<Songli Zhu>

<2022-Mar-18>

<Foundations of Programming (Python)>

<Assignment 08>

# Introduction

The assignment required me to continue modify a CD inventory program by utilizing the concept of Object Oriented Programming (OOP). It already has some pseudocodes and the goal here to add code and use class and objects to finish the assignment.

First thing is to finish the CD class. The docstring has suggested that there are three properties and no methods. So, three attributes (id, title, and artist) were added in the constructor. Since we don’t want the user to modify these attributes directly, I will use private methods and create each property for each of them. Although the id should be integer, but setter function won’t check for it when you create a new instance of CD class. So, I skip it and add the handling error in IO class instead.

Next thing, finish the FileIO class. No properties and two methods were mentioned in the docstring. So, I decided to use static method instead of classic method. The codes are similar to pervious assignment, but I noticed that I cannot use dictionary to store the CD information (get the hint from Friday night Office hour). I should directly store the instance into the list. So, the list contains multiple CD class instances. To save it to the file, I decided to make an empty list and store the id, title and artist information into the list and write it into the file. Load\_inventory is easier, just need to store the data into an instance of CD class and append it to the list of CD objects.

Then, to finish the IO class. Still decided to use static methods (seems easier). Print\_menu and menu\_choice methods are easy to do. Be careful about the display method because it is a list of CD objects instead of dictionaries. Lastly, to finish cd\_data() method, use input function and add handing error in case the CD id is not a integer.

The main body is straightforward. Add another handling error for load data from file (file does not exist or empty document) and use while loop to perform the codes.

Lastly, run the script, check the file in a text editor. Make sure it works in both terminal and Spyder.

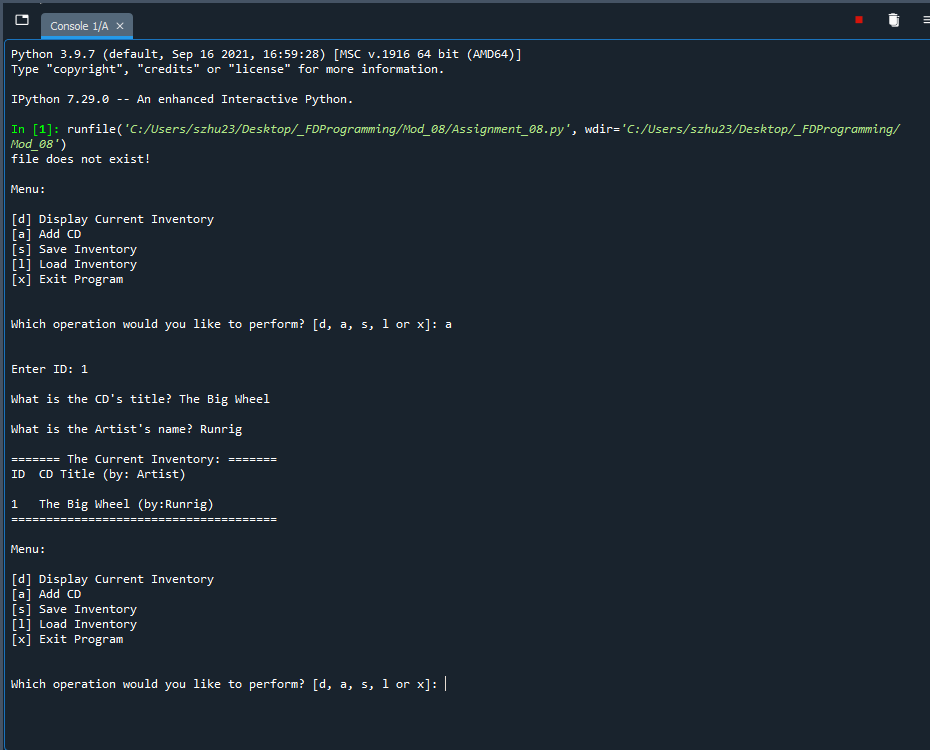


Figure Screen capture of input from CDInventory script in Spyder (Note: handling error when dat file is not exist).

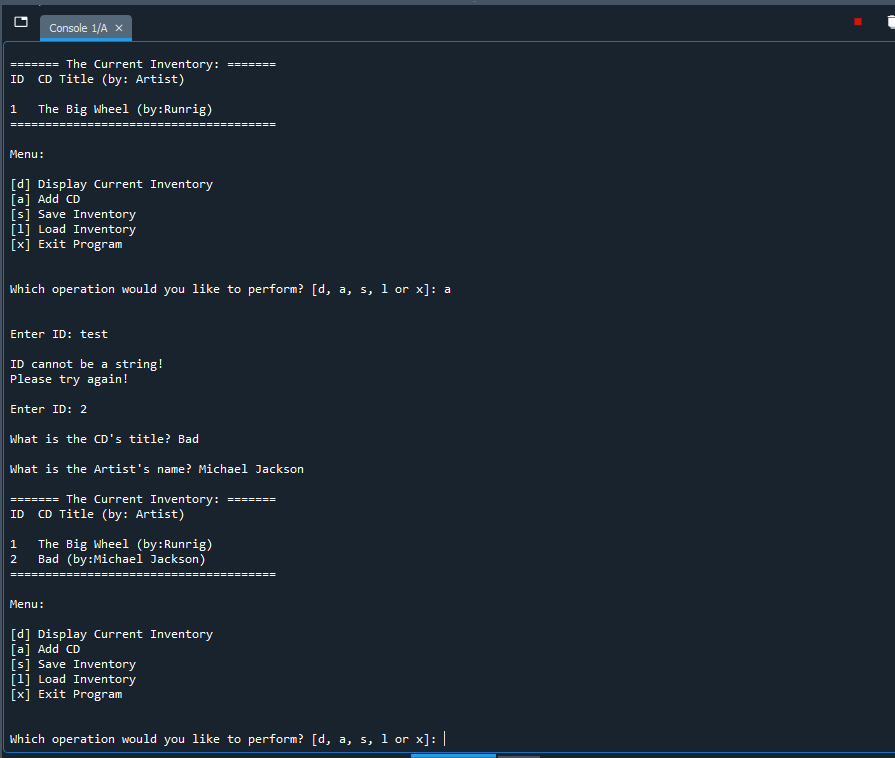


Figure Screen capture of deletion from CDInventory script in Spyder (Note: handling error when ID input is not integer).

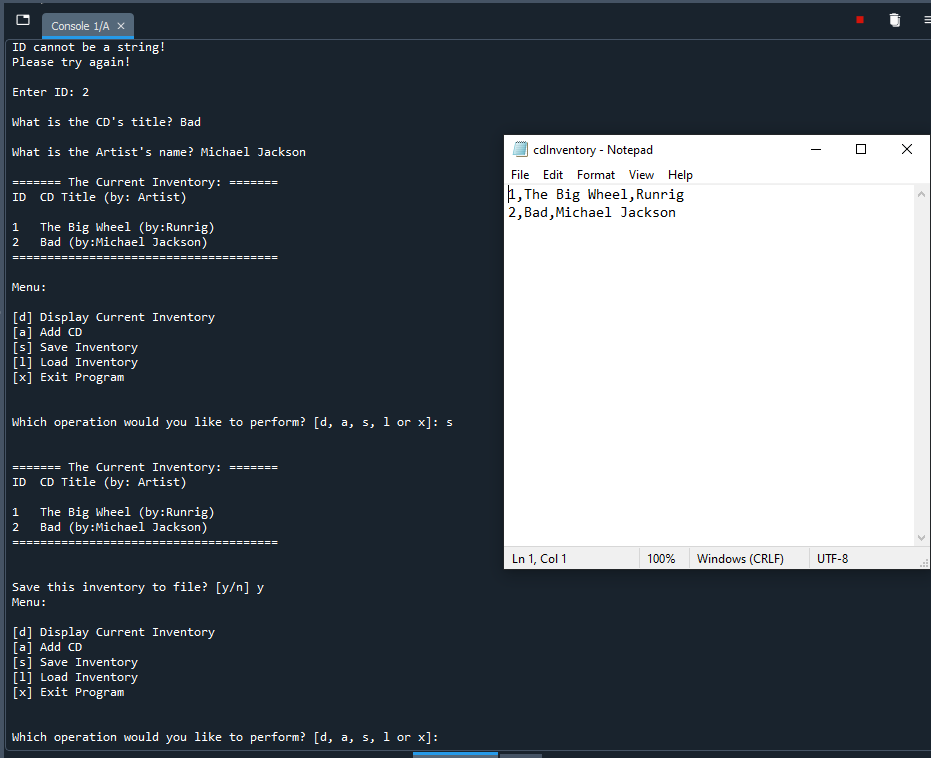


Figure Screen capture of load and save function of cdInventory scripts in Spyder (Note: CDInventory.txt file also included).

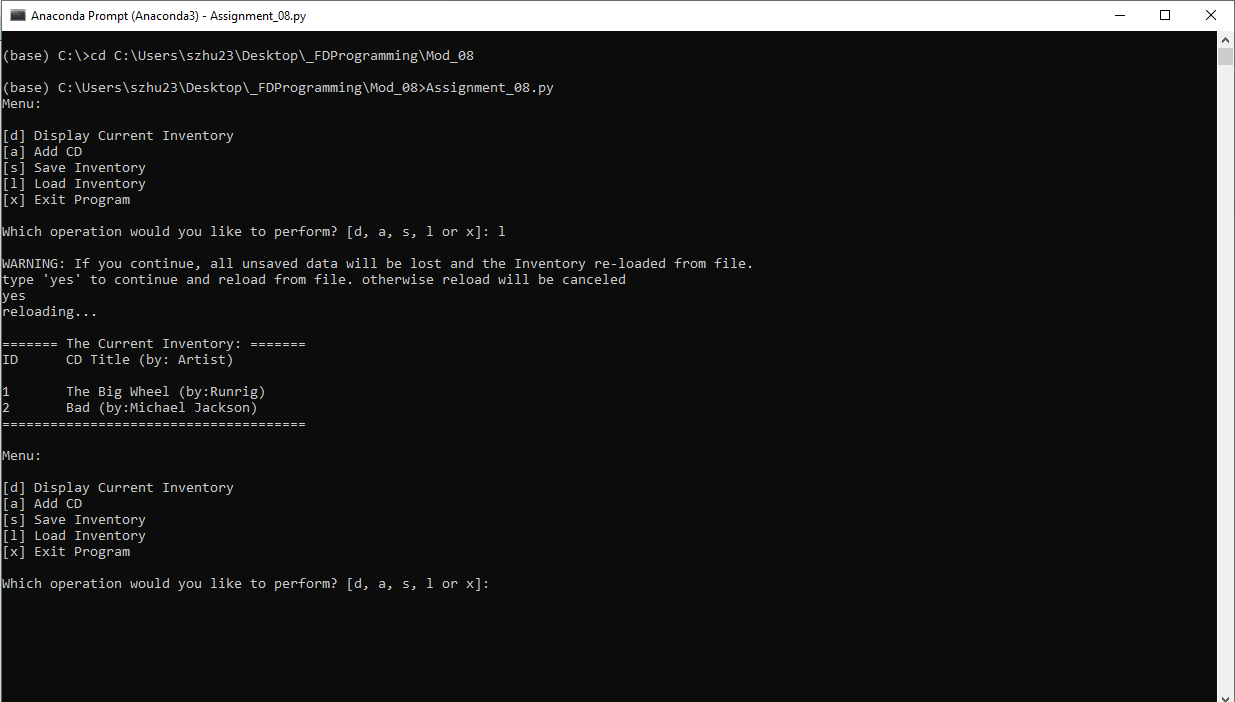


Figure Screen capture of input from CDInventory script in Terminal (Note: load inventory from cdInventory.txt which is created just now from Spyder).

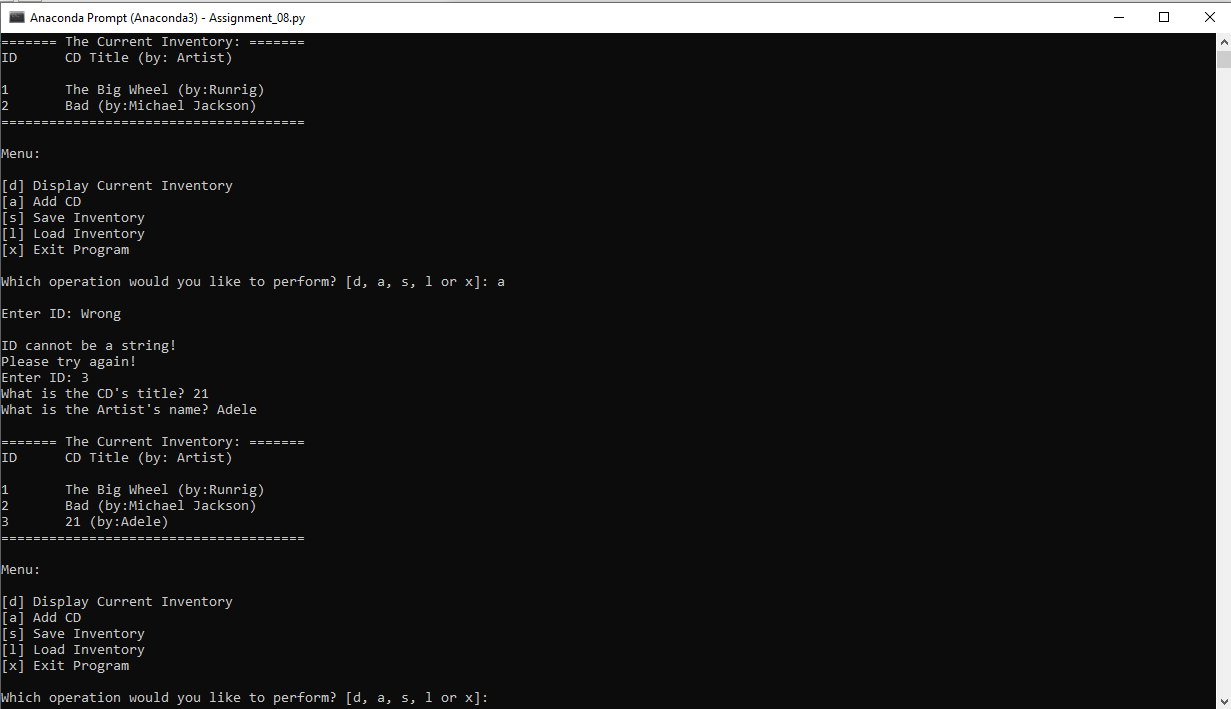


Figure Screen capture of deletion from CDInventory script in Terminal (Note: handling error when ID input is not integer).

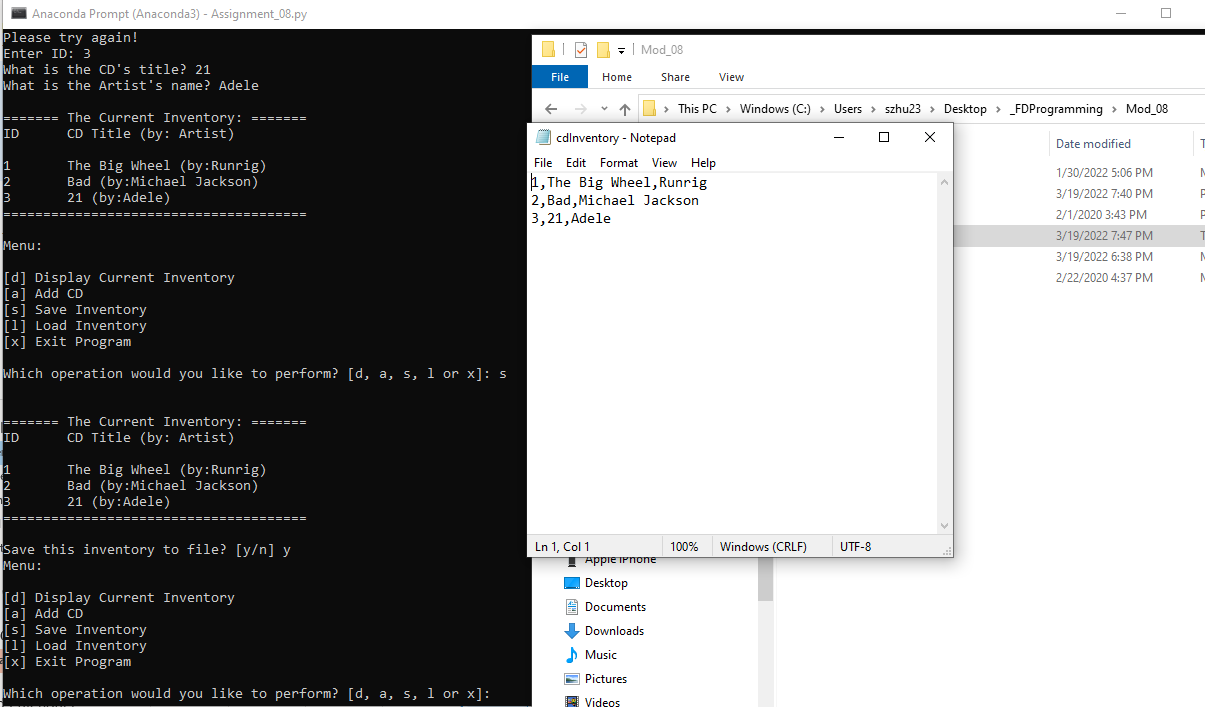


Figure Screen capture of load and save function of CDInventory scripts in Terminal (Note: cdInventory.txt file also included).

# Summary

From module 8, I learnt the concept of Object Oriental Programming. Although we have used these before (including list, dictionary, dot function), I am still confused about the class concepts, including attributes, self, property, setter, methods (mainly static methods in this module). It took a while to understand these ideas especially some private and public methods concepts are introduced at the same time. I think the whole idea here is about encapsulation. It is like a blueprint for the house when you build a class. To follow the blueprint, you can create various houses (instances) based on the concept of blueprint (class). Then, they could have different parameters of attributes, properties, and even methods. But they shared the same concept. I am still trying to understand the property, getter, setter function and methods to make sure I use them properly. I am still confused about the self and have no idea when and how I should use it in methods. When I should use static methods or classic methods (the module doesn’t cover it). Overall, this is totally different than previous knowledge. I know this is important, but still get used to it. From the assignment, I still haven’t enjoyed benefit from the OOP concept. But it is a start.

Appendix

GitHub link: <https://github.com/synbiomotif/Assignment_08>

## List function assignment\_08.py

Using [Saravjishut](https://saravjishut.org/syntax) (external reference) [[1]](#footnote-1)web page

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  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  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236 | *#------------------------------------------#*  *# 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*  *# Songli Zhu, 2022-Mar-18, add codes with classes and objects of OOP*  *#------------------------------------------#*  *# -- DATA -- #*  strFileName = 'cdInventory.txt'  lstOfCDObjects = []  **class** **CD**:  *"""Stores data about a CD:*  *properties:*  *cd\_id: (int) with CD ID*  *cd\_title: (string) with the title of the CD*  *cd\_artist: (string) with the artist of the CD*  *methods:*  *"""*  *# TODONE Add Code to the CD class*  *# == Constructor == #*  **def** \_\_init\_\_(self, intID, strTitle, stArtist):  self.\_\_id = intID  self.\_\_title = strTitle  self.\_\_artist = stArtist    *# -- Properities -- #*  @property  **def** cd\_id(self):  **return** self.\_\_id    @property  **def** cd\_title(self):  **return** self.\_\_title  @property  **def** cd\_artist(self):  **return** self.\_\_artist  *# -- 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)*  *"""*  *# TODONE Add code to process data from a file*  @staticmethod  **def** save\_inventory(file\_name, lst\_Inventory):  *"""save current inventory into the file*  *Args:*  *file\_name (string): name of file used to save the data from list of CD objects*  *lst\_Inventory (list): list of CD objects that holds the data during runtime*  *Returns:*  *None*  *"""*  objFile = open(file\_name, 'w')  **for** row **in** lst\_Inventory:  lst = []  lst.extend([row.cd\_id, row.cd\_title, row.cd\_artist])  lst[0] = str(lst[0])  objFile.write(','.join(lst) + '**\n**')  objFile.close()    *# TODONE Add code to process data to a file*  @staticmethod  **def** load\_inventory(file\_name):  *"""load data from file into the list of inventory*  *Args:*  *file\_name: (string): name of file which includes the CD data*  *Returns:*  *None*  *"""*  lstOfCDObjects.clear()  objFile = open(file\_name, 'r')  **for** line **in** objFile:  data = line.strip().split(',')  cd\_info = CD(data[0], data[1], data[2])  lstOfCDObjects.append(cd\_info)  objFile.close()  *# -- PRESENTATION (Input/Output) -- #*  **class** **IO**:  *# TODONE add docstring*  *"""Handling Input / Output*    *properties:*  *methods:*  *print\_menu(): -> None*  *menu\_choice(): -> None*  *show\_inventory(): -> None*  *cd\_data(): -> (a CD object)*  *"""*  *# TODONE add code to show menu to user*  @staticmethod  **def** print\_menu():  *"""Displays a menu of choices to the user*  *Args:*  *None.*  *Returns:*  *None.*  *"""*  print('Menu:**\n\n**[d] Display Current Inventory**\n**[a] Add CD**\n**[s] Save Inventory')  print('[l] Load Inventory**\n**[x] Exit Program**\n**')    *# TODONE add code to captures user's choice*  @staticmethod  **def** menu\_choice():  *"""Gets user input for menu selection*  *Args:*  *None.*  *Returns:*  *choice (string): a lower case string of the users input out of the choices d, a, s, l or x*  *"""*  choice = ' '  **while** choice **not** **in** ['d', 'a', 's', 'l', 'x']:  choice = input('Which operation would you like to perform? [d, a, s, l or x]: ').lower().strip()  print() *# Add extra space for layout*  **return** choice  *# TODONE add code to display the current data on screen*  @staticmethod  **def** show\_inventory(lstOfCDObjects):  *"""Displays current inventory table*  *Args:*  *lstOfCDObjects (list of CD objects): list of CD objects that holds the data during runtime.*  *Returns:*  *None.*  *"""*  print('**\n**======= The Current Inventory: =======')  print('ID**\t**CD Title (by: Artist)**\n**')  **for** row **in** lstOfCDObjects:  print('**{}\t{}** (by:**{}**)'.format(row.cd\_id, row.cd\_title, row.cd\_artist))  print('======================================**\n**')  *# TODONE add code to get CD data from user*  @staticmethod  **def** cd\_data():  *"""Gets user input for cd information*  *Args:*  *None.*  *Returns:*  *cd\_info (object): an instance of class CD store cd information from user input*  *"""*  flag = 1  **while** flag:  strID = input('Enter ID: ').strip()  **try**:  intID = int(strID)  flag = 0  **except**:  print('**\n**ID cannot be a string!**\n**Please try again!')  strTitle = input('What is the CD**\'**s title? ').strip()  stArtist = input('What is the Artist**\'**s name? ').strip()  cd\_info = CD(strID, strTitle, stArtist)  **return** cd\_info  *# -- Main Body of Script -- #*  *# TODONE Add Code to the main body*  *# Load data from file into a list of CD objects on script start*  **try**:  FileIO.load\_inventory(strFileName) *# it won't work if the file exist and empty causing out of range*  **except** **FileNotFoundError**:  print('file does not exist!**\n**')  *# Display menu to user*  **while** **True**:  IO.print\_menu()  strChoice = IO.menu\_choice()  *# show user current inventory*  **if** strChoice == 'd':  IO.show\_inventory(lstOfCDObjects)  **continue** *# start loop back at top.*  *# let user add data to the inventory*  **elif** strChoice == 'a':  cd = IO.cd\_data()  lstOfCDObjects.append(cd)  IO.show\_inventory(lstOfCDObjects)  **continue**  *# let user save inventory to file*  **elif** strChoice == 's':  *# Display current inventory and ask user for confirmation to save*  IO.show\_inventory(lstOfCDObjects)  strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()  *# Process choice*  **if** strYesNo == 'y':  *# save data*  FileIO.save\_inventory(strFileName, lstOfCDObjects)  **else**:  input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')  **continue** *# start loop back at top.*  *# let user load inventory from file*  **elif** strChoice == 'l':  print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')  strYesNo = input('type **\'**yes**\'** to continue and reload from file. otherwise reload will be canceled**\n**') *# add extra \n*  **if** strYesNo.lower() == 'yes':  print('reloading...')  FileIO.load\_inventory(strFileName)  IO.show\_inventory(lstOfCDObjects)  **else**:  input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')  IO.show\_inventory(lstOfCDObjects)  **continue** *# start loop back at top.*  *# let user exit program*  **elif** strChoice == 'x':  **break**  **else**:  print("**\n**Sorry, but", strChoice, "isn't a valid choice.") |
|  |  |

1. Retrieved 2022-Mar-19 [↑](#footnote-ref-1)