Kenneth Chiu

March 20, 2022

Foundations of Programming: Python

Assignment 08

The Magic CD Inventory Program - Continued

# Introduction

In this module, we started diving into more details on Object Oriented Programming (“OOP”), which essentially means all components in a program are constructed as objects. We learned how to create our own customized classes and use constructors, attributes, property getters, setters and methods to process data and transform into meaningful information. Then, we implemented the custom classes throughout the Magic CD Inventory Program.

# What is a Class?

A class is like a blueprint or template which contains structural components of an object and it can contain some or all of the followings:

* Fields
* Constructor
* Attributes
* Properties
* Methods.

To create a new object, a class is instantiated with required attributes and to ensure proper data is provided when creating a new object, we can use the \_\_init\_\_() function to assign provided values to attributes accordingly. Additionally, to ensure the provided values are in the desired format, we can use property getters and setters to format and validate provided values to ensure the objects are with valid attributes. Note that in order to use the property functions, the attributes need to be set as private attributes by adding leading dunder or “\_\_” to each attribute. Finally, we can also create custom methods, such as the \_\_str\_\_() method, to perform just about anything you want the objects to do.

# The Magic CD Inventory program

GitHub link - https://github.com/kennchiu/Assignment\_08

To complete the Magic CD Inventory program, I loaded the Assignment\_08\_Starter.py file and updated the script header – List 1.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | *#------------------------------------------#*  *# Title: CDInventory.py*  *# Desc: CD Inventory Program for Assignment 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*  *# KChiu, 2022-Mar-20, added codes to complete assignment 08*  *#------------------------------------------#* |

List - Script Header

Next, I completed the CD() class by adding the required constructor, property setters and getters for the required attributes, cd\_id, cd\_title and cd artist – List 2. For cd\_id, I added property to ensure the provided ID number is numeric and can be converted to an integer and for cd\_title and cd\_artist, the respective property will make sure these two attributes cannot be blank. Additionally, I set the property for cd\_title and cd\_artist to be in title case for the CD Inventory program for consistency. Finally, I also added the print\_screen() and write\_file() functions to the CD class but I will come back to these later.

|  |  |
| --- | --- |
| 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 | **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:*  *\_\_int\_\_(cd\_id, cd\_title, cd\_artist): -> CD object*  *print\_screen(): -> string in screen display format*  *save\_file(): -> string in csv format*  *"""*  *# TODone Add Code to the CD class*  *#--Fields--#*  *#--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):  **return** self.\_\_cd\_id  @cd\_id.setter  **def** cd\_id(self, value):  **if** type(value) == str:  **try**:  self.\_\_cd\_id = int(value)  **except**:  **raise** **Exception**('Excuse Me? CD ID should be numeric!')  **elif** type(value) == int:  self.\_\_cd\_id = value  **else**:  **raise** **Exception**('Excuse Me? CD ID should be numeric!')  @property  **def** cd\_title(self):  **return** self.\_\_cd\_title.title()  @cd\_title.setter  **def** cd\_title(self, value):  **if** value == '':  **raise** **Exception**('Excuse Me? CD title can**\'**t be blank!')  **else**:  self.\_\_cd\_title = value  @property  **def** cd\_artist(self):  **return** self.\_\_cd\_artist.title()  @cd\_artist.setter  **def** cd\_artist(self, value):  **if** value == '':  **raise** **Exception**('Excuse Me? CD artist can**\'**t be blank!')  **else**:  self.\_\_cd\_artist = value  *#--Methods--#*  **def** print\_screen(self):  cd = '**{}\t{}** (by: **{}**)'.format(self.\_\_cd\_id, self.\_\_cd\_title, self.\_\_cd\_artist)  **return** cd  **def** wrtie\_file(self):  cd = '**{}**,**{}**,**{}**'.format(self.\_\_cd\_id, self.\_\_cd\_title, self.\_\_cd\_artist)  **return** cd |

List - class CD()

Next, following the pseudocodes, I added the load\_inventory() and save\_inventory() methods to the FileIO() class. For the load\_inventory() method, it clears the lstOfCDObjects list first then loops through all the rows in the CDInventory.txt file and create a new object for each CD by instantiating the CD() class, then each CD object is appended to the lstOfCDObjects list. For the save\_inventory() method, it loops through all the CD objects in the lstOfCDObjects list and write out the CD data to the CDInventory.txt. file – List 3. To write out the CD data in the csv format, I created a write\_file() method in the CD() class (List 2), which creates a string in the “CD ID,CD Title,CD Artist” format for each CD object and this method is called when each CD object is being written to the output file.

|  |  |
| --- | --- |
| 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 | *# -- PROCESSING -- #*  **class** **FileIO**():  *"""Processes data to and from file:*  *properties:*  *None.*  *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** load\_inventory(file\_name):  *"""Function to manage data ingestion from file to a list of objects*  *Reads the data from file identified by file\_name and use CD class to*  *convert data into a list of objects*  *Args:*  *file\_name (string): name of file used to read the data from*  *Returns:*  *None.*  *"""*  *# clear the CD table before loading from the inventory file.*  lstOfCDObjects.clear()  *# read data from file and use CD class to convert each CD into an object.*  *# append CD object to the lstOfCDObjects list.*  **try**:  **with** open(file\_name, 'r') **as** objFile:  **for** row **in** objFile:  obj = row.strip().split(',')  obj = CD(obj[0], obj[1], obj[2])  lstOfCDObjects.append(obj)  **except** **FileNotFoundError**:*# added error handling if CDInventory file doesn't exist.*  print('**\n**FileNotFoundError: ' + file\_name +' File does not exist!')  *# TODone Add code to process data to a file*  @staticmethod  **def** save\_inventory(file\_name, lst\_Inventory):  *# TODone Add code here*  *"""Function to sync the data in memory to file by saving current table in csv format*  *Writes CD data in the lstOfCDObjects in memory to the CDInventory.txt file.*  *Each CD object in list uses the wrtie\_file() function in CD class to create*  *csv format data to be saved in the file.*  *Args:*  *file\_name (string): name of file used to write/save the data to*  *table (list of objects): list that holds the current CD objects during runtime*  *Returns:*  *None.*  *"""*  *# save data*  **try**:  **with** open(file\_name, 'w') **as** objFile:  **for** obj **in** lst\_Inventory:  objFile.write(obj.wrtie\_file() + '**\n**')  **except** **Exception**:*# added error handling if error arrise during writing data to file.*  print('Something went wrong, please check the CDInventory file.') |

List - class FileIO()

Next, in the IO() class, I added print\_menu(), menu\_choice(), show\_inventory() and add\_cd() methods according to the pseudocodes. Most of the codes in this class were copied over from prior assignments with changes necessary to work with the CD() class. For the print\_menu() method, I removed the “[d] Delete CD from Inventory” option since it’s not required by the assignment. For the show\_inventory() method, it loops through the CD objects in the lstOfCDObjects list and calls the print\_screen() method in the CD() class (List 2) to display each CD in a “CD ID, CD Title (by: CD Artist)” string format. Finally the add\_cd() method, takes the user provided CD ID, CD Title and CD Artist inputs and creates a new CD object by calling the CD() class and appends to the lstOfCDObjects list. The add\_cd() method also validates the user inputs to make sure the CD ID is numeric and can be converted to an integer and both CD Title and CD Artist are not blank – List 4.

|  |  |
| --- | --- |
| 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 | *# -- PRESENTATION (Input/Output) -- #*  **class** **IO**():  *# TOoneO add docstring*  *"""Handling Input / Output*  *properties:*  *None.*  *methods:*  *print\_menu(): -> display CD menu*  *menu\_choice(): -> user choice (string)*  *show\_inventory(list of inventory): -> string of CD data in a 2D table*  *add\_cd(cd\_id = int, cd\_title = str, cd\_artist = str): -> CD objects*  *"""*  *# 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**[l] Load Inventory from file**\n**[a] Add CD')  print('[i] Display Current Inventory**\n**[s] Save Inventory to file**\n**[x] Exit')  *# 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 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()  **if** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:  print('Please enter a valid option!')  print() *# Add extra space for layout*  **return** choice  *# TODone add code to display the current data on screen*  @staticmethod  **def** show\_inventory(cd\_list):  *"""Displays current inventory table*  *Args:*  *table (list of object): that holds the data during runtime.*  *Returns:*  *None.*  *"""*  print('======= The Current Inventory: =======')  print('ID**\t**CD Title (by: Artist)**\n**')  **for** obj **in** cd\_list:  print(obj.print\_screen())  print('======================================')  *# TODone add code to get CD data from user*  @staticmethod  **def** add\_cd(cd\_id, cd\_title, cd\_artist):  *"""Function to add a new cd to lstOfCDObjects table if user chooses to*  *Use the CD ID, Title and Artist Name of the new CD to create a CD new object*  *and append to the lstOfCDObjects table*  *Args:*  *cd\_id (integer): ID number of the new CD*  *cd\_title (string): Title of the new CD*  *cd\_artist (string): Arist name of the new CD*  *Returns:*  *None.*  *"""*  *# create a new object for new cd*  obj = CD(cd\_id, cd\_title, cd\_artist)  **try**:  *# use CD class properties to ensure CD ID is an integer*  *# and CD title and artist are not blank*  obj.cd\_id = obj.cd\_id  obj.cd\_title = obj.cd\_title  obj.cd\_artist = obj.cd\_artist  *# append new CD object to lstOfCDObjects list*  lstOfCDObjects.append(obj)  **except** **Exception** **as** e:  print(e) |

List - class IO()

Finally, I copied in the main body of the script from prior assignments and updated the references to the CD objects list, lstOfCDObjects, wherever necessary.

|  |  |
| --- | --- |
| 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 | *# -- Main Body of Script -- #*  *# TODone Add Code to the main body*  *# 1. When program starts, read in the currently saved Inventory*  FileIO.load\_inventory(strFileName)  *# 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...')  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.*  *# 3.3 process add a CD*  **elif** strChoice == 'a':  *# 3.3.1 Ask user for new ID, CD Title and Artist*  strID = input('Enter ID: ').strip()  strTitle = input('What is the CD**\'**s title? ').strip()  strArtist = input('What is the Artist**\'**s name? ').strip()  IO.add\_cd(strID, strTitle, strArtist)  IO.show\_inventory(lstOfCDObjects)  **continue** *# start loop back at top.*  *# 3.4 process display current inventory*  **elif** strChoice == 'i':  IO.show\_inventory(lstOfCDObjects)  **continue** *# start loop back at top.*  **elif** strChoice == 's':  *# 3.5.1 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()  *# 3.5.2 Process choice*  **if** strYesNo == 'y':  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.*  *# 3.6 catch-all should not be possible, as user choice gets vetted in IO, but to be save:*  **else**:  print('General Error') |

List - Main Body of Script

# Summary

The complexity and confusion of this module dwarfs all the lessons and materials we’ve gone through so far. There were many more new concepts introduced all at once which definitely made my head spin for a while. However, after going through the materials, asking too many questions during the office hours and completing the assignment, I think I was finally able to connect the dots and have a better understanding on how to create a custom class and incorporate it into other functions and classes throughout the CD Inventory program.

Everything below is required screenshots per assignment instructions.

Below are screenshots showing the CD Inventory program working on Spyder – Figure 1 through 10.

Text

Description automatically generated

Figure - Menu with FileNotFoundError Message

Text

Description automatically generated

Figure - Add CD 1

Text

Description automatically generated

Figure - Add CD 2

Text

Description automatically generated

Figure - Display Current Inventory

Text

Description automatically generated

Figure - Save CD Inventory to CDInventory.txt File

Text

Description automatically generated

Figure - Exit Program

Text

Description automatically generated

Figure - CD ID Type Error

Text

Description automatically generated

Figure - CD Title Blank Error

Text

Description automatically generated

Figure - CD Artist Blank Error

Graphical user interface, text, application

Description automatically generated

Figure - CDInventory File after Running through Spyder

Below screenshots show Magic CD Inventory program run in the Terminal – Figure 11 through 14.

Text

Description automatically generated

Figure - CD Inventory Program via Terminal - Part 1

Text

Description automatically generated

Figure - CD Inventory Program via Terminal - Part 2

Text

Description automatically generated

Figure - CD Inventory Program via Terminal - Part 3

Graphical user interface, text, application

Description automatically generated

Figure - CDInventory File after Running through Terminal