Amanda Larson

December 5, 2021

IT FDN 110 – Foundations of Programming: Python

Assignment 08

https://github.com/Amanda-Larson/Assignment_08

Module 08 – Assignment 08

Introduction

Module 08 / Assignment 08 focused on learning about object-oriented programing (OOP). Labs A-E walked through a step-wise process to adding in each new component of the object class including Constructors, Attributes, Properties, and Methods. The Assignment itself was to take a starter script with pseudocode and TODOs and recreate the CD Inventory program that we've worked on for the last several weeks of this course, while using OOP. The goal was to create the same program but to have the CDs be objects and to store them in a list of objects rather than a list of lists or a list of dictionaries.

What went well

A lot of this week's assignment was reconstructing the program from the labs from this week as well as previous week's assignments: a Frankenstein of code from different python programs we've created. My code included the OOP code from this week's lab and then the error handling from week 7 and the file processing from week 6. The first step was to copy/paste code blocks (entire functions) into classes that made sense for this week's program. I made sure to keep the file handling separate from the IO separate from the processing. I kept the Object creation/handling in its own class.

As I started to piece together this program, I realized that there were different naming conventions throughout the weeks and the labs as well. So, there was a fair amount of time spent changing all the variables to be consistent throughout the program. I also had to update the file handling class because we were no longer using a list of dictionaries. I was surprised to find that you can trat an object similarly to a list and access its attributes via an index. That was super convenient. It made sense, though, and I'm really starting to understand and even predict the python logic which makes the assignments a lot easier. It seemed almost too easy, so to make sure that I didn't somehow exclude the object creation code and just use a list of lists, I went back to test. I remembered from Friday's lectures that you don't have to name the objects, but then it makes it hard to reference them. There was a good example of how to print out the objects even if you haven't named them from Friday's lecture so I adopted that into my main body to see if I could get it to print out the same number of objects as rows in my inventory… just to make sure. It worked well and I was able to see a matching number of objects as rows in my inventory. It was very satisfying to know that I understood that material enough to spend only a few hours on it and was successful.

```
elif strChoice == 'i':
    IO.show_inventory(lstOfCDObjects)
    #test for objects - for testing only
    #for cds in lstOfCDObjects:
    #    objcd = CD(cds[0], cds[1], cds[2])
    #    print(objcd)
```

Figure 1 - A commented out version of the test I used to prove that CD objects were being created.

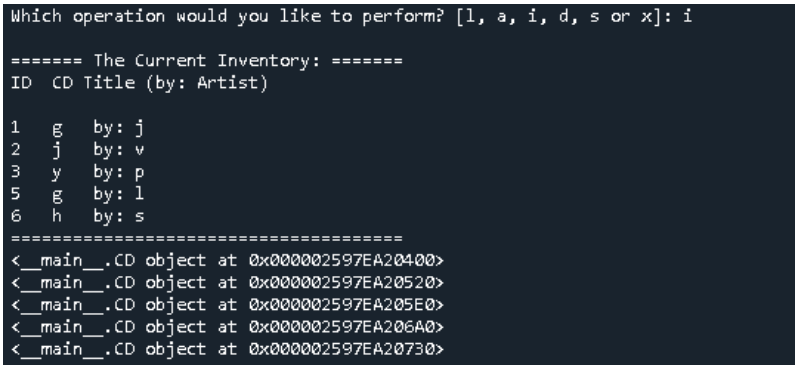

Figure 2- The results of the test I created to prove that CD objects were being created. There are as many objects printed as there are rows in the inventory list.

What didn't go well

Honestly, this was one of the easier assignments for me. I'm not too confused yet about the OOP. I don't totally get what the properties and setters are doing and if they're even necessary in this program. But I was able to figure out how to connect all the pieces of the Frankenstein program pretty quickly between changing variables and replacing dictionaries with objects. For me, it doesn't seem like you need to "get" objects so much as you need to follow the object style formatting and special dunder methods. Its like a recipe: only use @staticmethod when you want it to apply to everything - not one particular instance, use special object-related dunder methods, use a combination of fields, constructors, attributes, properties setters, and methods. I was able to help other students this weekend and describe why you need or don't need certain parts of code – which helps me to solidify my understanding.

Summary

I was pretty worried about how long this assignment would take and the general concepts seemed daunting. But, I think (hope) I learned enough to bring the week 08 program together quicker than I had been able to in previous weeks. I feel good about how far I've come from having almost no exposure to any type of programing aside from a few very basic sql queries for filtering databases. I still feel like I must be missing something. Since most people don't get it the first time and most people struggle with this more advanced concept, I am unconvinced that I actually grasped the concepts. Writing a test in my program seemed to prove that I was creating a list of objects, but I wouldn't be too surprised if when this is graded, I somehow evaded the use of objects and really just used a list of lists. I am interested in learning more about objects and how they can be used to better a program. I'm not sure that a list of objects is necessarily better or easier than a list of lists. I'm not sure I would know when to use objects versus lists or tuples or dictionaries. I don't know that I'd be comfortable creating my own from-scratch program using objects – I leaned heavily on the examples this week.

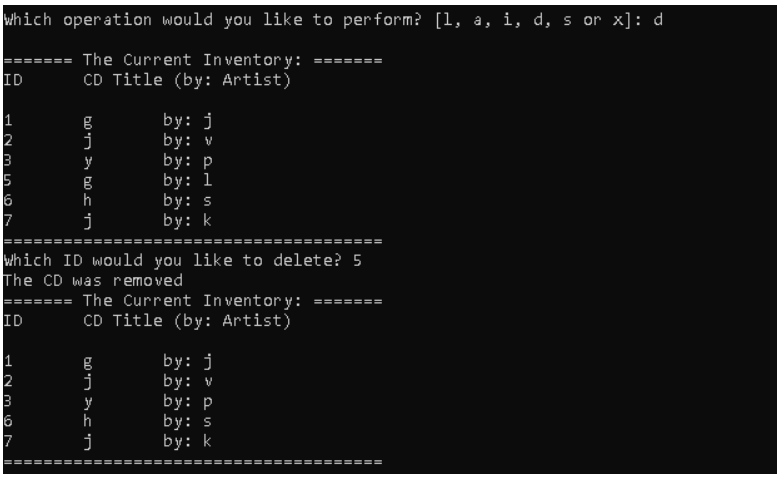

Figure 3 - Deleting a CD object by running the program in anaconda prompt window.

Appendix

I used a site called (Highlight your source code, 2021)[1] to format the code into the colored version. See below:

```
#------------------------------------------#
# 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
# ALarson, 2021-Dec-04, added code for OOP functionality, completed all TODOs
# ALarson, 2021-Dec-05, cleaned up notes, docstrings, and historic remnants and
#                       tested for OOP functionality
#------------------------------------------#


import sys


# -- 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:

    """

    #---Fields---#
    cdID = None
    cdArtist = ''
    cdTitle = ''

    #---Constructor---#
    def __init__(self, idnum, title, artist):
        #---Attributes---#
        self.__cdID = idnum
        self.__cdTitle = title
        self.__cdArtist = artist
```

[1] https://highlight.hohli.com/index.php

```
42.
43.     #cdTitle
44.     @property
45.     def cdTitle(self):
46.         return self.__cdTitle
47.
48.     @cdTitle.setter
49.     def cdTitle(self, title):
50.         if str(title).isnumeric():
51.             raise Exception('Title must be a string')
52.         else:
53.             return self.__title
54.
55.     #cdArtist
56.     @property
57.     def cdArtist(self):
58.         return self.__cdArtist
59.
60.     @cdArtist.setter
61.     def cdArtist(self, artist):
62.         if str(artist).isnumeric():
63.             raise Exception('Artist must be a string')
64.         else:
65.             return self.__artist
66.
67.
68.
69.
70. # -- PROCESSING -- #
71.
72.
73. class DataProcessor:
74.     """The DataProcessor class processes the user data.
75.
76.     properties:
77.
78.     methods:
79.         add_CD(idnum, title, artist) -> None
80.         delete_Row(intIDDel) -> None
81.         """
82.     @staticmethod
83.     def add_CD(idnum, title, artist):
84.         """Function to add a new user-defined CD to the table in-memory.
85.
86.         Args:
87.             intID: the CD ID number as provided by the user
88.             strTitle: the CD title as provided by the user
89.             stArtist: the CD artist as provided by the user
90.
91.         Returns:
```

```
            None.
        """

        cdLst = [idnum, title, artist]
        lstOfCDObjects.append(cdLst)


    @staticmethod
        def delete_Row(intIDDel):
            """ Function to delete row in table.

            Args:
                intIDDel: the ID of the CD to delete

            Returns:
                None.

            """
            intRowNr = -1
            blnCDRemoved = False
            for row in lstOfCDObjects:
                intRowNr += 1
                if row[0] == intIDDel:
                    del lstOfCDObjects[intRowNr]
                    blnCDRemoved = True
                    break
            if blnCDRemoved:
                print('The CD was removed')
            else:
                print('Could not find this CD!')


    class FileProcessor:
        """Processing the data to and from text file

        properties:

        methods:
            write_file(file_name, table): -> None
            read_file(file_name, table): -> (a list of CD objects)

        """

        @staticmethod
        def read_file(file_name, table):
            """Function to manage data ingestion from file to a list of objects

            Reads the data from file identified by file_name into a 2D table
            (list of objects) one line in the file represents one object row in table.

```

```
142.             Args:
143.                 file_name (string): name of file used to read the data from
144.                 table (list of objects): lstOfCDObjects = 2D data structure (list of
    objects) that holds the data during runtime
145.
146.             Returns:
147.                 None.
148.             """
149.
150.             table.clear()  # this clears existing data and allows to load data from file
151.
152.             try:
153.                 objFile = open(file_name, 'r')
154.                 for line in objFile:
155.                     data = line.strip().split(',')
156.                     row = [int(data[0]), data[1], data[2]]
157.                     table.append(row)
158.                 objFile.close()
159.                 return table
160.             except IOError:
161.                 print('Unable to open the file', file_name,'.'.strip(), 'Ending
    program.\n')
162.                 input('\n Press Enter to exit.')
163.                 sys.exit()
164.             except:
165.                 print('Unknown Error')
166.             finally:
167.                 objFile.close()
168.
169.
170.         @staticmethod
171.         def write_file(file_name, table):
172.             """Function to manage the saving of data to a text file.
173.
174.             Saves/writes the data into a file identified by file_name from a 2D list of
    objects.
175.
176.
177.             Args:
178.                 file_name (string): name of file to be saved, presumably the same as has
    already been loaded in.
179.                 table (list of objects): lstOfCDObjects = 2D data structure (list of
    objects) that holds the data during runtime
180.
181.             Returns:
182.                 None.
183.                 """
184.             try:
185.                 objFile = open(strFileName, 'w')
186.                 for eachobj in lstOfCDObjects:
```

```
                    eachobj[0] = str(eachobj[0])
                    objFile.write(','.join(eachobj) + '\n')
                objFile.close()
            except IOError:
                print('Unable to open/write to file.')
            finally:
                objFile.close()


    # -- PRESENTATION (Input/Output) -- #

    class IO:

        """Handling Input / Output

        properties:


        methods:
            print_menu() -> None
            menu_choice() -> None
            show_inventory -> None
            get_CD() -> a tuple of the user's input: strID, strTitle, stArtist

        """


        @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.

```

```
236.             Returns:
237.                 choice (string): a lower case sting of the users input out of the choices
    l, a, i, d, s or x
238.
239.             """
240.             try:
241.                 choice = ' '
242.                 while choice not in ['l', 'a', 'i', 'd', 's', 'x']:
243.                     choice = input('Which operation would you like to perform? [l, a, i,
    d, s or x]: ').lower().strip()
244.                 print()  # Add extra space for layout
245.                 return choice
246.             except TypeError:
247.                 print('Numbers aren\'t allowed, please choose a letter from the menu
    choices above.')
248.
249.
250.         @staticmethod
251.         def show_inventory(table):
252.             """Displays current inventory table
253.
254.
255.             Args:
256.                 table (list of dict): 2D data structure (list of dicts) that holds the
    data during runtime.
257.
258.             Returns:
259.                 None.
260.
261.             """
262.             print('======= The Current Inventory: =======')
263.             print('ID\tCD Title (by: Artist)\n')
264.             for eachCD in table:
265.                 print(str(eachCD[0]) + '\t' + eachCD[1] + '\t' + 'by: ' + eachCD[2])
266.             print('======================================')
267.
268.
269.         @staticmethod
270.         def get_CD():
271.             """Function to get user input/information for the CD ID number, the CD title,
    and the CD artist.
272.
273.             Args: None
274.
275.             Returns:
276.                 a tuple of the user's input: strID, strTitle, stArtist
277.
278.             """
279.
280.             while True:
```

```
281.                 try:
282.                     idnum = int(input('Enter ID: ').strip())
283.                     strID = str(idnum)
284.                     break
285.                 except ValueError:
286.                     print('The ID must be a number. Please enter a numeric ID.')
287.
288.
289.
290.             title = input('What is the CD\'s title? ').strip()
291.             artist = input('What is the Artist\'s name? ').strip()
292.             return strID, title, artist #returns a tuple
293.
294.
295.
296.     # -- Main Body of Script -- #
297.
298.     # 1.0 Load data from file into a list of CD objects on script start
299.     print() #for spacing
300.     print('Welcome to the CD Inventory Program \n')
301.     lstOfCDObjects = FileProcessor.read_file(strFileName, lstOfCDObjects)
302.     IO.show_inventory(lstOfCDObjects)
303.
304.     while True:
305.         # 2.1 Display Menu to user and get choice
306.         IO.print_menu()
307.         strChoice = IO.menu_choice()
308.
309.         # 3. Process menu selection
310.         # 3.1 process exit first
311.         if strChoice == 'x':
312.             break
313.         # 3.2 process load inventory
314.         if strChoice == 'l':
315.             print('WARNING: If you continue, all unsaved data will be lost and the
    Inventory re-loaded from file.')
316.             strYesNo = input('type \'yes\' to continue and reload from file. otherwise
    reload will be canceled')
317.             if strYesNo.lower() == 'yes':
318.                 print('reloading...')
319.                 lstOfCDObjects = FileProcessor.read_file(strFileName, lstOfCDObjects)
320.                 IO.show_inventory(lstOfCDObjects)
321.             else:
322.                 input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue
    to the menu.')
323.                 IO.show_inventory(lstOfCDObjects)
324.             continue  # start loop back at top.
325.         # 3.3 process add a CD
326.         elif strChoice == 'a':
327.             # 3.3.1 Ask user for new ID, CD Title and Artist
```

```
            strID, title, artist = IO.get_CD() #unpack and call at the same time
            # 3.3.2 Add item to the table
            idnum = int(strID)
            CD(idnum, title, artist)
            DataProcessor.add_CD(idnum, title, artist)
            IO.show_inventory(lstOfCDObjects)
            continue  # start loop back at top.
        # 3.4 process display current inventory
        elif strChoice == 'i':
            IO.show_inventory(lstOfCDObjects)
            #test for objects - for testing only
            #for cds in lstOfCDObjects:
            #    objcd = CD(cds[0], cds[1], cds[2])
            #    print(objcd)
            continue  # start loop back at top.
        # 3.5 process delete a CD
        elif strChoice == 'd':
            # 3.5.1 get Userinput for which CD to delete
            # 3.5.1.1 display Inventory to user
            IO.show_inventory(lstOfCDObjects)
            # 3.5.1.2 ask user which ID to remove
            while True:
                try:
                    intIDDel = int(input('Which ID would you like to delete? ').strip())
                    break
                except ValueError:
                    print('Please type an ID (number) to delete')
            # 3.5.2 search thru table and delete CD
            blnCDRemoved = False
            DataProcessor.delete_Row(intIDDel)
            IO.show_inventory(lstOfCDObjects)
            continue  # start loop back at top.
        # 3.6 process save inventory to file
        elif strChoice == 's':
            # 3.6.1 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()
            # 3.6.2 Process choice
            if strYesNo == 'y':
                FileProcessor.write_file(strFileName, lstOfCDObjects)
            else:
                input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
            continue  # start loop back at top.
        # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
        else:
            print('General Error')
```