Anna Willie

December 5, 2021

IT FDN 110

Assignment 08

OOP and Python

# Introduction

For this week’s assignment our goal was to change our CD inventory data type from a list of dictionaries to a list of objects. This was done by creating a class that uses constructors, fields, and properties.

# Topic 1

First, I worked on my CD class and worked on the syntax of making a new CD into an object and how to add that to a list. I struggled with this originally because I was using self as a parameter, but that is not necessary for a static method. Then I worked on my delete function, I had issues with connecting the CD ID to the number that the user inputs, but once I figured out the syntax it worked smoothly.

For this assignment I chose to pickle the data again, because then I do not have to format it for writing to a text file. I included my error handling around writing and reading to a file to make sure it was loading correctly.

Here are some screen shots of my code working on Spyder:

Text

Description automatically generated

Figure 1 add and delete functions

Text

Description automatically generated

Figure 2 read and write functions

My script running in a shell:

Graphical user interface, text

Description automatically generated

Figure 3 script running in a shell

Here is a link to my [GitHub](https://github.com/awillie32/Assignment_08).

# Summary

For this week’s class it was interesting to see how object-oriented programming can encapsulate data. I did struggle a lot with connecting the examples given from class and through the book to this assignment. I also had trouble researching outside the given material, can you help me with some better resources that can guide me in the correct direction? I can’t make office hours due to work, and only really have a lot of time to work on an assignment the day, and the day prior, it’s due. Overall, I thought this assignment was challenging and I am interested in seeing how we can dive further into object-oriented programming.

# Appendix

This was accessed from [Syntax Generator.](https://ajblk.github.io/SyntaxHighlightGenerator-v3.0/OnlineGenerator.html)[[1]](#footnote-1)

|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040  041  042  043  044  045  046  047  048  049  050  051  052  053  054  055  056  057  058  059  060  061  062  063  064  065  066  067  068  069  070  071  072  073  074  075  076  077  078  079  080  081  082  083  084  085  086  087  088  089  090  091  092  093  094  095  096  097  098  099  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202 | #------------------------------------------#  # Title: Assignmen08.py  # Desc: Assignnment 08 - Working with classes  # Change Log: (Who, When, What)  # DBiesinger, 2030-Jan-01, created file  # DBiesinger, 2030-Jan-01, added pseudocode to complete assignment 08  # AWillie, 2021-Dec-03, added script for adding a CD  # AWillie, 2021-Dec-04, added more script  # AWillie, 2021-Dec-05, worked on object script and finalized everything else  #------------------------------------------#    # -- DATA -- #  strChoice **=** '' # User input  strFileName **=** 'cdInventory.dat'  lstOfCDObjects **=** []  **import** pickle  **class** CD(object):      """Stores data about a CD:        properties:          ID: (int) with CD ID          Title: (string) with the title of the CD          Artist: (string) with the artist of the CD      methods:          add\_CD(list)          del\_CD(list)        """      # Fields      ID **=** int()      Title **=** ''      Artist **=** ''  **def** \_\_init\_\_(self, ID, Title, Artist):          self.\_\_ID **=** ID          self.\_\_Title **=** Title          self.\_\_Artist **=** Artist      @property  **def** ID(self):  **return** self.\_\_ID      @property  **def** Title(self):  **return** self.\_\_Title      @property  **def** Artist(self):  **return** self.\_\_Artist      @staticmethod  **def** add\_CD(cd):  **global** lstOfCDObjects          lstOfCDObjects.append(cd)      @staticmethod  **def** del\_CD(ID):  **global** lstOfCDObjects          intRowNr **=** **-**1          blnCDRemoved **=** False  **for** cd **in** lstOfCDObjects:              intRowNr **+=** 1  **if** cd.ID **==** ID:  **del** lstOfCDObjects[intRowNr]                  blnCDRemoved **=** True  **break**  **if** blnCDRemoved:              print('The CD was removed')  **else**:              print('Could not find this CD!')        # -- PROCESSING -- #  **class** FileIO:      """Processes data to and from file:        properties:        methods:          save\_inventory(file\_name, lst\_Inventory): -> None          load\_inventory(file\_name): -> (a list of CD objects)        """      # load file      @staticmethod  **def** read\_file(FileName):          """"Function to read a binary file and to return the list data            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.          """  **global** lstOfCDObjects          lstOfCDObjects.clear()          with open(FileName, 'rb+') as objFile:  **try**:                  lstOfCDObjects **=** pickle.load(objFile)  **except**:                  print('Something has gone wrong!')  **else**:                  print ('Successful read: %s' **%** (objFile))              objFile.close()  **return** list(lstOfCDObjects)      @staticmethod  **def** write\_file(FileName):          # Added code here          """Function to save data to a binary file            Takes the current memory and moves it to a binary file            Args:              file\_name(string): name of file used to copy memory to              table (list of dict): 2D data structure holding the inventory          Returns:              None.          """  **global** lstOfCDObjects          with open(FileName, 'wb+') as objFile:  **try**:                  pickle.dump(lstOfCDObjects, objFile)  **except**:                  print('An error has occured')  **else**:                  print ('Successful write: %s' **%** (objFile))          objFile.close()    # -- PRESENTATION (Input/Output) -- #  **class** IO:      # User Menu  **def** print\_Menu():           print('\n[a] add data to list\n[w] to write data to file\n[r] to read data from file')           print('[d] display data\n[i] to show inventory\n[x] to quit')      # Menu choice  **def** user\_Choice():          choice **=** ' '  **while** choice **not** **in** ['a', 'w', 'r', 'd', 'i', 'x']:              choice **=** input('Which operation would you like to perform? [a, w, r, d, i, x]: ').lower().strip()          print()  # Add extra space for layout  **return** choice      # Inventory  **def** show\_inventory():  **global** lstOfCDObjects           print(':::::::The Current Inventory: :::::::')           print('ID\tCD Title (by: Artist)\n')  **for** cd **in** lstOfCDObjects:              print('{}\t{} (by: {})'.format(cd.ID, cd.Title, cd.Artist))           print(':::::::::::::::::::::::::::::::::::::')      # CD information  **def** CD\_addition():          ID **=** input('Please provide an ID: ')          title **=** input('Please provide the CD title: ')          artist **=** input('Please provide the CD artist: ')  **return** int(ID), title, artist  # -- Main Body of Script -- #  # Menu loop  # Load data from file into a list of CD objects on script start  # Display menu to user      print('Write or Read file data.')  **while** True:      IO.print\_Menu()      strChoice **=** IO.user\_Choice()      # let user exit program  **if** strChoice **==** 'x':  **break**      # let user add data to the inventory  **if** strChoice **==** 'a':          intID, strtitle, strartist **=** IO.CD\_addition()          newCD **=** CD(intID, strtitle, strartist)          CD.add\_CD(newCD)          IO.show\_inventory()  **continue**       # let user load inventory from file  **elif** strChoice **==** 'r':          print('WARNING: If you continue, all current memory will be lost.')          strYesNo **=** input('Load inventory from file? [y/n] ').strip().lower()  **if** strYesNo.lower() **==** 'y':              print('reading file now...')              FileIO.read\_file(strFileName)              IO.show\_inventory()  **else**:              input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')              IO.show\_inventory()  **continue**  # start loop back at top           # show user current inventory  **elif** strChoice **==** 'i':          IO.show\_inventory()  **continue**        # let user save inventory to file  **elif** strChoice **==** 'w':          IO.show\_inventory()          strYesNo **=** input('Save this inventory to file? [y/n] ').strip().lower()          FileIO.write\_file(strFileName)  **continue**  **elif** strChoice **==** 'd':          intIDDel **=** input('Which ID would you like to delete? ')          CD.del\_CD(int(intIDDel))          IO.show\_inventory()  **continue**  **else**:          print('Please choose either a, l, s, d, or exit!') |

1. Reviewed December 5, 2021 [↑](#footnote-ref-1)