## Katrina Hauser

## 07 August 2021

## BIDD Fundamentals of Programming (Python)

## Assignment 05

# Assignment 05: Lists, Dictionaries, Separation of Concerns, GitHub.

## Introduction

In assignment 05, the focus was on exploring lists, dictionaries, separation of concerns, error handling, functions, script templates and GitHub. To understand this functionality, several videos and references were reviewed.

## Topic 1: Python Features Review

The following Python modules were reviewed in addition to reading the second chapter of the course textbook: <https://saravji.github.io/saravjis_hut/FDN_Prog/Modules.html> - Module 5 only.

The following websites were reviewed and videos watched:

<https://realpython.com/python-lists-tuples/>

<https://realpython.com/python-dicts/>

<https://www.geeksforgeeks.org/file-handling-python/>

1. The goal of this module is to understand several questions:
2. How do you read data from a file into a list?
3. How do you read data from a file into a dictionary?
4. Why is it making sense to organize data in a 2-dimensional way?
5. What is the programming pattern “Separation of Concerns”?
6. How would you use a function to organize your code?
7. Why is a script template useful?
8. Why is error handling (try-except) useful?
9. What is GitHub and why is it used?
10. What is GitHub’s mascot?

## Topic 2: Python Script Development

To demonstrate knowledge learned from the modules and references above, a Python script to modify the previous version of our CD Inventory program that uses a list of dictionaries, adds the functionality of loading existing data, deleting an entry and saving data to a CDInventory.txt data file was developed.

From the previous script, a menu is presented to the user to selection options to control the CD inventory. The script loops through the options and executes the commands selected by the user. If the user enters an incorrect value, the script will catch this and present an error message and ask the user to enter a correct value. The new script adds functionality to load existing data, delete entries and presents the menu options as letters.

The script is arranged into two main sections – (1) presenting the menu and (2) processing the selection. To start, variables, lists and dictionaries are initialized. The file name is also assigned to a variable. The loop control is also changed. At the end of this section, the script still presents a menu to the user and the user is asked to choose an option. A WHILE loop is employed to process the user’s choice and allow selecting from the menu multiple times. The WHILE loop continues presenting the menu until it is no longer TRUE or the users chooses to exit (‘x’) the program:

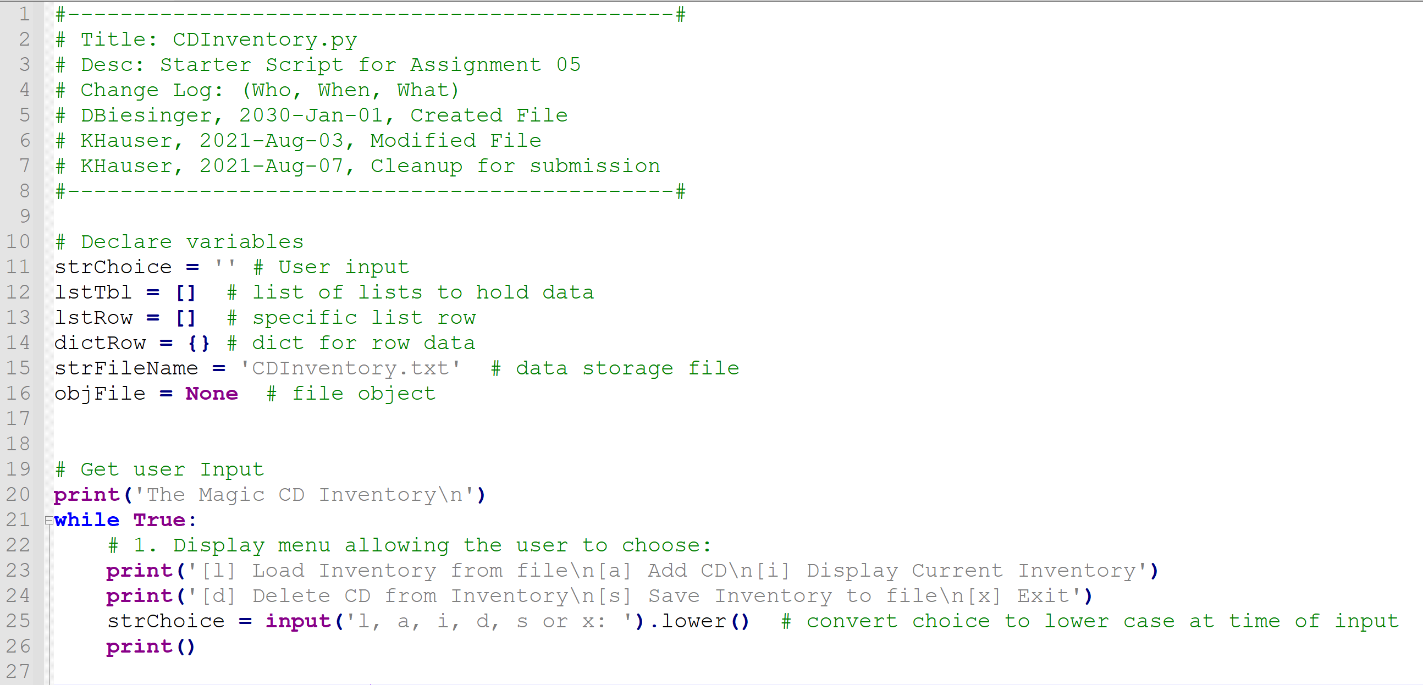


Figure 1: Script CDInventory.py asking the user to selection from a menu.

In the processing section, IF statement logic is still used to evaluate the user’s menu selection control the scripts behavior. Structured similar to a CASE statement in other languages, the IF statement routes the script depending on the option selected by the user. The first two user options in this section is exit the script or load data from the file. When the user selects ‘x’ from the menu, the script will terminate.

If the user selects ‘l’, the script load data from the ‘CDInventory.txt’ file. The file is opened in read mode and the script loops through each comma separated line. The script will read each row in the file, split the values from the row and remove the comma separators. The elements in each row will be added as values assigned to specific keys in the dictRow dictionary. The dictionary is then appended to the lstTbl list. This effectively reads the raw data and organizes it into a list of dictionaries. Once EOF is reached, the user is presented with a load complete message and the file is closed.

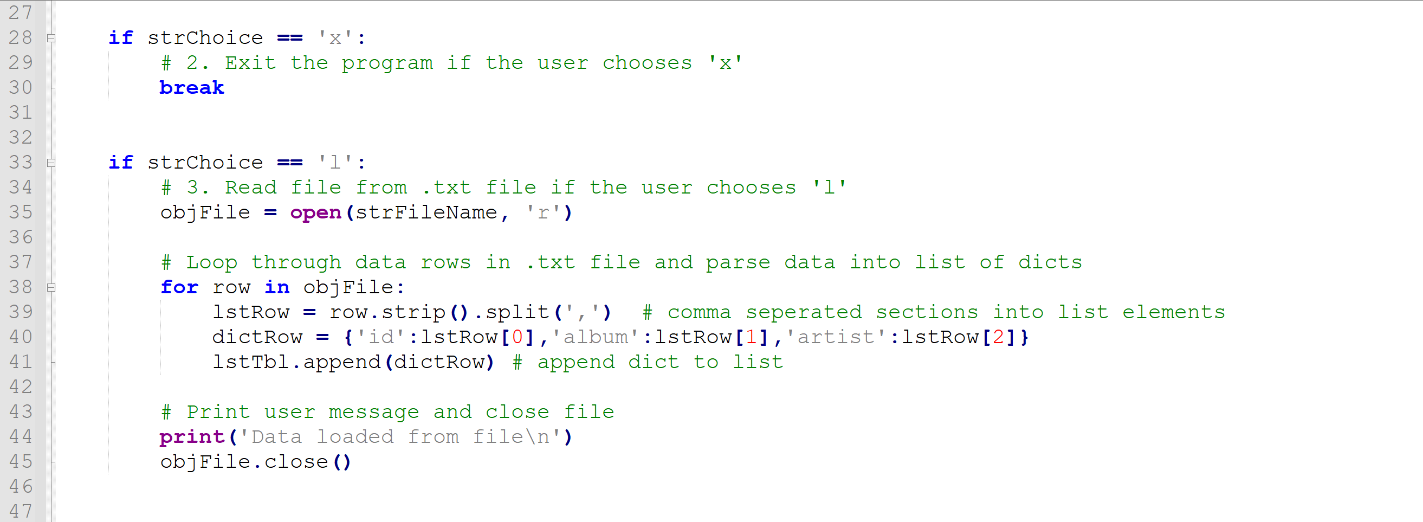


Figure 2: Script CDInventory.py processing details exit (‘x’) and load (‘l’) commands. Data is read in a list of dicts.

When the user selects ‘a’ from the menu, the user is able to add a new entry. The script will ask and capture ID, Title and Artist details. The data is capture in the dictRow dictionary and appended to the lstTbl list to append the new dict to the list of dicts.

When the user selects ‘i’ from the menu, the script will list data that has been loaded or appended. The script will print a header first. A loop is then used to loop through the list of dicts to print the values by key and present as ID, Album and Artist to the user.

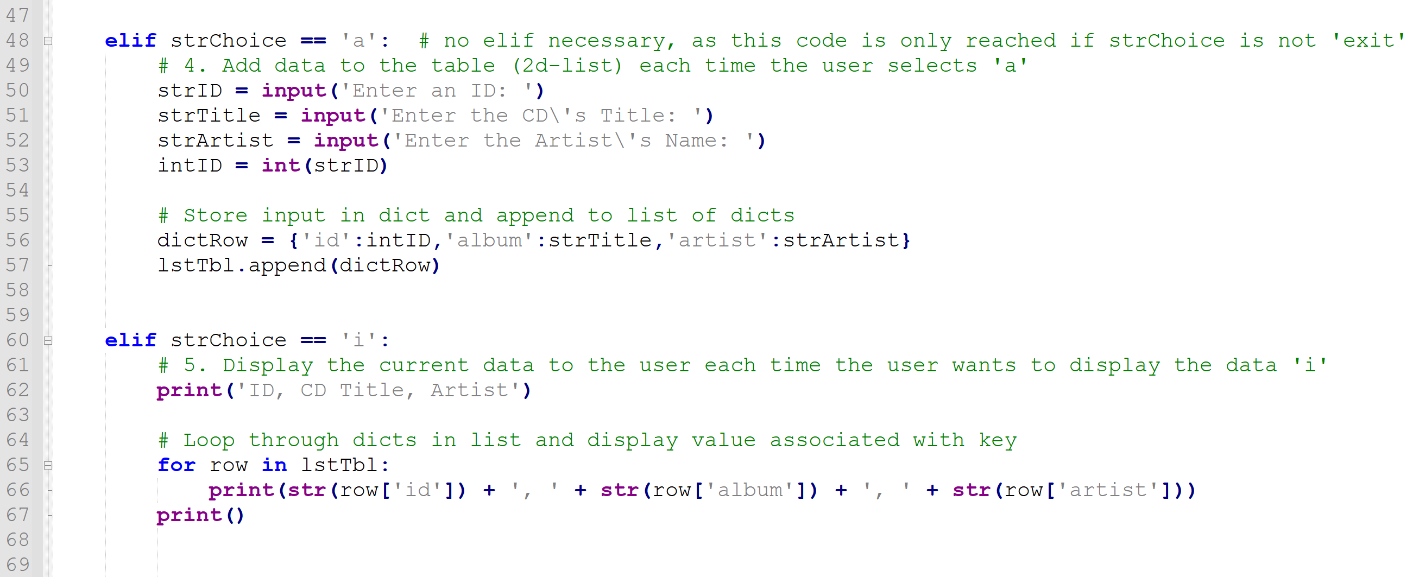


Figure 3: Script CDInventory.py process details Add (‘x’) and Display (‘I’) commands. The user can add data and display current inventory. The display appears as a comma separated rows of data.

When the users selects ‘d’ from the menu, new functionality allows the user to delete rows of data by ID. To process this request, the script asked the user which ID to delete and loops through the list of dicts to find the row for deletion. The loop first determines the number if rows (dicts) in the list and then evaluates ID in each dict. If the user entered value equals the ID key value, the dicts is deleted from the list and the loop exits (“break”).

When the user selects ‘s’ from the menu, the data in the current list of dicts is written to the file. The file is open in write mode (‘w’) which will overwrite the existing file data with the new data. The script starts by opening the ‘CDInventory.txt’ file in write mode. A loop is used to determine the number of rows (dicts) in the list. The loop process each row (dict) and saves the data as a comma separated row of data. Once each dict has been processed and written, a message is displayed to the user and the file is closed.

The final section of the primary loop is the Else statement. This is used for error handling. If the user selects any value that is not expected, it is caught here. The user is presented with a message and returned to the menu for selection an appropriate menu option.

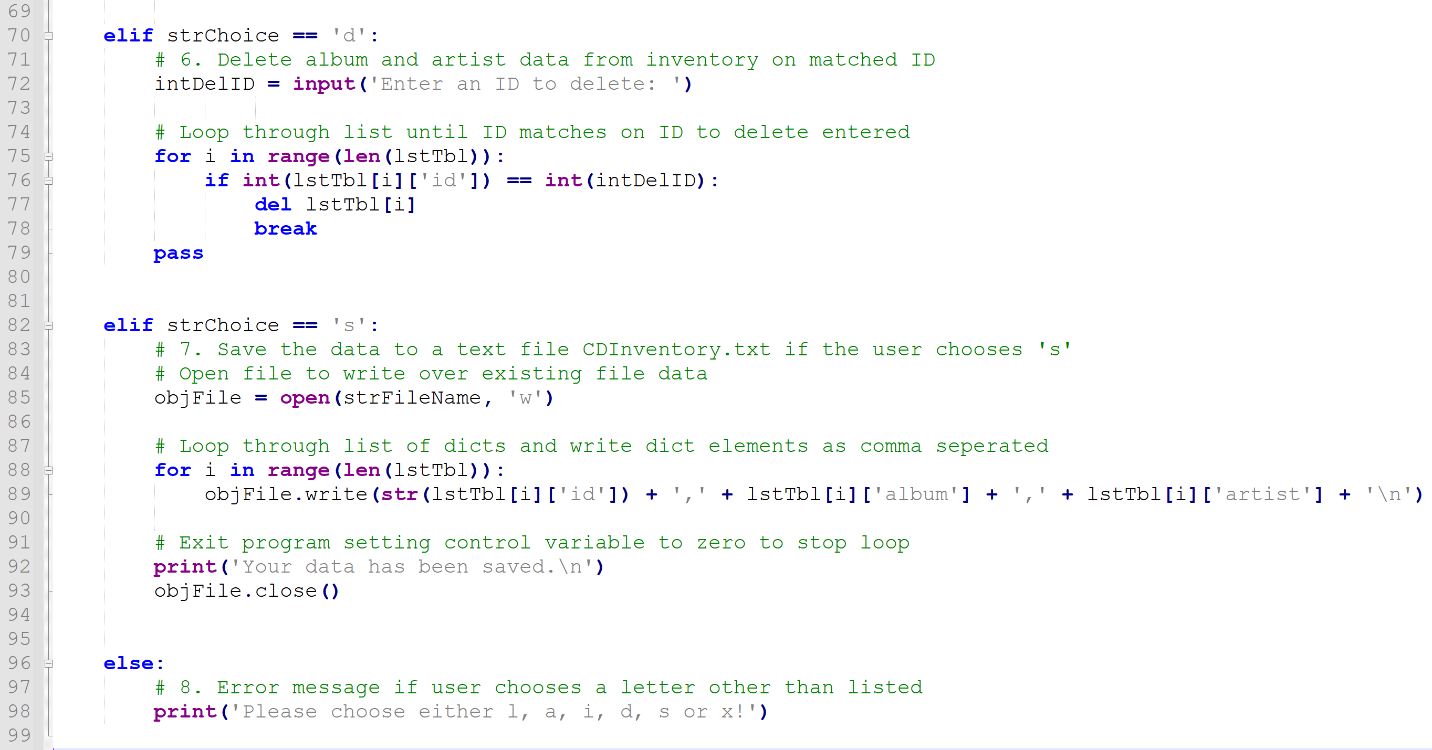


Figure 4: Script CDInventory.py processing Delete (‘d’) and Save (‘s’) options. Deletion removing rows by ID and saving as comma separated format. Error handling using Else at the end of the loop..

## Summary

In assignment 05, dictionaries, separation of concerns, error handling, functions, script templates and GitHub were reviewed. To demonstrate knowledge of the basics from this assignment, the CDInventory Python script from Assignment 4 was modified to incorporate load, deletion and error catching functionality. In addition, the internal script data structure as changed to employ a list of dicts. The script developed now allows users to load data from a file, add data, delete data, view data and save the data back to a file.

## Appendix

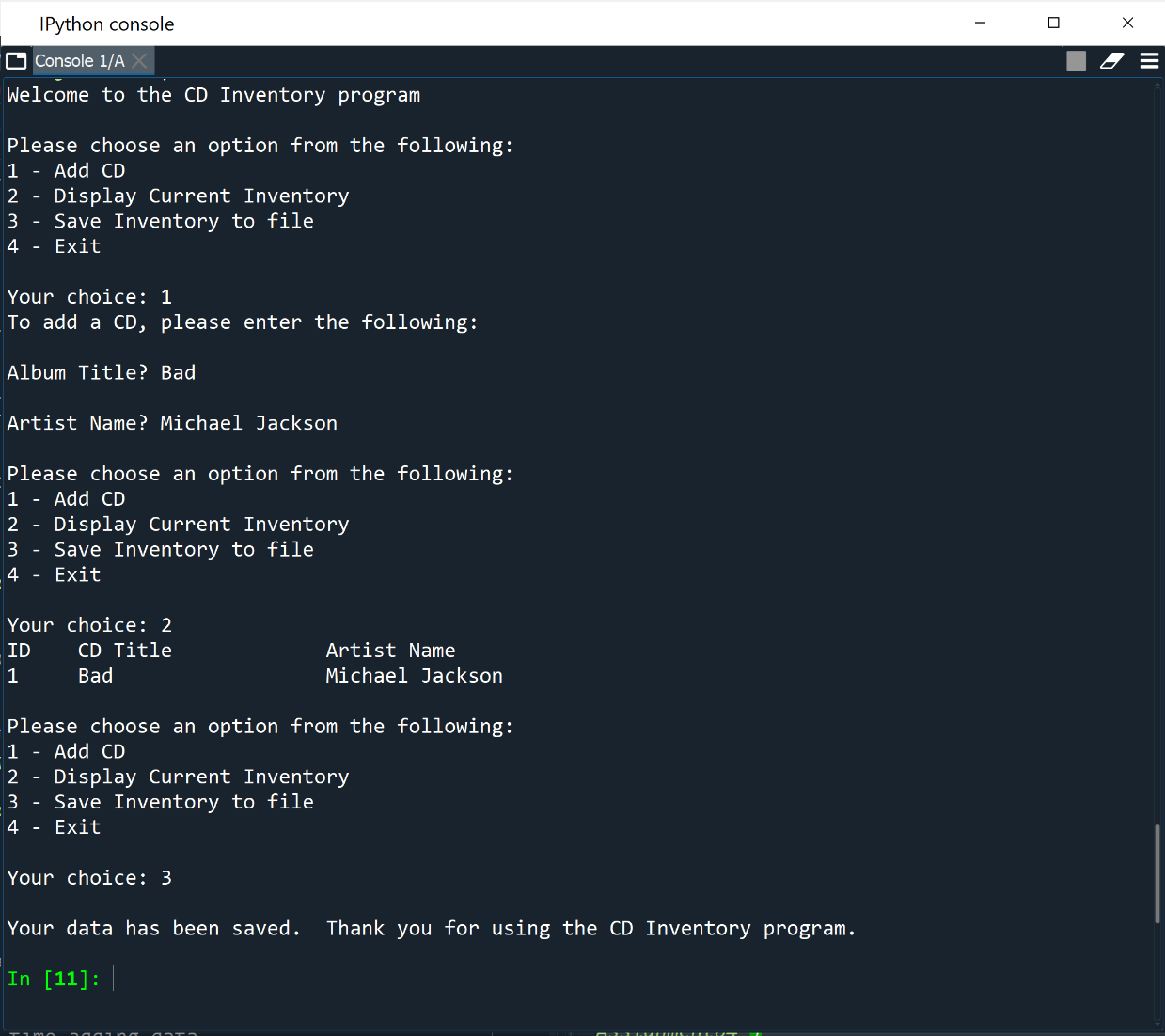


Figure 5: GitHub repository of Assignment05