Jurg Huerlimann

Aug 12. 2020

IT FDN 110 B SU 20: Foundations of Programming: Python

Assignment Module 05

Lists, Dictionaries, Error Handling, GitHub

# Introduction

In the fifth module and assignment of this course, we expanded our knowledge of lists and got introduced to dictionaries. We’ll also covered the basics about Separation of Concerns (SoC) programming pattern, error handling, functions, how to configure script templates in Spyder and how to use GitHub.

Questions we had to investigate and answer:

## What is the difference between a Dictionary and a List?

Lists are just like the arrays, declared in other languages. A list may contain DataTypes like Integers, Strings, as well as Objects. Lists are mutable, and hence, they can be altered even after their creation.  
  
Dictionaries on the other hand is an unordered collection of data values and consist of a key:value pair. Each key-value pair in a Dictionary is separated by a colon : , whereas each key is separated by a ‘comma’. The key in a Dictionary is immutable and cannot be changed.[[1]](#endnote-1)

## What is the difference between an index and a key?

In a sequence, the stored data is accessible via an index (subscript).

Dictionaries replace the index with key: value pairs. This means instead using an index to point to data in a sequence, a key is used under which the values are being stored and accessed.

## How do you read data from a file into a list?

See the following code snippet that shows how data is being read from a file row by row into in-memory, using lists and a loop:

1. # code to read the file line by line into in-memory using list
2. lstTbl.clear()
3. objFile = open(strFileName, 'r')
4. **for** row **in** objFile:
5. lstRow = row.strip().split(',')
6. lstTbl.append(lstRow)
7. objFile.close()

Listing 1 Reading data from a file into in-memory using list

## How do you read data from a file into a dictionary?

See the following code snippet that shows how data is being read from a file row by row into in-memory, using a dictionary and a loop:

1. # code to read the file line by line into in-memory disctionary
2. lstTbl.clear()
3. objFile = open(strFileName, 'r')
4. **for** row **in** objFile:
5. lstRow = row.strip().split(',')
6. dicRow = {'artist': lstRow[0], 'title': lstRow[1]}
7. lstTbl.append(dicRow)
8. objFile.close()

Listing 2 Reading data from a file into in-memory using dictionary

## Why is it making sense to organize data in a 2-dimensional way?

Organizing data in a 2-dimensional way allows the data to be organized in rows and columns, very much like a matrix. An index or a key allows to point to a particular row, in which then the column can be specified too.

## What is the programming pattern “Separation of Concerns”?

Separation of Concern is the Best Practice of organizing the code into distinct sections. This makes it better organized and more readable. Most programs can be divided into three distinct sections: Data, Processing and Presentation (or Input-Output).

* Data: The Data section contains the definition of variables and constants.
* Processing: The Processing section contains the tasks being performed with the data.
* Presentation: The Presentation section contains the functionality for getting user input or displaying the resulting output.

Using functions allows to implement this concept much easier than our current continuous flow of code.

How would you use a function to organize your code?  
Functions are reusable pieces of code. A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide modularity for an application and a high degree of reusing code.

Why is a script template useful?  
Saving a script template in Spyder is useful so that one doesn’t have to type the header information from scratch every time one starts a new script. The template contains the default information and then just needs to be updated with the applicable relevant information.

Why is error handling (try-except) useful?  
Structured error handling is useful to catch and prevent certain errors that otherwise would cause the program to crash. (Example: division by 0). I can anticipate such an error, I can add code to my script that handles the error before it leads to a crash. An error message or warning can be displayed instead, making the user aware of the potential issue.

## What is GitHub and why is it used?

GitHub is a company that provides hosting for software development and version control using Git, but GitHub has now become synonymous for the actual code repository it hosts.

It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.[[2]](#endnote-2)

## What is GitHub’s mascot?

GitHub’s mascot is the Octocat. Its character is part cat, part octopus.



Figure 1 Octocat Logo[[3]](#endnote-3)

# Assignment05: Enhance Python Script to manage CD data

In this fifth assignment we were tasked to create a version of a CD Inventory program that uses dictionaries as the data repository. In addition to the previous features of allowing the user to enter CD data, view the current inventory, saving data to a CDInventory.txt data file and exit the program, we also had to add the feature to delete entries from in-memory.

Pseudo Code:  
Display a menu allowing the user to choose from:

[l] load Inventory from file

[a] Add CD

[i] Display Current Inventory

[d] delete CD from Inventory

[s] Save Inventory to file

[x] exit

For [a] create a loop to add data to the list each time the user wants to add data.

Ask for:   
 Enter an ID:

Enter the CD's Title:

Enter the Artist's Name:

Once everything is entered, display the main menu again.

For [i] create a loop to read from the current data in-memory and display it to the user.

For [d] display question which line of the inventory to delete.

Delete the defined entry in the corresponding table space.

For [s] Save the data to a text file CDInventory.txt if the user chooses so.

Create a loop that reads the in-memory information and writes it to the file, overwriting the existing data, so that the file does not get duplicate data written from memory.

For [x] Exit the program if the user chooses so.

Test for x from user input. If true, end the program. If not true, return to main menu.

## Resulting script:

The listing of the resulting script can be found in the Appendix. It is the result of doing the lab exercises in Module 05, reading up on these commands and methods, lots of experimenting and last but not least, feedback from the class forum as well as the usual expert advice from Dirk and Doug.

## Testing script in Spyder:

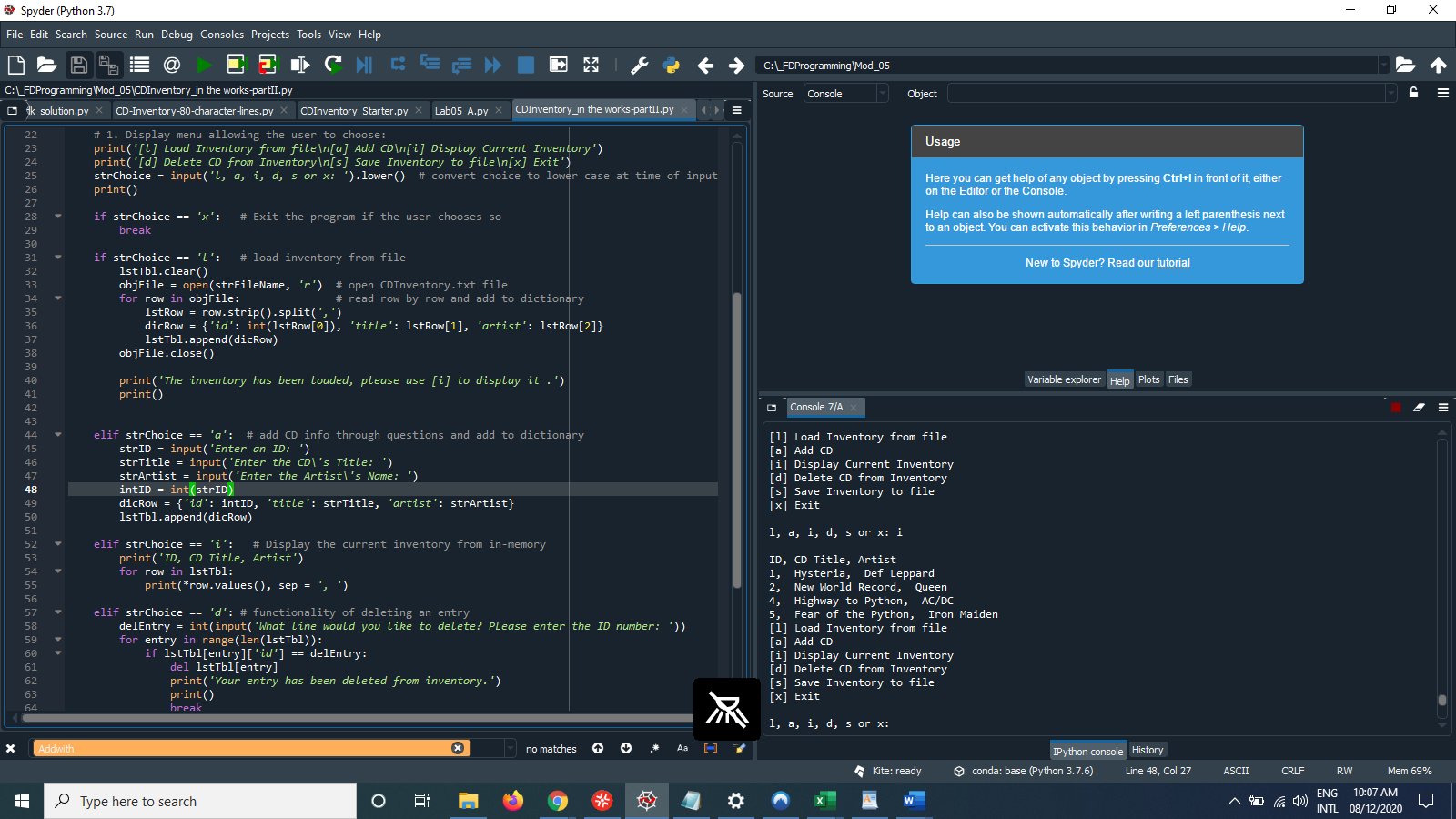


Figure Script running in Spyder

## Testing script in Terminal:

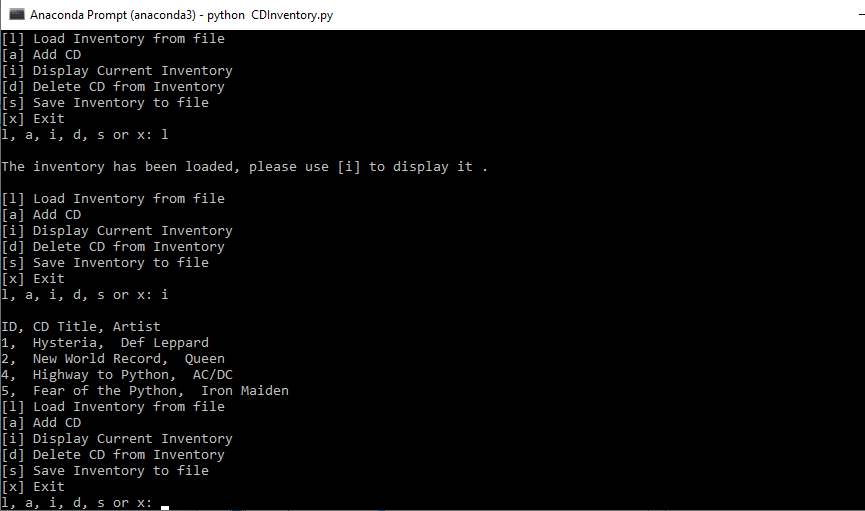


Figure Script running in Terminal

# Summary

This fifth assignment was based on all the knowledge we had learned so far, plus new features like using dictionaries and using the GitHub repository to post the assignment.

The challenge started right from scratch, when the starter script mentioned in the assignment was nowhere to be found, so everything had to be typed in by hand from existing examples in the training material.

Getting the different functionalities to work took quite some time and experimentation and research. It was useful to use print to display the value of the variables at different places in the script.   
  
Initially, I misunderstood the task for deleting data, I thought it had to be deleted from the CDInventroy.txt file, which proved to be quite a challenge. As usual, some solutions found on Stackoverflow.com and other websites proved not the be as easy to implement as excepted.   
After getting clarification that the data should be deleted from in-memory and then written to the file, it made the issue somewhat easier. But it still took a while getting the script working correctly.   
  
One issue remains when the inventory contains 2 or more entries with the same ID number, how to pick the correct one to delete.   
  
I also tried to implement the nice formatting of the strings we learned in module 04, for the output of displaying the row values of the inventory. I played with the print command on line 55, but this failed miserably, so I omitted it.

Last but not least, I tried to implement the feedback I received from Doug regarding my documentation, adding citations where I thought they were needed.

# Posting to GitHub:

The CDInventory.py script and knowledge document can be found on GitHub under this link:  
  
<https://github.com/jlhuerlimann/Assignment_05>

Appendix

1. #-----------------------------------------------------------------------------#
2. # Title: CDInventory.py
3. # Desc: Script for Assignment 05, managing a CD inventory using dictionary.
4. # Change Log: (Who, When, What)
5. # Jurg Huerlimann 2020-Aug-07, Created File from CDInventory\_starter.py script
6. # Jurg Huerlimann 2020-Aug-08 Changed functionality to use dictonary
7. # Jurg Huerlimann 2020-Aug-09 Added functionality to load existing data from file
8. # Jurg Huerlimann 2020-Aug-10 Added functionality to delete info from inventory
9. # Jurg Huerlimann 2020-Aug-11 Added functionality to save inventory to file
10. #-----------------------------------------------------------------------------#
12. # Declared variables
13. strChoice = ''  # User input
14. lstTbl = []     # list of lists to hold data
15. dicRow = {}     #list of data row
16. strFileName = 'CDInventory.txt'  # Name of data file
17. objFile = None  # file object
19. # Get user Input, display menu options
20. **print**('The Magic CD Inventory\n')
21. **while** True:
22. # 1. Display menu allowing the user to choose:
23. **print**('[l] Load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
24. **print**('[d] Delete CD from Inventory\n[s] Save Inventory to file\n[x] Exit')
25. strChoice = input('l, a, i, d, s or x: ').lower()  # convert choice to lower case at time of input
26. **print**()
28. **if** strChoice == 'x':   # Exit the program if the user chooses so
29. **break**
31. **if** strChoice == 'l':   # load inventory from file
32. lstTbl.clear()
33. objFile = open(strFileName, 'r')  # open CDInventory.txt file
34. **for** row **in** objFile:               # read row by row and add to dictionary
35. lstRow = row.strip().split(',')
36. dicRow = {'id': int(lstRow[0]), 'title': lstRow[1], 'artist': lstRow[2]}
37. lstTbl.append(dicRow)
38. objFile.close()
40. **print**('The inventory has been loaded, please use [i] to display it .')
41. **print**()

44. **elif** strChoice == 'a':  # add CD info through questions and add to dictionary
45. strID = input('Enter an ID: ')
46. strTitle = input('Enter the CD\'s Title: ')
47. strArtist = input('Enter the Artist\'s Name: ')
48. intID = int(strID)
49. dicRow = {'id': intID, 'title': strTitle, 'artist': strArtist}
50. lstTbl.append(dicRow)
52. **elif** strChoice == 'i':   # Display the current inventory from in-memory
53. **print**('ID, CD Title, Artist')
54. **for** row **in** lstTbl:
55. **print**(\*row.values(), sep = ', ')
57. **elif** strChoice == 'd': # functionality of deleting an entry
58. delEntry = int(input('What line would you like to delete? PLease enter the ID number: '))
59. **for** entry **in** range(len(lstTbl)):
60. **if** lstTbl[entry]['id'] == delEntry:
61. **del** lstTbl[entry]
62. **print**('Your entry has been deleted from inventory.')
63. **print**()
64. **break**
66. **elif** strChoice == 's':  # Save the data to a text file CDInventory.txt
67. objFile = open(strFileName, 'w')  # using w rewrites the content of the file
68. **for** row **in** lstTbl:
69. strRow = ''
70. **for** item **in** row.values():
71. strRow += str(item) + ','
72. strRow = strRow[:-1] + '\n'
73. objFile.write(strRow)
74. objFile.close()
75. **else**:
76. **print**('Please choose either l, a, i, d, s or x!')

Listing Code for CDInventory.py

1. Source: <https://www.geeksforgeeks.org/difference-between-list-and-dictionary-in-python/> [↑](#endnote-ref-1)
2. Source: Wikipedia [↑](#endnote-ref-2)
3. Source: Google [↑](#endnote-ref-3)