at top

# 3.5 Process to delete a CD

**elif** strChoice **==** 'd'**:**

# 3.5.1 get Userinput for which CD to delete

# 3.5.1.1 Display Inventory to user

IO**.**show\_inventory**(**lstTbl**)**

# 3.5.1.2 Ask user which ID to remove

intIDDel **=** **int(input(**'Which ID would you like to delete? '**).**strip**())**

# 3.5.2 Search thru table and delete CD (Was a TODO)

DataProcessor**.**del\_cd**(**lstTbl**)**

IO**.**show\_inventory**(**lstTbl**)**

**continue** # Start loop back at top

# 3.6 Process to save inventory to file

**elif** strChoice **==** 's'**:**

# 3.6.1 Display current inventory and ask user for confirmation to save

IO**.**show\_inventory**(**lstTbl**)**

strYesNo **=** **input(**'Save this inventory to file? [y/n] '**).**strip**().**lower**()**

# 3.6.2 Process choice

**if** strYesNo **==** 'y'**:**

# 3.6.2.1 Save data (Was a TODO)

FileProcessor**.**write\_file**(**strFileName**,** lstTbl**)**

**pass**

**else:**

**input(**'The inventory was NOT saved to file. Press [ENTER] to return to the menu.'**)**

**continue** # Start loop back at top

# 3.7 Catch-all should not be possible, as user choice gets vetted in I/O, but to be safe:

**else:**

**print(**'General Error'**)**

1. Retrieved 2022-Aug-14 [↑](#footnote-ref-1)