Evan Slodysko

04 December 2022

FDN110 – Foundations of Programing

Assignment 08

**Introduction**

The purpose of Assignment 08 was to introduce Object Oriented Programming. Within the Object Oriented Programming, key concepts covered included difference between a class and the object made from a class, components that make up a standard pattern of a class, the class constructor, the purpose and use of the key word *self* , using @staticmethod, when and why to use fields, attributes and properties, and the difference between property and methods.

**Topic 1**

This section describes the steps performed to complete Step 5 from this assignment. Step 5 asks to use the starter code provided and complete each prompt as defined by the pseudocode including

* Class CD:
  + Add Code to the CD class
* Class FileIO:
  + Add code to process data from a file
  + Add code to process data to a file
* Class IO:
  + Add docstring
  + Add code to show menu to user
  + Add code to capture users choice
  + Add code to display the current data on screen
  + Add code to get CD data from user
* Main Body of Script
  + Add code to main body
  + Load data form file into a list of CD objects on script start
  + Display menu to user (show current inventory, add data to inventory, save inventory to file, load inventory to file, exit program.

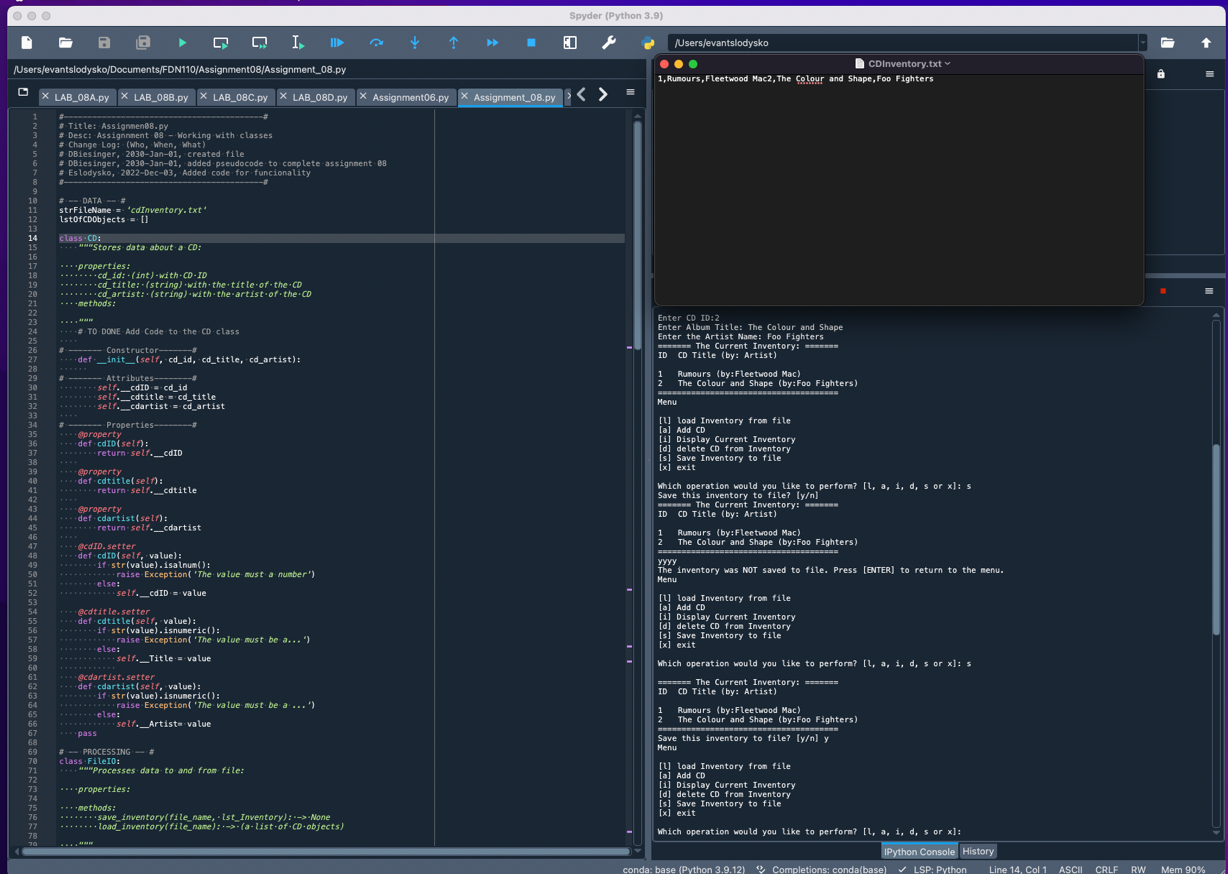
Additionally,

* The script should include some error handling
* The script must be saved in a specified course folder
* Run the script in Spyder and the Terminal and,
* Provide proof of the script running from both locations, and the CD Inventory.txt File created

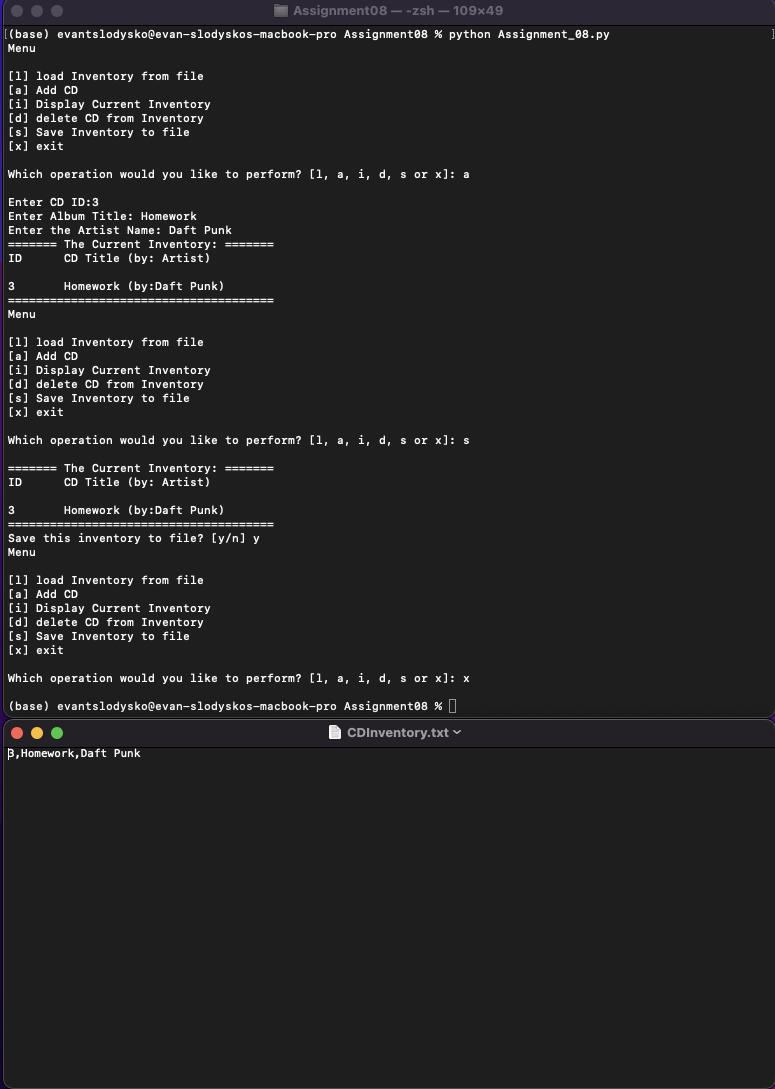
I started by reviewing the starter script and the previous weeks script to look for opportunities where script could be reused with minor modifications. I started with the menu as this is the body of the script and essentially a road map for each subsection of script that will need to be developed. Due to issues in with assignment 06/07 and time constraints portions of the menu are not functioning/present. The items below details key parts of the script.

* Class CD
  + Following the pseudocode prompt, code was added to the CD Class. This code is the ‘blueprint’ that packages the data for the cd inventory program (*cd\_id, cd\_title, cd\_artist*) and functionalities. Later in the script, the class is called to create multiple instantiations for each cd. The standard python constructor method \_\_init\_\_() was used and attributes defined. Following the script created for LAB08\_D, properties and setters were added. The intent of the 3 setters added (@cdID.setter @cdtitle.setter and @artist.setter) was to validate the data provided by the user and to provide some error handling. In retrospect, probably the only truly necessary setter / error handling is for the cdID as cd titles, and artist names could be integers.
  + Script was then added to the to the main body for adding an entry which calls for the class IO and the user inputs. The CD class constructor is called and an instantiation of the object (*CD)* is performed and assigned to a variable which is then appended to the list defined in the starter script.
  + Based on the original code, once an entry is added a print display of the current cd inventory is returned to the user. To maintain this functionality, code for the show\_inventory static method was added. The code was modified from handling lists/dictionaries to handling the object instantiated (Cd)
* Class IO (print menu, menu choice, user\_input, show\_inventory)
  + Within the Class IO several static methods were added as indicated by the pseudocode including print\_menu, user\_input, show\_input). The method type (static method) wasn’t changed from the previous week which seems reasonable as these methods are within the Class IO which handles data for the program.
* Class FileIO
  + Within the FileIO class, script from the previous week was re-used and modified to complete the actions of loading / saving cd objects to from/to an external txt file. Modifications were completed to reflect handling objects instead of lists/dictionaries. Due to time limitations, only the method for saving the file is functional.

The figures below show the program working



*Figure 1 – Program running with two entries saved to list created and saved to external text file*



*Figure 2 – Script running from terminal window and saving to external text file*

**Topic 2**

Once complete with creating a code per instructions, step 7 requires

* Upload files and commit changes
* Share link to Canvas discussion board
* Perform a peer review

A git hub account was created and a repository was created for assignment 07. The link below can be used to access the repository.

* [*https://github.com/eslodysko/Assignment\_08*](https://github.com/eslodysko/Assignment_08)

**Summary/Observations**

In this assignment we continued to expand on our working knowledge of computer programming basics by introducing and studying Object Oriented Programming. Concepts covered within Object Orient Programing included

* Difference between a class (blueprint) and the object made from a class,
* components that make up a standard pattern of a class,
* the class constructor,
* the purpose and use of the key word *self*
* using @staticmethod,
* when and why to use fields, attributes and properties, and the
* difference between property and methods.

I found Object Oriented Programming to be an abstract concept and it took reviewing lecture videos and working examples several times before it I could get a general understanding of its purpose and how to apply to solving this week’s homework assignment. The labs this week were manageable and items A-D were helpful for breaking down each component of classes and their purpose (constructors, properties, getters, and setters). As with other concepts in this course, there are subtle distinctions with classes (and its components) that I will have to review once again to confirm my understanding. In particular, understanding the advantages or when it is preferable to use @static method. Additionally, one the challenges I struggled with again was not having a complete week 6 code to work off. In most cases I was able to take what I started in week 6 and re-work for this week’s assignment but due to time limitations not all items of the code were completed.

**Appendix**

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | 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 | *#------------------------------------------#*  *# 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*  *# Eslodysko, 2022-Dec-03, Added code for funcionality*  *#------------------------------------------#*  *# -- 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:*  *"""*  *# TO DONE Add Code to the CD class*    *# ------- Constructor-------#*  **def** \_\_init\_\_(self, cd\_id, cd\_title, cd\_artist):    *# ------- Attributes--------#*  self.\_\_cdID = cd\_id  self.\_\_cdtitle = cd\_title  self.\_\_cdartist = cd\_artist    *# ------- Properties--------#*  @property  **def** cdID(self):  **return** self.\_\_cdID    @property  **def** cdtitle(self):  **return** self.\_\_cdtitle    @property  **def** cdartist(self):  **return** self.\_\_cdartist    @cdID.setter  **def** cdID(self, value):  **if** str(value).isalnum():  **raise** **Exception**('The value must a number')  **else**:  self.\_\_cdID = value  @cdtitle.setter  **def** cdtitle(self, value):  **if** str(value).isnumeric():  **raise** **Exception**('The value must be string type')  **else**:  self.\_\_Title = value    @cdartist.setter  **def** cdartist(self, value):  **if** str(value).isnumeric():  **raise** **Exception**('The value must be string type')  **else**:  self.\_\_Artist= value  **pass**  *# -- 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)*  *"""*  *# TODO Add code to process data from a file*  @staticmethod  **def** load\_inventory(file\_name,lst\_Inventory):  lstOfCDObjects.clear()  objFile= open(file\_name, 'r')  **for** cd **in** objFile:  data = cd.strip().split(',')  lstOfCDObjects.append(data)  objFile.close()      *# TO DONE Add code to process data to a file*  @staticmethod  **def** save\_inventory(file\_name, lst\_Inventory):  *"""Function for saving information to CDInventory.txt'"""*  objFile = open(file\_name, 'w')  **for** cd **in** lst\_Inventory:  objFile.write(str(cd.cdID) + ',' + cd.cdtitle + ',' + cd.cdartist)  objFile.close()  **pass**    **pass**  *# -- PRESENTATION (Input/Output) -- #*  **class** **IO**:  *# TO DONE add docstring*  *# TO DONE add code to show menu to user*  *"""Handling Input / Output"""*  @staticmethod  **def** print\_menu():  *"""Displays a menu of choices to the user*  *Args:*  *None.*  *Returns:*  *None.*  *"""*  print('Menu**\n\n**[l] load Inventory from file**\n**[a] Add CD**\n**[i] Display Current Inventory')  print('[d] delete CD from Inventory**\n**[s] Save Inventory to file**\n**[x] exit**\n**')  @staticmethod  **def** menu\_choice():  *"""Gets user input for menu selection*  *Args:*  *None.*  *Returns:*  *choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x*  *"""*  choice = ' '  **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:  choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()  print()  **return** choice      *# TO DONE add code to captures user's choice*  @staticmethod  **def** user\_input(): *# function for user input of CD*  *"""Function for getting user input for CDs (Cd Id, Album Name, Artist)*    *args:*  *None*    *Returns: CD ID (int), Album Name (Title) (str) and Artist (str) '"""*      intID= int(input('Enter CD ID:').strip())  strTitle= str(input('Enter Album Title: ')).strip()  strArtist = str(input('Enter the Artist Name: ')).strip()  **return**(intID,strTitle,strArtist)  *# TODO add code to display the current data on screen*  @staticmethod  **def** show\_inventory(table):  *"""Displays current inventory table*  *Args:*  *table (list of cds): that is defined by properties in class CD that holds the data during runtime.*  *Returns:*  *None.*  *"""*  print('======= The Current Inventory: =======')  print('ID**\t**CD Title (by: Artist)**\n**')  **for** cd **in** table:  print('**{}\t{}** (by:**{}**)'.format(cd.cdID,cd.cdtitle,cd.cdartist))  print('======================================')  *# TODO add code to get CD data from user*  **pass**  *# -- Main Body of Script -- #*  *# TODO Add Code to the main body*  *# Load data from file into a list of CD objects on script start*  *# 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*    *# 1. When program starts, read in the currently saved Inventory*  *#FileIO.read\_file(strFileName, lst\_Inventory)*  *# 2. start main loop*  **while** **True**:  *# 2.1 Display Menu to user and get choice*  IO.print\_menu()  strChoice = IO.menu\_choice()    *# 3. Process menu selection*  *# 3.1 process exit first*  **if** strChoice == 'x':  **break**    *# 3.2 process load inventory*  **if** 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')  **if** strYesNo.lower() == 'yes':  print('reloading...')    **else**:  input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')  **continue**  *# 3.3 process add a CD*  **elif** strChoice == 'a':  *# 3.3.1 Ask user for new ID, CD Title and Artist*  cdID, cdTitle, cdArtist = IO.user\_input()  cdInfo = CD(cdID, cdTitle, cdArtist) *#creating an instance of the object, assigning to variable*  lstOfCDObjects.append(cdInfo)  *# Call constructor, store returned cd object in variable, add to list of cd objects*  IO.show\_inventory(lstOfCDObjects)  *# 3.4 process display current inventory*  **elif** strChoice == 'i':  IO.show\_inventory(lstOfCDObjects)    *# 3.5 process delete a CD*  **elif** strChoice == 'd':  **continue**  *# 3.6 process save inventory to file*  **elif** strChoice == 's':    IO.show\_inventory(lstOfCDObjects)  strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()  *# 3.6.2 Process choice*  **if** strYesNo == 'y':  *# 3.6.2.1 save data*  *# TODO move processing code into function*  FileIO.save\_inventory(strFileName, lstOfCDObjects)  **else**:  input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')  **continue**  *# 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:*  **else**:  print('General Error') | |  |

*Script used for Assignment 08. The script Syntax was highlighted using Saravji’s Hut[[1]](#footnote-1)*

1. Saravji’s Hut – Accessed 2022 Dec-04 [↑](#footnote-ref-1)