Manuel Ortiz

2021/08/08

## IT FDN 110: Introduction to Programming (Python)

Assignment 05

# Assignment 05 / Module 05 work and results

## Introduction

Lessons for week 5 were a bit more challenging. At this point we are breaking our fears or what code is and building new task of putting together code from what we learned so far. Then we step into a new realm of github, where does the code go?

## List – loading data from a file and splits

One of the first operators we learned of is the \* operator. This is used to unpack a list. This is a very convenient way to unpack and format the content when using print().

Text

Description automatically generated

Figure Using the \* operator to unpack a list

## Saving List in a File

We learned this last module, but in this module we learned more about the split

Text

Description automatically generated

Figure Loading data using the split function

Using split() function on comma , to generate a list out of the string read in from the file, with items being separated on comma , in the file.

At the \n at the end of the last item on the first printout. Adding strip() cleans the \n from the string.

Text

Description automatically generated

Figure Split removed the comma

## LAB\_05-A

We used the starter code, which was a version of the CD inventory. My code was not exactly the same as this starter code, it was a challenge to follow along. I did some research on how others worked with these task.

# TODO ask user to input data and store it in the in-memory list

addID = input('Enter ID: ')

addTitle = input('Enter Title: ')

addArtist = input('Enter Artist: ')

intID = int(addID)

lstRow = [addID, addTitle, addArtist]

lstTbl.append(lstRow)

# TODO add code here to write from in-memory list to file

objFile = open(strFileName, 'a')

for row in lstTbl:

strRow = ''

for item in row:

strRow += str(item) + ','

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

objFile.write(strRow)

objFile.close()

# TODO read the file line by line into in-memory list.

strFileName = open('CDInventory.txt', 'r')

print(strFileName.read())

strFileName.close()

# TODO display the data to the user

Text

Description automatically generated

Figure Menu items for starter code lab 5

## Working with Dictionaries

Dictionary keys are a lot like named columns in a spread sheet or database, and the dictionary is a row of data. Like creating a list of list to form a 2D list to resemble a 2D table, dictionaries can be inserted into lists to create a list of dictionaries 2D table.

Text

Description automatically generated

Figure Dictionary values displayed

Text

Description automatically generated

Figure Adding new items in the dictionary

## Improving your Scripts

Using the Separation of Concerns programming method, we learned how to itemize our code to make it easier to work with.

## Modifying Spyder Templates

We also modified the script template in spyder. This was something I had thought about the last two weeks. Always copying the script we used for class to automated scripts in Spyder. I am very happy we covered this.

## Github

I already had a github account, but haven’t really used it much. I did learn how to rename the repository I already created to something easier to identify with the UW class.

Graphical user interface, text, application, email

Description automatically generated

Figure New Github repo for class

## The Assignment: Step 5 - Apply your knowledge

After reviewing the videos, book, and websites, it is time to create your own program:

The task is to create a version of our CD Inventory program that has a menu structure and allows the user to enter CD data, view the current inventory, save data to a CDInventory.txt data file and exit the program.

* Modify the script as required to replace the inner data structure by dictionaries.
* Add the functionality of loading existing data.
* Add functionality of deleting an entry.
* Your finished script must use a list of dictionaries as 2D table.

strChoice = '' # User input

lstTbl = [] # list of lists to hold data

# TODO replace list of lists with list of dicts

dicRow = [] # list of data row

strFileName = 'CDInventory.txt' # 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':

# TODO Add the functionality of loading existing data

strFileName = open('CDInventory.txt', 'r')

print(strFileName.read())

strFileName.close()

pass

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)

lstRow = [intID, strTitle, strArtist]

lstTbl.append(lstRow)

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, sep = ', ')

elif strChoice == 'd':

# TODO Add functionality of deleting an entry

delChoice = input("What delChoice do you want me to delete?: ")

if delChoice in lstTbl:

if delChoice in dicRow:

del dicRow[delChoice]

print("\nOkay, I deleted", delChoice)

else:

print("\nI can't do that!", delChoice, "doesn't exist in the dictionary.")

pass

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:

strRow += str(item) + ','

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

objFile.write(strRow)

objFile.close()

else:

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

## Summary

This week wound up being harder than I was thinking it would be. I had to go back a few chapters and re-watch modules. The reality comes out when you realize that all code has difference. You have to be mindful in what those are and how slight the variant is to make those changes.