Santi Syjuco

03/20/2022

Foundation of Programming (Python)

Assignment\_08 – Knowledge Document

# Introduction

The assignment for this module entailed starting with a blank starter script to build the same CD inventory program that we have been working on for the past couple of weeks but with much less code to start our programs with. There was also an additional requirement to work with Objects.

The coding task for this assignment was to modify the start Assignment 08 code to: (1) Understand the pseudocode then add code to make the application work (2) Add docstrings and headers where applicable (3) Include error handling (4) Run the script in Spyder and Terminal to show that the use cases are working. (5) Create a knowledge document to describing the steps used to complete the assignment.

# Steps Followed

## Step 1: Read Chapter 7 of the book

## Step 2: Read the website and watched the videos in the Assignment handout.

If I were to redo the his module all over again, I would have started with the [website](https://realpython.com/python3-object-oriented-programming/) first, read the textbook, and then went through the module packet. I thought the website did a good job of explaining the basic concepts of object oriented programming especially with the ;’dogs’ example. I thought it was important to understand how to work with multiple attributes before diving deeper into other topics, which is what I observed from the module handout.

## Step 3: Proceeded to tackle the pseudocode

I started with the main body of the code and then worked through each of the specific section requirements.

I started with the main loop first and then went through each of the options, x (exit), I (show inventory), a (add cd), s (save), and then finally l (load data). I realized midway through my coding that the options were arranged in order of increasing difficulty.

I also kept handy other past assignments that I could use to leverage as I went through this assignment. I kept the following assignments and labs open as reference:

1. The Dogs.py example from the Assignment 08 recommended website reading.
2. Assignment07 – for me to be able to leverage error handling.
3. Assignment06 – to be able to pull batches of code to easily recreate the menus
4. Assignment05 – to reference working with lists rather than dictionaries in reading and saving to file.

## Step 4: Challenge 1 of 4 of the assignment

The first challenge I felt of the assignment was working with objects rather than dictionaries for the user’s CD input. At a high level, this was a relatively easy challenge by taking the user’s inputs and then converting them into the CD object’s attributes.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | @staticmethod  **def** add\_cd():  *""" Function to add CD details to memory*    *Args:*  *None.*    *Returns:*  *IO.show\_invenotry(lstOfCDObjects) # this calls the IO function show\_inventory(lstOfCDObjects) to confirm to the user what was added to the table*      *"""*  newlst = []  newlst = IO.input\_cd()  new\_cd = CD(newlst[0], newlst[1], newlst[2])  lstOfCDObjects.append(new\_cd)  **return** IO.show\_inventory(lstOfCDObjects) |

## Step 5: Challenge 2 of 4 of the assignment.

The second challenge then was to ensure that the contents of the lst of objects would be printed to the user in a human readable way. I was able to achieve this by cycling through the individual elements of the table and setting up code to display to the user a more friendly version of the CD attributes.

|  |  |
| --- | --- |
| 1  2  3  4 | **def** \_\_str\_\_(self):  *""" Object instance method to display human friendly version of CD attributes"""*    **return** f"**{**self.cdid**}** **{**self.cdtitle**}** **{**self.cdartist**}**" |

## Step 6: Challenge 3 of 4 of the assignment.

The third challenge was to then save the contents of the list to a text file. This was a little tricky because directly saving a string version of the table would lead to a cryptic text, which was the location of the CD objects in memory. I opted to go with saving a version of the list based on what was printed out of the [i] show inventory option.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | @staticmethod  **def** write\_file(file\_name, table):  *"""Function to save the CDs currently in memory to the text file CDInventory.txt*    *Args:*  *file\_name: name of the file to be written to*  *table: the list of CD objects that will be written to the file*    *Returns:*  *None.*    *"""*  objFile = open(strFileName, 'a')  **for** row **in** table:  strCDs = ''  strCDs += str(row) + ','  strCDs = strCDs[:-1] + '**\n**'  objFile.write(strCDs) |

## Step 7: challenge 4 of 4 of the assignment.

The hardest challenge I realized was trying to load data from the textfile that was a list of CD objects.

I got stuck here and instead settled for loading the data as items from a list

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | @staticmethod  **def** read\_file(file\_name, table):  *"""Function to manage data ingestion from file to a list of dictionaries*  *Reads the data from file identified by file\_name into a 2D table*  *(list of dicts) table one line in the file represents one dictionary row in table.*  *Args:*  *file\_name (string): name of file used to read the data from*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*  *Returns:*  *None.*  *"""*  objFile = open(file\_name, 'r')  **for** row **in** objFile:  lstRow = row.strip().split(',')  lstOfCDObjects.append(lstRow) |

I wasn’t sure if the items were actually printed out as objects but it seemed to be compatible with any new CD’s I’d add to the listofCdobjects table. The telling sign though that something was amiss was the formatting of the CDs that were displayed using the [i] option after loading the CDs from the text file.

I was very tempted to convert the save file to a .dat to pickle the file to make it easier to work with from a read and write standpoint.

## Step 8: Added docustrings to my functions and commented on my code to make it easier for others to follow along.

I deleted all the old TODOs and TO Done comments. I then added new TO Done comments to show the changes I made for this assignment.

## Step 9: Added Error handling

I added error handling for the following cases:

1. Added a ‘finally’ clause to allow the user to proceed with main menu even if there was no textfile created yet.
2. Added try – except to force the user to input an integer for the CD ID.
3. Added try – except when loading the show inventory option.
4. Added warning for the save and load functions.

## Step 10: Capture Code working on Spyder

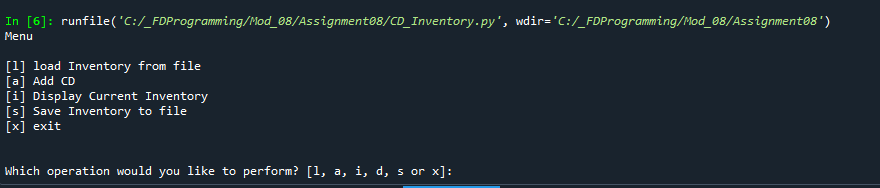


Figure Menu

Text

Description automatically generated

Figure 2 Add CD option

Text

Description automatically generated

Figure 3 Display current inventory

Text

Description automatically generated

Figure 4 Save Option

Graphical user interface, text, application

Description automatically generated

Figure 5 Save Inventory to File

Text

Description automatically generated

Figure 6 Load inventory Note that the formatting of the loaded CD is different from the newly inputted CDs

I was unable to debug this issue.

Graphical user interface, text

Description automatically generated

Figure Exit

## Step 11: Capture Code Working on Terminal

Text

Description automatically generated

Figure 8 Program startup

Text

Description automatically generated

Figure Add CD

Text

Description automatically generated

Figure 11 Display option

Graphical user interface

Description automatically generated

Figure 12 Save option

Text

Description automatically generated

Figure 13 Load Option.

Figure Exit

## Step 10: Uploaded python script and assignment to Github

[syjuco/Assignment08: Assignment for Module\_08 for UW Course Foundations of Programming (Python) (github.com)](https://github.com/syjuco/Assignment08)

# Summary

The most challenging parts of this assignment was reloading the data as a list of CD objects from a text file. I couldn’t figure out how to save the contents of the list of CD objects into a text file that would make it easy to read out and append back to a list of CD objects as well. I felt that this could be done much easier by pickling the list of objects into a .dat file and then unpickling them again out of the .dat file. It really would have been helpful to see how this was done more incrementally through an example of a refresher material before tackling this problem.