Eric Hoyle

02-14-21

Foundations in Programming; Python

Assignment\_05

# Introduction

In this assignment the level of complexity and the amount of “self-learning” I had to do was considerably elevated from previous assignments. This week provided some tough challenges but also a lot of growth in my ability. I continued to develop the CDInventoy.py program and incorporated new functionality. In addition, I created a GitHub user account and create my first repositories. Below will be an overview of how I created my CDInventory.py program with discussion about key features and implementation of newly acquired knowledge.

# Accessing the Menu

Similar to the program I created last week and in conforming to the design principle of separation of concerns (SoC), I started the program out with a “data” section wherein the starting variables are listed. Because we were asked not to use functions in this assignment, I had to modify my approach from last week considerably but in many ways this was to my benefit as it required me to build the program over again which helped to reinforce my understanding.

I also wanted to experiment with some string formatting so I made the title of the program a variable so I could manipulate the format later on with ease.

A picture containing text

Description automatically generated

Figure 1-first lines of the program providing defining variables.

# Program Body

The user is first brought to the menu and asked to make a selection from four available options. Here ,in the print function for the menu I tried to “spice it up” a bit by employing some string formatting. This was a useful exercise for me to play around with the various formatting options, namely the fill and align arguments. In this case I centered the string in the field with a few stars used to fill in on either side. Who doesn’t love how that looks?!

This program is essentially a single-body while loop with seven branches; one for each designated function based on the choice entered by the user at the main menu. The user selects the menu option by inputting the letter corresponding to the menu function (l,a,i,d,s, or x).

# Menu selections a and i

Selections a and i which are used to add a cd entry or display the current inventory were relatively straightforward to build after having done so similarly last week with lists. Here though, by using dictionaries the process is a bit more complex but also, in my mind, made more sense. I struggled here however with assigning values to keys in my first attempt where I had wanted to assign the values directly by way of:

﻿ cdRow['id']= int(input('Enter an ID: '))

I couldn’t get this to work the way I wanted and gave up reverting to assigning each value as a variable and then building the dictionary by inserting the variables as values to preset keys.

Text

Description automatically generated

Figure -dictionary value assignment

I’d like to take the time to go back and review this because I feel like there is a more elegant way to compose this list of dictionaries. I also reverted to self-assignment if ‘ID’ in this version because the approach I used last week, assigning the index based on the length of the list, suffers some fatal setbacks in the presence of menu options that allow the user to delete previous entries. I look forward to learning how to solve that issue. Lastly, I chose to display to the user the newly added data and included a “status” indicator for confirmation. It’s always nice to have some confirmation when you updated your data.

Menu selection i was also very similar to last week’s approach to printing the contents of the list based on iteration of rows in the table. However, this time I incorporated the use of string formatting to make this more readable and more succinct in the lines of code used to print this information. I chose to use the \*rows.values() approach to unpacking the dictionaries because the data appears much cleaner to the user. I developed an understanding for the effect of using \*rows in this case but I would like to better understand the mechanics of this interaction.

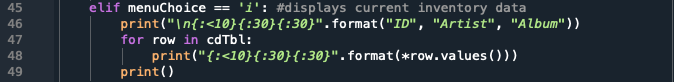


Figure 3- The inventory display branch of CDInventory.py

# Menu Selections l and s

Menu selections for loading and saving data were interesting to work on because they are so practically inverse operations of one another. For the loading ‘l’ function the data is read in, stripped and split into a string. From here the dictionary can be produced using the index position in the string of data to assign the value of the key:value pair. This is nearly identical the process I use to assign values in a new dictionary in the a’’ branch of the menu. What I learned, after struggling for more time than I’ll admit, is that simply displaying the data isn’t enough here. You must construct the list of dictionaries from the data pulled into male it useful in this program. I forsee a lot of packing and unpacking data in this way in the future. Although it did take me a little bit to see this, once I did it “clicked” and all my frustration melted away.

Text

Description automatically generated

Figure - the loading function of the CDInventory.py program

Very similarly, the packaging of data for output follows a similar paradigm. Again, using string formatting here simplifies the process and was an added feature I am learning to appreciate more and more. This is considerably easier than creating an entirely new string from the components of the list as we did last week by iteration. It was also much less error prone in the development of the code. Here again I display a print statement confirming the data has bbeen written to the file. I chose to write the data using the ‘w’ argument to avoid duplication of entries. Overall this has the flaw of potentially losing data if the user doesn’t load the data into memory first. I think I’d like to automate the reading in of the data on the next go-around to avoid this disastrous potential for error.

Text

Description automatically generated

Figure - writing data to file in the CDInventory.py program

# Menu Selection d

Menu selection ‘d’ allows the user to delete data in the inventory in memory. This was probably the most challenging part of the program to write. In class we had looked at the example using the ‘id’ in a dictionary to identify the row number for determining the data to be deleted. But, there were some drawbacks to that approach. Specifically when there were more than one entry with the same ID, or multiple entries were needing to be removed. So, I set out to find a way to compile a list of indexes of entries in the list with the identified key:value to be removed. I figured if I could find the position in the list of dictionaries to be removed I could delete it based on the index. That was harder than it sounds, or at least I need to learn a bit more before I tackle that again. In the end I reverted to paradigm we covered in class but added a confirmation step as a way to error-proof accidental deletion of data and to feel like I contributed something to this block.

Text

Description automatically generated

Figure - the deletion function in CDInventory.py

# GitHub

This week I also created a new suer account and repository on GitHub. I am excited to learn more about this resource but for now All I know is I have an account and can now show my work to others! Here is a link to the source code on [GitHub](https://github.com/ELHoyle/_FDN-Python/blob/main/README.md#assignment_05)

# Summary

So, after last week I was feeling pretty good about everything I had learned to do in Python. This week’s assignment let me know that I don’t really know that much. Nonetheless, I picked up some more skills with string formatting, creating and using dictionaries, and became more comfortable with reading and writing data from a file. Overall though, I think this week’s assignment created more questions than answers for me. I dove into what I thought should be quick fixes and solutions to some problems I saw and found that the solutions weren’t as readily available to me at the level of skill I possess. I also found that it is a little challenging to work form someone else’s starter code. Clearly a practice one can get better at and an important part of learning to understand a different perspective and approach to solving these problems. More growth ahead!

Here it is in Spyder:

Text

Description automatically generated

Figure 7-In Spyder

And in Terminal:

Text

Description automatically generated

Figure -CDInventory in terminal

# Appendix

Source code contextualization from [Planetb’s](http://www.planetb.ca/syntax-highlight-word) [[1]](#footnote-1) web page.

Text

Description automatically generated with medium confidence

Figure 9-Context-highlighted source code

1. Retrieved 02-05-21 [↑](#footnote-ref-1)