Beemnet Workeneh

02/18/20

FDN 100

Assignment 05

Dictionaries, Keys, Reading Data, and Script Organization

# Introduction

In this assignment we will learn the difference between Dictionaries and List as well as index and keys. We will learn how to read data from a file into a list and into a dictionary. We will learn the benefits of organizing data in two dimensions. We will learn how to organize scripts using separation of concerns, functions, and script templates. We will discover the benefits of error handling and briefly learn about GitHub. Lastly we will update the instructor’s code to Assignment\_04 to use a list of dictionaries as a 2D table and perform the same functionalities ad Assignment\_04.

# Exploration Questions

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

Lists work with sequences of information and dictionaries work with pairs of data. Dictionaries allow you to look up one value using another (just like an actual dictionary allows you to look up a meaning using the vocabulary word). In python you look up a *key* and get its *value[[1]](#footnote-1).* A pair of key and value is called an item. Key is used to find individual values in a dictionary instead of index. Dictionary views cannot be indexed meaning dictionaries are not ordered[[2]](#footnote-2)--however you don’t need to know where an item is in the dictionary you just need to know the key. Dictionaries use the {} operator, the full syntax being {key: value}. Items added to a dictionary are added at the end where as we have more control of where items go in a list by specifying an index or a slice.

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

When using index we use a position number in the sequence to return an individual value in a string, list or tuple. Where as to get an individual value in a dictionary a key has to be used, and a key is not limited to a number, it can be a string or a tuple. If a list is nested you can get to an individual value using multiple indices. It is possible to get an individual value from a list by using a negative or positive index however you can only access an individual item in a dictionary using one unique key. It is possible to use the items() with a function to return a list of tipules containing the key-value pairs. We can use indexing with the items() function to get to an individual key or value in a dictionary without knowing the key2.

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

Reading data from a file has similar steps as writing data from lists to a file. You open a list that will get/read data from a file, you initiate an object file, and open the file using the open(filename, r’) function where ‘r’ is the designator for read mode—although it does not need to be called as it is the default[[3]](#footnote-3). You iterate through a for loop to append to the list. For that you use the strip().split(‘,’) functions together. In this case the script assumes that the file is coma delimited.

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

Similar to how to read data from a file into a list, you begin by opening a dictionary row that you want to add data to. You initiate an object file and open the file using the open(filename, ‘r’). You iterate through a for loop to append to the dictioinary by using the strip().split(‘,’) function. After you’ve read a dictionary row from the file you can append the row to a list that contains the rows of dictioniaries.

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

Organizing data in a two dimensional way allows for easy access of information. It is naturally how we organize information outside of computer programming. We often tabulate data in a way that makes visual sense.

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

Separation of concerns is a design principle for separating a computer program into distinct sections such that each section addresses a separate concern—a set of information that affects the code of a computer program. A common method of separating concerns is by breaking script into data, processing, and presentation(input-output) sections.

*How would you use a function to organize your code?*

Functions allow you to group a set of statements and access them using the function name. Functions allow the processing section of separation of concerns to remain together and organized.

*Why is a script template useful?*

If you have a script template with the header of your choice it prevents you from having to type a header for every new script. You can also add the sections for data, processing, and presentation in the script template so that you can better organize your pseudocode and code once you begin to write.

*Why is error handling (try-except) useful?*

Encountering errors without error handling can lead to script execution getting aborted consequently leaving work and data unsaved or datafile corrupted. Error handling allows more control over the program. For example if you have a code that takes user input to add, subtract, multiply, and divide numbers, and you insert a a try-except code that prints an error statement for inputs that don’t work with the divide operation, the program won’t quit because the user tries to divide a number by zero.

*What is GitHub and why is it used?*

Git is a version control system that stores modifications to a program in a central repository. Developers can collaborate and download new versions of the program, make changes, and upload the newest version. Git is preferred because of its advantages such as storing file changes more efficiently and ensuring file integrity.

GitHub is the cloud based server for hosting and maintaining a group’s code base. GitHub is a global company that is a subsidiary of Microsoft. It is the largest host of source code in the world.

*What is GitHub’s mascot?*

GitHub’s mascot is a “Octocat”: part octopus part cat[[4]](#footnote-4).

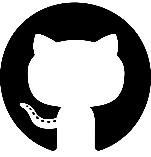
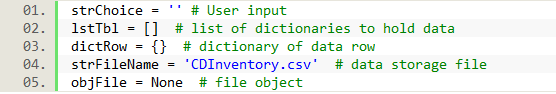


Figure 1 Octocat

# CDInventory Program

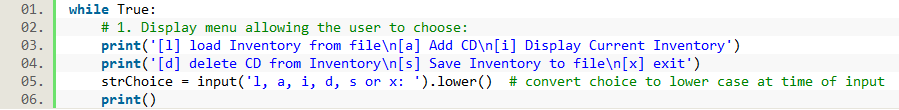
Here I modified the instructors script to use a list of Dictionaries. It was difficult at times to work on code that I did not write from scratch. The way the base code is organized is slightly different. For example for my Assignment 04 I organized the data by ID, Artist, Title whereas the instructor organizes it by ID, Title, and Artist.

The code begins by declaring some variables. This partially resembles SoC in which data is the first section of code, however in the while statement data, processing, and presentation are intermingled. I changed the variable lstRow to dictRow and initiated an empty dictionary instead of an empty list—also changed the comments accordingly. I changed the extension of the file from .txt to .csv.



Listing Declaring variables

Next is starting the while loop and displaying options for the user input which came directly from the instructor’s starter code.

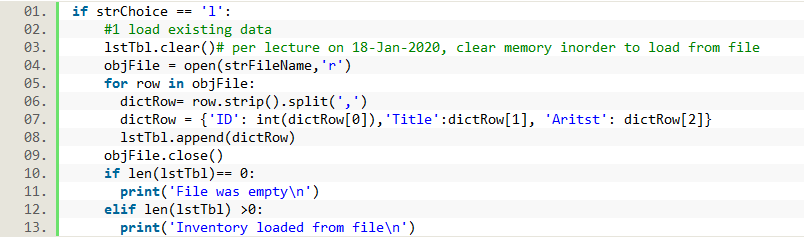
Listing Begin while loop and display options

‘x’ exit

Simply exit the while loop using the break statement. Consequently the program is also exited as there is no block of code after the while loop.

‘l’ load Inventory from file

Here I read data into memory from file. I go row by row in the file and extract the coma delimited values and store them in a dictionary that holds three key-value pairs. I make sure to convert the ID that is stored as a string in the .csv file to an integer. I then append each dictionary to lstTbl which is the memory where I do all the viewing and manipulating of the data. I also added lines of code in case the file to read from was empty. If not empty and data was loaded I display that inventory was loaded. After reading data I close the file.



Listing - load Inventory from file

Table load inventory from file -code execution

|  |  |  |
| --- | --- | --- |
|  | Code executed in spyder | Code executed in anaconda prompt |
| Case 1, file is not empty | The directory is different from my other assignments because I’m using a different computer | The directory is different from my other assignments because I’m using a different computer |
| Case 2, file is empty |  |  |

‘a’ Add CD

Here I take the variables that store the user input values for ID, title, and artist and I pair them with their corresponding key to put into a dictionary. I then append the dictionary into lstTbl.

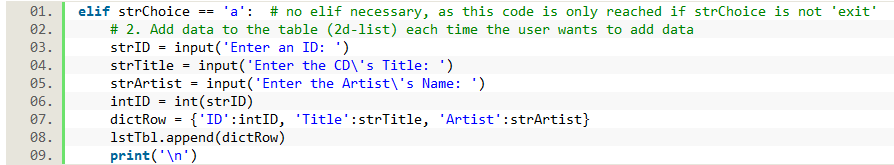
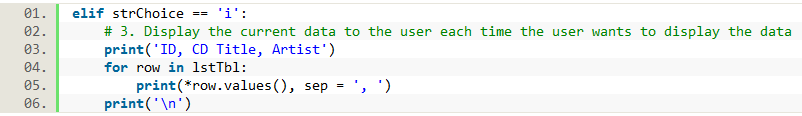
Listing Add CD

Table Add CD - code execution

|  |  |
| --- | --- |
| Code executed in spyder | Code executed in anaconda prompt |
|  |  |

‘i’ Display Current Inventory

Here the only change I made to the code is to the print statement. Instead of printing from rows of a list of a lists it needs to read from rows of a list of dictionaries. Therefore instead of row in needs row.values() to read the values of the dictionaries. We do not to display the keys as that would be redundant and useless.



Listing Display Current Inventory

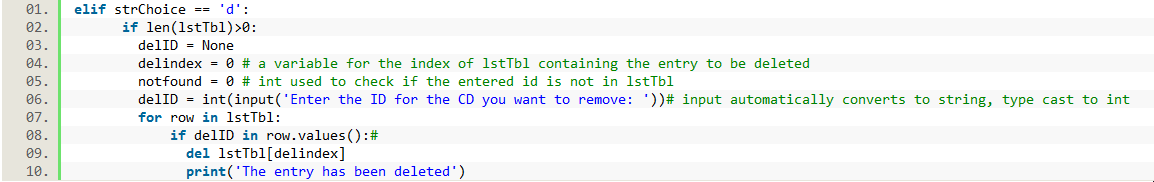
Table Display Current Inventory - code execution

|  |  |
| --- | --- |
| Code executed in spyder | Code executed in anaconda prompt |
|  |  |

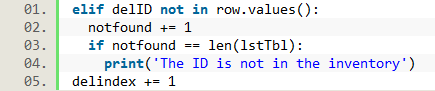
‘d’ Delete CD from Inventory

I split this block of code to three sections.

In the first part, I check if the lstTbl is not empty and we actually can delete entries. If so then I create variables that I use to store user input for entry to be deleted, as well as to track which index of lstTbl containing the entry to be deleted. Here I chose to use the ID values for the user to input in order to delete from inventory. Album titles and IDs would be unique values in a data structure such as this but It felt to me that it would be less computation to iterate through data and compare ID numbers (which are integers ) instead of CD titles (which are strings). This program does not insure unique ID-title-artist groups. Ideally I believe there should unique IDs and the IDs should be auto assigned by the program and not by the user. In this code it falls upon the user to assign a unique ID, otherwise multiple entries could be deleted if they have the same ID.

Listing Delete CD from Inventory pt 1

In the Second part if the lstTbl is not empty but the ID is not in it, I display that the ID is not in the inventory.

Listing Delete CD from inventory pt 2

In the third part , I check if the lstTbl is empty. If it is then I display that there are no entries to delete.



Listing Delete CD from inventory pt 3

Table Delete CD from inventory - Code execution

|  |  |  |
| --- | --- | --- |
|  | Code executed in spyder | Code executed in anaconda prompt |
| Case 1, inventory is not empty and ID exists |  |  |
| Case 2, inventory is not empty and ID does not exist |  |  |
| Case 3, inventory is empty |  |  |

‘s’ Save Inventory to file

Here I open the csv file using write mode so that it overwrites previously existing data in the file. I then iterate row by row in the list of dictionaries and store a coma delimited string row containing the values of dictionaries (the keys are not stored in file as they are assigned by the program within the program). After storing data I close the file.

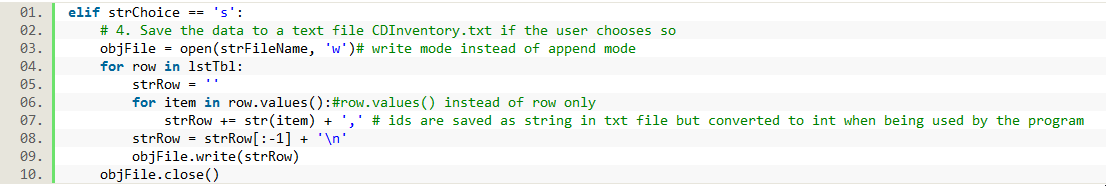
Listing Save Inventory to file

Table Save Inventory to file - code execution

|  |  |
| --- | --- |
| Code executed in spyder | Code executed in anaconda prompt |
|  |  |

# Summary

In this assignment we learned that the main difference between a dictionary and a list is how we retrieve their individual values. Lists are ordered and we get individual values using index. Dictionaries are not ordered and we get individual items by using the key. We read data from a file into a list or a dictionary by initiating an object file and opening the file by using open(filename,r) function. Organizing data in a 2 dimensional way makes sense because that is how we often organize information outside of programming. Separation of concerns breaks up the script into data, processing, and presentation sections. Functions organize a code by collecting the processing portions of a text together in one place. A script template can auto generate the header section of our code and give us a structured place to put the data, processing, and presentation sections of our code in their separate locations. Error handling prevents the script from aborting due to an error and from possible datafile corruption. GitHub is a cloud based server for hosting and maintaining a groups code, it allows for easier collaboration and deployment of source code. GitHub’s mascot is a Octocat. One difficult aspect of the assignment was keeping up with the ever increasing number of operators and functions. Another difficult aspect was working on someone else’s code because of the small yet different styles of coding. Also I noticed the need to slowly consider how a set of data such as a list of dictionaries is organized because with each assignment we are working with larger amount of data and more complex code. Trying to quickly solve or modify someone else’s starter code without fully understanding will lead to a lot of error messages as I discovered while working on this assignment. It was also challenging to anticipate and code for user error such as user trying to read from an empty file—what’s especially difficult was to decide whether to allow a user to read from empty file or to exit the block of code. There were other cases where I felt the code could have been simpler if I decided the user to make errors such as display an empty inventory.

# Appendix

## GitHub Repository

The CDInventory code can be found under the following link

<https://github.com/beemnet20/Assignment_05>

## CDInventory.py

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

# Title: CDInventory.py

# Desc: A script to input, output and store CD inventory data using dictionaries

# Change Log: (Who, When, What)

# DBiesinger, 2030-Jan-01, Created File

# BWorkeneh, 2020-Feb-18, Modified file

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

# Declare variabls

strChoice = '' # User input

lstTbl = [] # list of dictionaries to hold data

dictRow = {} # dictionary of data row

strFileName = 'CDInventory.csv' # data storage file

objFile = None # file object

# Get user Input

print('The Magic CD Inventory\n')

while True:

# 1. Display menu allowing the user to choose:

print('[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')

strChoice = input('l, a, i, d, s or x: ').lower() # convert choice to lower case at time of input

print()

if strChoice == 'x':

# 5. Exit the program if the user chooses so

break

if strChoice == 'l':

#1 load existing data

lstTbl.clear()# per lecture on 18-Jan-2020, clear memory inorder to load from file

objFile = open(strFileName,'r')

for row in objFile:

dictRow= row.strip().split(',')

dictRow = {'ID': int(dictRow[0]),'Title':dictRow[1], 'Aritst': dictRow[2]}

lstTbl.append(dictRow)

objFile.close()

if len(lstTbl)== 0:

print('File was empty\n')

elif len(lstTbl) >0:

print('Inventory loaded from file\n')

elif strChoice == 'a': # no elif necessary, as this code is only reached if strChoice is not 'exit'

# 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)

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

lstTbl.append(dictRow)

print('\n')

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.values(), sep = ', ')

print('\n')

elif strChoice == 'd':

if len(lstTbl)>0:

delID = None

delindex = 0 # a variable for the index of lstTbl containing the entry to be deleted

notfound = 0 # int used to check if the entered id is not in lstTbl

delID = int(input('Enter the ID for the CD you want to remove: '))# input automatically converts to string, type cast to int

for row in lstTbl:

if delID in row.values():#

del lstTbl[delindex]

print('The entry has been deleted')

elif delID not in row.values():

notfound += 1

if notfound == len(lstTbl):

print('The ID is not in the inventory')

delindex += 1

elif len(lstTbl)==0:

print('The inventory is empty')

print('\n')

elif strChoice == 's':

# 4. Save the data to a text file CDInventory.txt if the user chooses so

objFile = open(strFileName, 'w')# write mode instead of append mode

for row in lstTbl:

strRow = ''

for item in row.values():#row.values() instead of row only

strRow += str(item) + ',' # ids are saved as string in txt file but converted to int when being used by the program

strRow = strRow[:-1] + '\n'

objFile.write(strRow)

objFile.close()

else:

print('Please choose either l, a, i, d, s or x!')

1. Dawson, Michael. Python® Programming for the Absolute Beginner, Third Edition. Course Technology PTR, 2009. [↑](#footnote-ref-1)
2. <https://realpython.com/python-dicts/> (retrieved 2/18/20) [↑](#footnote-ref-2)
3. <https://www.geeksforgeeks.org/file-handling-python/> (retrieved 2/18/20) [↑](#footnote-ref-3)
4. <https://en.wikipedia.org/wiki/GitHub#Mascot> (retrieved 2/18/20) [↑](#footnote-ref-4)