Kenneth Chiu

February 27, 2022

Foundations of Programming: Python

Assignment 05

The Magic CD Inventory Program

# Introduction

In this module, we learned more about accessing files and loading data with better formatting by using functions such as strip() and split(). Additionally, we learned about Dictionary and Separation of Concerns which are used to transform the CD Inventory program into the Magic CD Inventory Program.

# What are Dictionaries?

Unlike Sequences, Dictionaries store data in key: value pairs format inside of braces {} operator. To access data in a Dictionary, instead of using numeric indexes, data is accessed using programmer defined keys. Similar to List, multiple Dictionaries can be combined into Lists to create 2D tables such as the CD Inventory table we have been working with.

# Separation of Concerns

The Separation of Concerns (“SoC”) is a programming pattern which is a principle adopted in programming. The SoC divides programs into Data section, where variables and constants are declared, Processing section, where specific tasks and operations are performed on the defined data and lastly the Presentation sections, where data is inputted from users or data is presented to the users. The SoC organize codes in these three sections to provide better readability and maintenance as the programs become much bigger.

# The Magic CD Inventory program

To complete the Magic CD Inventory program, I loaded the CDInventory\_Starter.py file and updated the script header – List 1.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | *#------------------------------------------#*  *# Title: CDInventory.py*  *# Desc: Starter Script for Assignment 05*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *# KChiu, 2022-Feb-27, Add codes to complete the CD Inventory program*  *#------------------------------------------#* |

List - Script Header

Next, I updated the lstTbl from a List to a Dictionary per assignment instructions as you can see on Line 14 in List 2.

|  |  |
| --- | --- |
| 9  10  11  12  13  14  15  16  17 | *# Declare variabls*  strChoice = '' *# User input*  lstTbl = [] *# list of lists to hold data*  *# TODone replace list of lists with list of dicts*  dicRow = {} *# dictionary of data row*  strFileName = 'CDInventory.txt' *# data storage file*  objFile = **None** *# file object* |

List - Declaring Variables

Next, I added the functionality of loading data from the CDInventory.txt file in read-only mode. To ensure the data is in a clean and appropriate format for further processing, the strip() and split() functions were used to remove the extra blank lines and chain all the values in each Dictionary with commas. Finally, by using a for-Loop, the CD inventory data is added to the lstTbl table row by row and the CDInventory.txt file is closed.

|  |  |
| --- | --- |
| 30  31  32  33  34  35  36  37 | **if** strChoice == 'l': *# no elif necessary, as this code is only reached if strChoice is not 'exit'*  *# TODone Add the functionality of loading existing data*  objFile = open(strFileName, 'r')  **for** row **in** objFile:  lstRow = row.strip().split(',')  dicRow = {'ID': int(lstRow[0]), 'CD Title': lstRow[1], 'Artist': lstRow[2] }  lstTbl.append(dicRow)  objFile.close() |

List - Loading Existing Inventory

To accommodate the Dictionary inner data structure, several parts of the existing codes need to be updated.

In the Add CD function, a new dicRow row is now combing the ID, CD Title and Artist Name into a new Dictionary instead of a List as you can see on Row 44 of the codes – List 4.

|  |  |
| --- | --- |
| 38  39  40  41  42  43  44  45 | **elif** strChoice == 'a':  *# 2. Add data to the table (2d-list) each time the user wants to add data*  strID = input('Enter an ID: ')  strTitle = input('Enter the CD**\'**s Title: ')  strArtist = input('Enter the Artist**\'**s Name: ')  intID = int(strID)  dicRow = {'ID': intID, 'CD Title': strTitle, 'Artist': strArtist}  lstTbl.append(dicRow) |

List - Add CD

In the Display Current Inventory Data function, the \* operator can no longer be used to unpack each row of CD data since it’s now in a Dictionary format; thus, I’ve updated the code to print out each of the ID, CD Title and Artist using the pre-defined Dictionary Keys on Row 50 of the codes – List 5.

|  |  |
| --- | --- |
| 46  47  48  49  50 | **elif** strChoice == 'i':  *# 3. Display the current data to the user each time the user wants to display the data*  print('ID, CD Title, Artist')  **for** row **in** lstTbl:  print(row['ID'], row['CD Title'], row['Artist'], sep = ', ') |

List - Display Current Inventory Data

In the Save Inventory to File function, in order to save the values in the Dictionaries to the CDInventory.txt file, the values() function is added to Row 78 of the code; otherwise, the Dictionary Keys would be saved instead – List 6.

|  |  |
| --- | --- |
| 73  74  75  76  77  78  79  80  81  82 | **elif** strChoice == 's':  *# 4. Save the data to a text file CDInventory.txt if the user chooses so*  objFile = open(strFileName, 'a')  **for** row **in** lstTbl:  strRow = ''  **for** item **in** row.values():  strRow += str(item) + ','  strRow = strRow[:-1] + '**\n**'  objFile.write(strRow)  objFile.close() |

List - Save Inventory to File

Last but not least, I tackled the most challenging part of this assignment, which is adding a functionality to remove a CD entry from the lstTbl table. The function starts with compiling all CD IDs into a list so we can check if the ID users chooses exists in the CD inventory. Then the program asks user to provide the ID for the CD entry to be deleted. Next, we check If the ID exist in the CD ID list and if it doesn’t, an error message is raised then program returns to the main menu. If the provide CD ID exists, a for-Loop is initiated to loop through all the Dictionaries in the lstTbl table and compare with each CD ID and if the CD ID isn’t the same as the user input, then the Loop stops and moves to the next Dictionary but if the CD ID is the same as the user input, the entire Dictionary row is deleted from the lstTbl table – List 7. I added a try/pass-loop in case the program behaves unexpectedly so the program will return to the main menu and continues.

|  |  |
| --- | --- |
| 51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 | **elif** strChoice == 'd':  *# TODone Add functionality of deleting an entry*  *# Creeate a list of all the CD IDs.*  id\_lst = []  **for** row **in** range(len(lstTbl)):  row\_id = lstTbl[row]['ID']  id\_lst.append(row\_id)  *# Ask user which ID to be deleted based on ID number*  del\_key = int(input('Which CD do you want to delete? **\**  **\n**Please enter a CD ID: '))  **if** del\_key **not** **in** id\_lst:  *# Raise an error if CD ID doesn't exist.*  print('Boooo! CD doesn**\'**t exist!')  **pass**  **for** row **in** range(len(lstTbl)):  **try**:  **if** lstTbl[row]['ID'] != del\_key:  **pass**  **else**:  **del** lstTbl[row]  **except**:  **pass** |

List - Delete an Entry

# Summary

This assignment has been the most challenging and time consuming so far. Particularly, adding a function to delete a CD entry took me countless experiments and corrections to finally get it to work. I started by having the codes compare the user input ID and the List index of each Dictionary to determine if an entry should be deleted; however, I realized that the indexes would change if an entry with lower index is deleted first. For example, if CD 1 has been deleted, then CD 2’s index would change and become the 1st row with index 0 and if the user inputs CD ID 2 for deletion, the program will error out since CD 2’s index has now changed. After many frustrations screaming and a few hair-pulling, I was able to figure it out and have the codes compare the actual CD ID value of each Dictionary with the user input and delete the entire Dictionary if there’s a match.

Everything below is required screenshots per assignment instructions.

Below are screenshots showing the CD Inventory program working on Spyder – Figure 1 through 6.

Text

Description automatically generated

Figure - Load Inventory from File

Text

Description automatically generated

Figure - Display Current Inventory

Text

Description automatically generated

Figure - Add CD

Text

Description automatically generated

Figure - Display Inventory After CD Addition

Text

Description automatically generated

Figure - Delete CD 1

Text

Description automatically generated

Figure - Current CD Inventory after CD 1 Deletion

Text

Description automatically generated

Figure - Delete CD 2

Text

Description automatically generated

Figure - Current CD Inventory after CD 2 Deletion

Text

Description automatically generated

Figure - Save Inventory to File (Only CD 3 remains)

Text

Description automatically generated

Figure - CDInventory File after CD 3 Addition Saved

Text

Description automatically generated

Figure - Exit Magic CD Inventory Program

Below screenshots show Magic CD Inventory program run in the Terminal.

Text

Description automatically generated

Figure - Magic CD Inventory in Terminal Part 1

Text

Description automatically generated

Figure - Magic CD Inventory in Terminal Part 2

Text

Description automatically generated

Figure - Magic CD Inventory in Terminal Part 3

Text

Description automatically generated

Figure - Magic CD Inventory in Terminal Part 4

CD Inventory file after Magic Inventory Program runs in both Spyder and Terminal. Note that in Spyder run, I added CD 3 and in Terminal, I added CD 4.

Text, application

Description automatically generated