Miroslava Meza

03/20/2022

IT FDN 110B Wi 22: Foundations of Programming Python

Assignment 08, Module 08

# Object Oriented Programming (OOP). Classes and Objects, Constructors, Fields, Attributes and Methods.

## Introduction

This assignment has the goal of learning how to define and use Classes and Objects, Constructors, Fields, Attributes and Methods.

## Github repository

This assignment is stored in two different places, (1) https://github.com/Mirka3m-Sea/Assignment\_08.git and (2) CANVAS.

## Defining Objects and classes

### Constructors

Constructors are a specialized method. They are run once during creation of the object. They are limited to this one purpose and hence are implicitly called when creating an object with a function call like syntax.

Figure 1. Constructor. Objects and classes.

Text

Description automatically generated

**Fields:** These are added to the class/object. They are where data is stored for a class, the syntax for reading or referring to a field is: objectName.fieldName. For example,

**Attributes** are internal fields or variables that hold data.

**Setting up fields**: This assignment had two classes – CD, File IO. Each of these classes stored different data in different internal fields.

First challenge: The “***class CD”*** creates an object CD that has three instance attributes – id, artist, title. Originally, I tried to set those three variables as fields (see Figure 1. Setting attributes for the "class CD") but it only returned an error.

Solution: did not set those variables in the field section and defined them within the constructor.

Second challenge: learning the difference between the function name (property) and the returned field. I spent a long time trying to align the return of each of the functions below.

Figure 2. Attributes/properties.

Text

Description automatically generated

### Methods

In this assignment I learned that methods are like functions. These methods help to organize statements into blocks that can be called out.

The difference between the functions we used in previous modules and methods is the way it is called in. These methods are part of an object, and their first attribute is the “self” reference.

Using decorators.

Static Methods:

For saving and loading the inventories, I used static-methods. The figures below show the final version of the functions for saving and loading the inventory.

Saving inventory into a file:

I encountered multiple issues while creating the functions to save inventory. I started with an OOP approach, that after multiple attempts and rewriting of the block was not successful (see Figure 5. Failed attempt to use OOP for loading and saving the file.). Finally, I opted to return to the original method used in assignment 6.

Figure 3. Saving inventory into a file.

*Text

Description automatically generated*

Loading inventory into a file:

For this function I encountered same challenge than for saving inventory into a file. I was not able to use OOP functions.

Figure 4. Loading inventory from an existing file.

*A screenshot of a computer

Description automatically generated with medium confidence*

Description of failed attempt of using OOP for saving and loading from a file.

Challenge, I tried to use an object-oriented approach to opening and saving, but I was not able to run it without errors.

The following is a version of the failed attempt to use OOP for loading and saving. I will be interested in learning how to implement this type of operation in the future.

Figure 5. Failed attempt to use OOP for loading and saving the file.

*Text

Description automatically generated*

## Running the Script in Spider

The first challenge I faced was that the script would yield an error message if I did not have a .txt file on the same repository. I eliminated that issue by opening/closing or creating/closing the txt file at the beginning of the script. That way, regardless of the sequence chosen by the user there will always be a destination file.

Testing sequence.

1. Started using the option[l] from the menu, loading data from file
2. [d] display inventory
3. [a] adding one more inventory entry.
4. [s] to save the information to the binary-data external file.
5. [x] to close the script.

.

Figure 6. Load from file and display data.

Text

Description automatically generated

Figure 7. Adding entries and saving into an external file.

Text

Description automatically generated

Figure 8. Exit

Text

Description automatically generated

## Running the Script in Terminal

Testing sequence.

1. Started using the option[l] from the menu, loading data from file
2. [d] display inventory
3. [a] adding one more inventory entry.
4. [s] to save the information to the binary-data external file.
5. [x] to close the script.

Figure 9. Load from file and add new entries to the inventory. Text

Description automatically generated

Figure 10. Display inventory, add new item, save data into file.

Text

Description automatically generated

Text

Description automatically generatedFigure 11. Exit

## Conclusions

This assignment required to modify two previous assignments (6 &7) by structuring the scripts as OOP. In this homework we continued to use classes, but this time they were part of an object.

The object created for this assignment was CD. This is a collection of classes and functions arranged as fields, constructors, attributes, and methods. Future challenge, learn how to use OOP to save and load files.

## Appendix Full Script

|  |  |
| --- | --- |
| 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 | *#------------------------------------------#*  *# 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*  *# Miroslava M., 2022-Mar-19. added code for class CD, FileIO*  *#Miroslava, 2022-Mar-20 fixed all the errors encoutered in the script*  *#------------------------------------------#*  *# -- 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:*    *(1) For class CD, added the constructor with instance attributes : cd\_id, cd\_tittle, and cd\_artist*  *(2) Define properties*  *"""*  *#--- Fields ---#*  *"""*  *Assignment 08*  *Adding the fields for the class CD*  *the three data depositories needed are:*  *Parameters:*  *ID: (Int) variable, this is the identifying number for each entry.*  *title: (str) variable. CD album name.*  *artist: (str) variable, artist name.*  *"""*    *# TO DONE Add Code to the CD class*  *#----- Constructor -----#*    *"""*  *Constructor to define the attributes for the Object CD.*  *#def \_\_init\_\_(self, cd\_id, cd\_title, cd\_artist):*  *"""*  **def** \_\_init\_\_(self, strID, title, artist):  self.\_\_id= strID  self.\_\_title = title  self.\_\_artist = artist  *#---Properties ----#*  *""""These properties will be used later as attributes. The returns*  *of each function match the values defined in the \_\_init\_\_ function.*  *"""*  @property  **def** cd\_id(self):  *#will handle input errors in I/O*  *""""*  *Parameters:*  *\_\_cd\_id: (object attribute)*  *Returns: self.\_\_id*  *"""*  **return** self.\_\_id  @property  **def** cd\_title(self):  *""""*  *Parameters:*  *cd\_title: (object attribute)*  *Returns: self.\_\_title*  *"""*  **return** self.\_\_title  @property  **def** cd\_artist(self):  *""""*  *Parameters:*  *\_\_cd\_artist : (object attribute)*  *Returns: self.\_\_artist*  *"""*  **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)*  *"""*  *#----------Fields ----------#*  *"""*  *Assignment 08*  *Added the fields to store data within this class.*  *lstOfCDObjects: (list of objects)*  *lst\_Inventory: (list)*  *"""*    *# TO DONE Add code to process data from a file*  *#----------Reading the file into a table of objects -------------}*  @staticmethod  **def** save\_inventory(file\_name, lst\_Inventory):  *"""staticmethod function, this is not an object*    *arguments:*  *file\_name (str): file name that will be opened*  *lst\_Inventory (list): this will be a list of objects*  *objFile (obj): object created to open and close a file*  *lstValues (list): temporary list values to be save into a external file*  *return:*  *None*  *"""*  objFile= open(file\_name, 'w')  **for** row **in** lst\_Inventory:  lstValues =[]  lstValues.extend([row.cd\_id, row.cd\_title, row.cd\_artist])  lstValues[0] = str(lstValues[0])  objFile.write(','.join(lstValues) + '**\n**')  objFile.close()      @staticmethod  **def** load\_inventory(file\_name):  *"""*  *staticmethod function, this is not an object*  *arguments:*  *file\_name (str): file name that will be opened*  *lstOfCDObjects (list): this a list of objects stored in temporary memory*  *objFile (obj): object created to open and close a file*  *data (list): list values read from a file to be loaded into a temporary table (lstOfCDObjects)*  *dicRow (CD object): object with the attributes of function CD.*  *return:*  *None*  *"""*  lstOfCDObjects.clear()  objFile= open(file\_name, 'r')  **for** line **in** objFile:  data = line.strip().split(',')  dicRow = CD(data[0], data[1], data[2])  lstOfCDObjects.append(dicRow) *# I had the wrong brackets and did not load.*  objFile.close()  *#*  *#------------- writing ------------------#*  *#TO DONE find the way to call the previous definitions.*  *# -- PRESENTATION (Input/Output) -- #*  **class** **IO**:  *# TO DONE add docstring*  *"""Handling Input / Output"""*    *# TO DONE 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 entry**\n**[d] Display Current Inventory')  print('[s] Save Inventory to file**\n**[x] Exit**\n**')  *# TO DONE 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', 'd', 's', 'x']:  choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()  print() *# Add extra space for layout*  **return** choice  *# TO DONE add code to display the current data on screen*  @staticmethod  **def** show\_inventory(lstOfCDObjects):  *"""Displays current inventory table*  *Args:lstOfCDObjects (list of objects) holds the data temporarily while the script is running.*  *Returns: None.*  *"""*  print('======= The Current Inventory: =======')  print('ID**\t**CD Title (by: Artist)**\n**')  **for** row **in** lstOfCDObjects:  print('**{}\t{}** (by:**{}**)'.format(row.cd\_id,row.cd\_artist, row.cd\_title))  print('======================================')    *# TO DONE add code to get CD data from user*  **def** data\_input() :  *"""*  *This function request the user to input data for each CD.*  *Arg:*  *ID, intID, title, artist. All of these entries are held*  *in the internal memory of this function. They are no global variables.*    *Returns: new\_entry (CD object), entry for CD witht attributes (ID, tittle, artist)*  *"""*  **while** **True**:  strID = input('Enter ID: ').strip()  **try**:  intID = int(strID)  **break**  **except** **ValueError**:  print('Invalid ID entry**\t**')  title = input('What is the CD**\'**s title? ').strip()  artist = input('What is the Artist**\'**s name? ').strip()  new\_entry= CD(strID, title, artist)  **return** new\_entry  *# -- Main Body of Script -- #*  *# TO DOING Add Code to the main body*  *# print('There isn\'t a file yet, please choose [a] or [s] first\n')*  file\_name2=open('cdInventory.txt', 'a') *#@Laura: I fixed this as you recommended, thanks!.*  file\_name2.close()  *# Display menu to user*  **while** **True**:  IO.print\_menu()  strChoice = IO.menu\_choice()    *# Start assessing choices.*  *#first choice is to exit.*  **if** strChoice == 'x':  **break**  *# show user current inventory*  **elif** strChoice == 'd':  IO.show\_inventory(lstOfCDObjects)  **continue**    *# let user add data to the inventory*  **elif** strChoice == 'a':  data1= IO.data\_input() *#data1 is a temporary variable, only used for appending the table*  lstOfCDObjects.append(data1)  IO.show\_inventory(lstOfCDObjects)  **continue** *# start loop back at top.*    *# let user save inventory to file*  **elif** strChoice == 's':  strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()  **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.*    *# 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**\t**')  **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)  **else**:  print('**\n**That is not a valid option, please select from the menu**\n**')  **continue** |

## Appendix II. Sources reviewed for this assignment

[IT FDN 110 WI 2022 Fri Office Hours - Zoom](https://washington.zoom.us/rec/play/serlOch-_pTDEklY4hvc-iLohKf6Kg8trYEacbGvYqYz59aaQirN35BEMXpAKF8iyZvIt20YKUK2w0Qm.SuZAqlSEfHtwepvt?continueMode=true)

[About David Amos – Real Python](https://realpython.com/team/damos/)

[Reading and Writing Files in Python (Guide) – Real Python](https://realpython.com/read-write-files-python/#appending-to-a-file)

[Object-Oriented Programming (OOP) in Python 3 – Real Python](https://realpython.com/python3-object-oriented-programming/#class-and-instance-attributes)