Name: Fabio Sacca

Date: Dec-04, 2022

Course: Foundations of Programming (Python)

Title: Assignment 08

# Assignment 08

## Introduction:

In this document, I will describe the steps I took to complete Assignment 8 completing the tasks identified in ‘Assignment\_08\_Starter.py’ and including error handling. I will then show the script running in Spyder and Terminal Mac window.

## Topic 1: Add code following pseudocode instructions

The starter code uses a strFileName variable, which is assigned to text file ‘cdInventory.txt’ for permanent data store.

### Add Code to the CD class

The CD class docstring indicates this class organizes functions that serve the purpose to store data about a CD. While it was not explicitly called out in the pseudocode, I understood this class to organize ALL functions that process data in runtime. So, I added two main functions:

* load\_cd: processes user request to load CD inventory from a file
* add\_cd: manages data ingestion from user input to a list of dictionaries

### 1.2. Add Code to the FileIO class

In this case the pseudocode is more explicit, so completed the following tasks by adding new functions, including error handling as appropriate:

* load\_inventory: manages data ingestion from file to a list of dictionaries
* save\_inventory: manages data storage from a list of dictionaries to a file

### 1.3. Add Code to the IO class

In this case the pseudocode is more explicit, so completed the following tasks by adding new functions, including error handling as appropriate:

* print\_menu: displays a menu of choices to the user
* menu\_choice: gets user input for menu selection
* show\_inventory: displays current inventory table
* get\_user\_input: function to get the user input for adding a CD entry

### 1.4. Add Code to the main body

Following the requirements of pseudocode, I’ve called the appropriate classes and functions to perform the required tasks:

* Display menu to user
* Show user current inventory
* Let user add data to the inventory
* Let user save inventory to file
* Let user load inventory from file
* Let user exit program

My understanding of the assignment was to ensure I can follow instructions of other developers and meet coding practices and conventions for docstrings, even if most of the code was repurposed from previous assignments.

For the purpose of applying the learnings of Module 8, I’ve attempted to better define properties (especially cd\_id, cd\_title, and cd\_artist in CD and IO classes, but I think my code presents still areas of opportunity.

## Topic 2: Running my script

I saved the script as 'CDInventory.py' in a folder called 'Assignment08' within the course folder '\_FDProgramming', and ran the script in Spyder. The image below shows specifically the error handling responses working on the program.

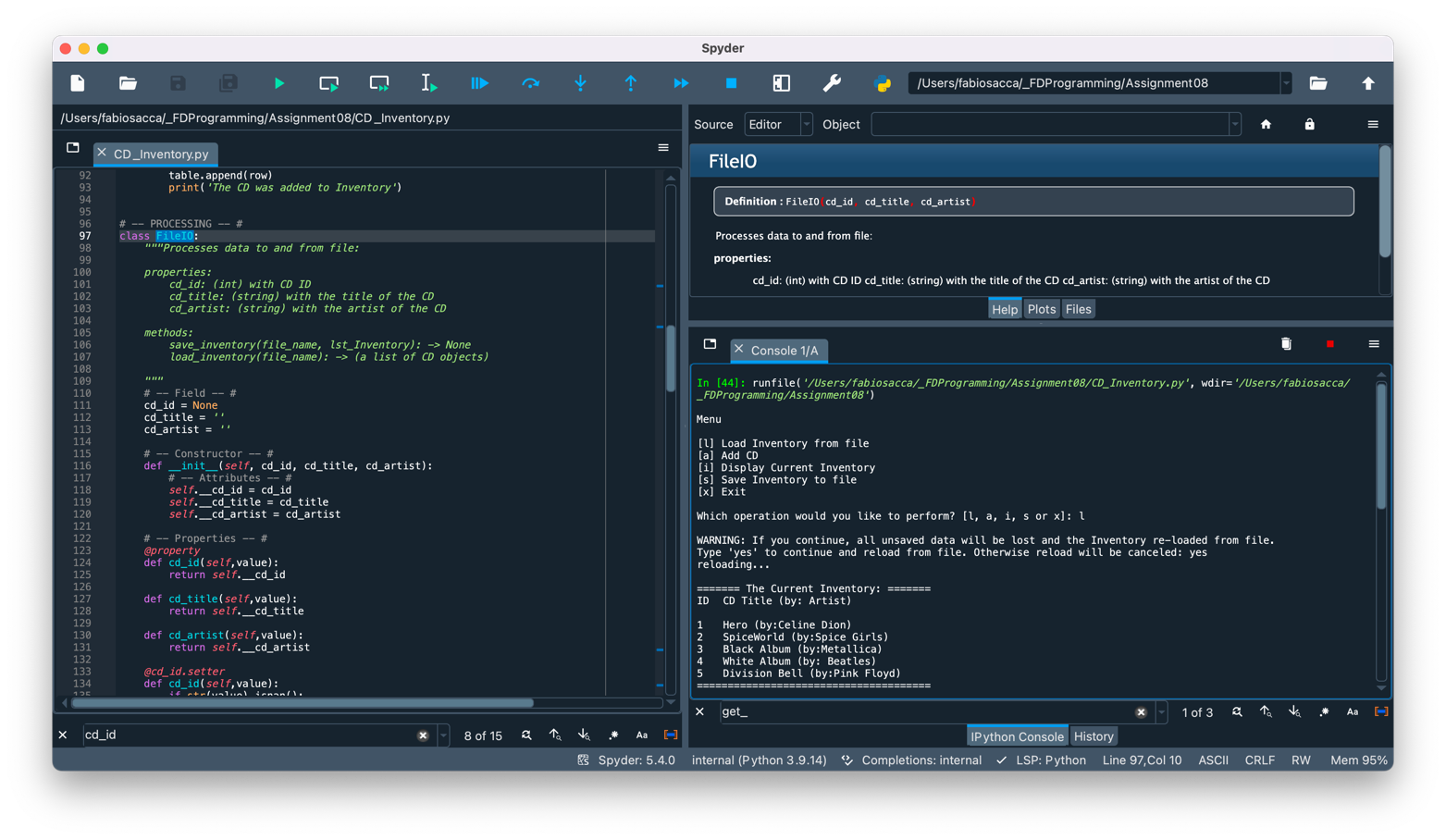


Figure 1 - Screen capture of Spyder returning CDInventory.py script

A screenshot of a computer

Description automatically generated with medium confidence

Figure 2 - Screen capture of Spyder returning CDInventory.py script (continued)

A screenshot of a computer

Description automatically generated with medium confidence

Figure 3 - Screen capture of Spyder returning CDInventory.py script (continued)

I repeated this step in a terminal window. The images below shows that the script works in my computer.

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Figure 4 - Screen capture of OS Terminal returning CDInventory.py script

Figure 5 - Screen capture of OS Terminal returning CDInventory.py script (continued)

## Topic 5: Verifying that my script worked

I located the file ‘cdInventory.dat and opened it in a text editor. The image below shows that the data I entered has been written to the file

Text

Description automatically generated

Figure 6 - Screen capture of cdInventory.txt displaying data entered.

## Topic 6: Submitting my work on GitHub

Both the CDInventory.py and the present knowledge document are posted on the following public repository

## <https://github.com/fabiosacca/Assignment_08>

## Summary

In this assignment, I covered the steps needed to:

* Create ‘CD Inventory’ program from a starter code.
* Follow instructions and documentation practices of other developers as defined in pseudocode
* Run the script in Spyder and a terminal window.
* Verify that the script in fact wrote in cdInventory.txt the data I entered in the correct format.
* Submit my work on a public GitHub repository.

# Appendix A

## Code of CDInventory.py script presented with [**Syntax Highlighter**](https://saravjishut.org/syntax)**[[1]](#footnote-1)**

|  |  |
| --- | --- |
| 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  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331 | *#------------------------------------------#*  *# Title: CD\_Inventory.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*  *# fabiosacca, 2022-Dec-04, Completed Todos*  *#------------------------------------------#*  *# -- 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:*  *load\_cd(file\_name, table): --> None*  *add\_cd(row, table): --> Confirmation message*  *"""*  *# TODone Add Code to the CD class*  *# -- Field -- #*  cd\_id = **None**  cd\_title = ''  cd\_artist = ''    *# -- Constructor -- #*  **def** \_\_init\_\_(self, cd\_id, cd\_title, cd\_artist):  *# -- Attributes -- #*  self.\_\_cd\_id = cd\_id  self.\_\_cd\_title = cd\_title  self.\_\_cd\_artist = cd\_artist    *# -- Properties -- #*  @property  **def** cd\_id(self,value):  **return** self.\_\_cd\_id  **def** cd\_title(self,value):  **return** self.\_\_cd\_title  **def** cd\_artist(self,value):  **return** self.\_\_cd\_artist  @cd\_id.setter  **def** cd\_id(self,value):  **if** str(value).isnan():  **raise** **Exception**('The value must a number')  @staticmethod  **def** load\_cd(file\_name, table):  *"""Function to process user request to load inventory from file*  *Confirms user choice before loading inventory data from runtime and deletes all entries in memory*  *Args:*  *file\_name (string): name of file used to write the data from*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*  *Returns:*  *None*  *"""*  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: ')  **if** strYesNo.lower() == 'yes':  print('reloading...')  table = FileIO.load\_inventory(file\_name)  IO.show\_inventory(table)  **else**:  input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')  IO.show\_inventory(table)  @staticmethod  **def** add\_cd(row, table):  *"""Function to manage data ingestion from user input to a list of dictionaries*  *Adds data from user entry into a 2D table (list of dicts) in memory during runtime.*  *Args:*  *data (list): values entered by user for ID, CD Title, Artist Name*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*  *Returns:*  *Confirmation message the CD was added to inventory.*  *"""*  table.append(row)  print('The CD was added to Inventory')  *# -- PROCESSING -- #*  **class** **FileIO**:  *"""Processes data to and from file:*  *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:*  *save\_inventory(file\_name, lst\_Inventory): -> None*  *load\_inventory(file\_name): -> (a list of CD objects)*  *"""*  *# -- Field -- #*  cd\_id = **None**  cd\_title = ''  cd\_artist = ''    *# -- Constructor -- #*  **def** \_\_init\_\_(self, cd\_id, cd\_title, cd\_artist):  *# -- Attributes -- #*  self.\_\_cd\_id = cd\_id  self.\_\_cd\_title = cd\_title  self.\_\_cd\_artist = cd\_artist    *# -- Properties -- #*  @property  **def** cd\_id(self,value):  **return** self.\_\_cd\_id  **def** cd\_title(self,value):  **return** self.\_\_cd\_title  **def** cd\_artist(self,value):  **return** self.\_\_cd\_artist  @cd\_id.setter  **def** cd\_id(self,value):  **if** str(value).isnan():  **raise** **Exception**('The value must a number')    *# TODone Add code to process data from a file*  @staticmethod  **def** load\_inventory(file\_name):  *"""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*  *Returns:*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*  *"""*  table = []  objFile = open(file\_name, 'r')  **for** line **in** objFile:  data = line.strip().split(',')  dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}  table.append(dicRow)  objFile.close()  **return** table  IO.show\_inventory(table)    *# TODone Add code to process data to a file*  @staticmethod  **def** save\_inventory(file\_name, table):  *# ToDONE Add code here*  *"""Function to manage data storage from a list of dictionaries to a file*  *Saves the data to file identified by file\_name from a 2D table*  *(list of lists).*  *Args:*  *file\_name (string): name of file used to write the data from*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*  *Returns:*  *None.*  *"""*  **while** **True**:  strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()  **if** strYesNo == 'y':  objFile = open(strFileName, 'w')  **for** row **in** table:  lstValues = list(row.values())  lstValues[0] = str(lstValues[0])  objFile.write(','.join(lstValues) + '**\n**')  objFile.close()  **break**  **elif** strYesNo == 'n':  input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')  **break**  **else**:  print('Incorrect choice!! Please try again.**\n**')  **continue**  *# -- PRESENTATION (Input/Output) -- #*  **class** **IO**:  *# TODone add docstring*  *"""Handling Input / Output*  *properties:*  *methods:*  *print\_menu(): -> None*  *menu\_choice(): --> (a lower case sting of the users input out of the choices)*  *show\_inventory(table): --> None*  *get\_user\_input(): --> (cd\_id, cd\_title, cd\_artist)*    *"""*    *# 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('**\n**Menu**\n\n**[l] Load Inventory from file**\n**[a] Add CD**\n**[i] Display Current Inventory')  print('[s] Save Inventory to file**\n**[x] Exit**\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 l, a, i, d, s or x*  *"""*  choice = input('Which operation would you like to perform? [l, a, i, s or x]: ').lower().strip()  **while** choice **not** **in** ['l', 'a', 'i', 's', 'x']:  print('Invalid choice. Please select one of the options listed.**\n**')  choice = input('Which operation would you like to perform? [l, a, i, s 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(table):  *"""Displays current inventory table*  *Args:*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.*  *Returns:*  *None.*  *"""*  print('**\n**======= The Current Inventory: =======')  print('ID**\t**CD Title (by: Artist)**\n**')  **for** row **in** table:  print('**{}\t{}** (by:**{}**)'.format(\*row.values()))  print('======================================')    *# TODone add code to get CD data from user*  @staticmethod  **def** get\_user\_input():  *""" Function to get the user input for adding a CD entry*  *The entry will be returned to be used by a DataProcessor function that will Add it to inventory.*  *Args:*  *None*    *Returns:*  *cd\_id (int): User supplied ID for entry*  *cd\_title (string): Title of CD*  *cd\_artist (string): Name of artist*  *"""*  **while** **True**:  cd\_id = input('**\n**Enter ID: ').strip()  **try**:  cd\_id = int(cd\_id)  **break**  **except** **ValueError** **as** e:  print('**\n**That is not a valid ID number. Please try again.')  print('**\n**Build in error info:', e.\_\_doc\_\_, sep='**\n**')  **except** **Exception** **as** e:  print('**\n**There was a general error!')  print('**\n**Build in error info:', e.\_\_doc\_\_, sep='**\n**')  cd\_title = input('What is the CD**\'**s title? ').strip()  cd\_artist = input('What is the Artist**\'**s name? ').strip()  cd = {'ID': cd\_id, 'Title': cd\_title, 'Artist': cd\_artist}  **return** cd    *# -- Main Body of Script -- #*  *# TODone Add Code to the main body*  *# Load data from file into a list of CD objects on script start*  lstOfCDObjects = FileIO.load\_inventory(strFileName)  *# Display menu to user*  **while** **True**:  IO.print\_menu()  strChoice = IO.menu\_choice()    *# show user current inventory*  **if** strChoice == 'i':  IO.show\_inventory(lstOfCDObjects)  **continue** *# start loop back at top.*    *# let user add data to the inventory*  **elif** strChoice == 'a':  CD.add\_cd(IO.get\_user\_input(), lstOfCDObjects)  IO.show\_inventory(lstOfCDObjects)  **continue** *# start loop back at top.*    *# let user save inventory to file*  **elif** strChoice == 's':  *# 3.6.1 Display current inventory and ask user for confirmation to save*  IO.show\_inventory(lstOfCDObjects)  FileIO.save\_inventory(strFileName, lstOfCDObjects)  **continue** *# start loop back at top*    *# let user load inventory from file*  **elif** strChoice == 'l':  CD.load\_cd(strFileName, lstOfCDObjects)  **continue** *# start loop back at top.*    *# let user exit program*  **elif** strChoice == 'x':  **break**  **else**:  print('Invalid choice. Please select one of the options listed.')  **continue** *# start loop back at top.* |

1. Last viewed: Dec-4, 2022 (11:29pt PT) [↑](#footnote-ref-1)