Kelly Kauffman

September 1, 2020

IT FDN 110: Introduction to Programming: Python

Assignment 08

Module 08:

# Introduction

In Module 08 I learned about Object Orientated Programming (OOP). I have developed an understanding of how to set up classes; create objects under those classes; and include attributes, properties, constructors, and methods to support the object and give it more functionality. In Assignment 08, I used this understanding of OOP to rewrite the CD inventory program, using the Assignment 08 starter code, to include objects. In this document I will share the steps I took creating Assignment 08 and the major difficulties I faced.

# Topic 1: Module 08 Learning and Labs

Object oriented programming has led me to think in a new way and structure my code differently. Overall, I think I understand how to set up classes and objects, but I am still struggling with how to include OOP effectively in my code and when to use OOP.

The labs from this module were helpful and gave me plenty of opportunity to try out OOP. What I struggled with was understanding exactly what the lab directions were asking me to do. For example, in Lab A, the directions were unclear to me, so I ended up just copying Listing 1 of the Module 08 PDF, but made sure to include position, title, and length like Lab A wanted. In class, Dirk went over all the labs, so I was able to double check my work. It turned out I completed the main objectives of all the labs correctly. My code just varied slightly from Dirk’s. The labs gave me confidence that I could set up a class, create an object, and provide that object and class with attributes, methods, properties, and fields. The only downside with the labs was that we only practiced creating objects/classes and printing out the values associated with the object and its attributes. I did not get a chance to practice integrating objects into a bigger piece of code. Because of this, Assignment 08 was very difficult for me.

# Topic 2: Developing Assignment 08

The starter code for Assignment 08 was overwhelming and I did not know where to begin. I settled on using what I learned from Lab E to create attributes, getters, and setters for class CD. I predicted I would probably need these things at some point in the program. Next, I built out the other two classes, IO and FileIO. I copy and pasted the functions I used in the previous assignment with the understanding that I will probably need to modify them later. I liked that I had a starting point for these functions instead of taking the time to code them from scratch. Then, I worked on the main body of the program. Once again, I copy and pasted elements from Assignment 07 into Assignment 08 so I would have a starting point to work from. Lastly, I implemented the edits Doug suggested on my Assignment 07 code.

At this point, my code was working, but I had not used a single object or created a list of CD objects like the starter code asked for. I very unsure how to integrate objects into my code. The first place in the code where I needed to use instance of the list of CD objects was loading the previously saved inventory from the text file into the local memory. In Assignment 07 this data was loaded into a list of dictionaries. At first, I thought I would need to take this list of dictionaries, unpack it, and convert it to a list of objects somehow. It was hard to wrap my brain around how to do this, so I asked Doug for help. See Listing #1 in the Appendix for this first draft of Assignment 08 that I sent to Doug.

Doug responded with some great hints and edits. He pointed out that my setters and getters were nested under my constructor. It would not have made sense to initialize the setters and getters every time I created an object, so I updated the indentation of the setters and getters. Next, he noticed the attributes I had set up were not private. I had intended them to be private but did not use the correct syntax. I just needed a double underscore in front of the attribute name. Lastly, and most importantly, Doug pointed out that I should not be using dictionaries in this assignment. The list of objects would do the job of the dictionary. This was the big piece I was missing. Like I stated above, I thought I needed to use a list dictionaries in addition to the list of objects.

While I was waiting for Doug’s response, I did do some more thinking and experimenting with my code. I tried out a solution for the FileIO.load\_inventory function where I took out the dictionary pieces and attempted to just use objects for the three CD data points. It did not work in that instance, but I realized later I was very close to the solution Doug gave me. I just did not connect the three pieces of data to the CD class. My attempt is shown in Figure 1. Doug’s solution is shown in Figure 2. In Doug’s solution, cd\_list is instantiated as an object. The ID, CD title, and CD artist data from the file have become attributes of the object. Now that the CD data from the text file is being correctly loaded into a list of CD objects, I went through the rest of my code and rewrote any parts specific to working with a list of dictionaries to now work with just a list.

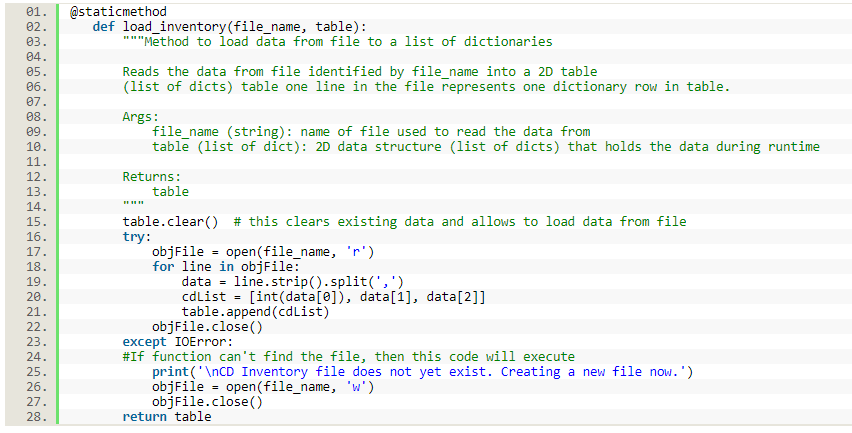


Figure My attempt to add loaded data to a list of CD objects without using a dictionary

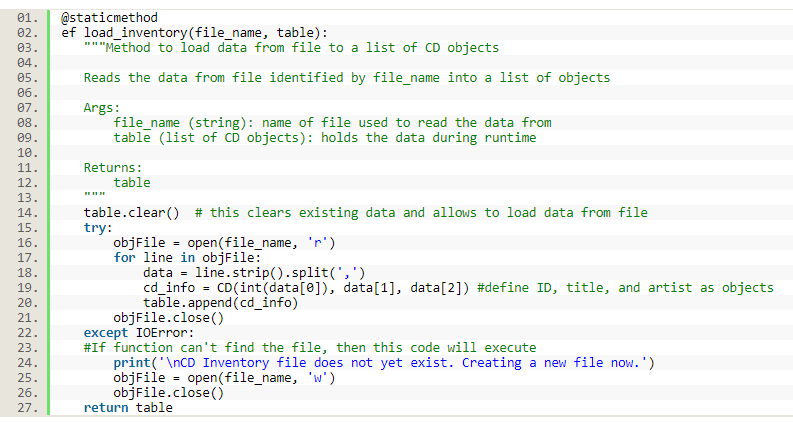


Figure Doug's soultion for load\_inventory where the three pieces of CD data are instantiated as an object

The next part I struggled with was displaying the list of CD objects to the user immediately after the FileIO.load\_inventory ( ) function was executed. The first error I got was "TypeError: format() argument after \* must be an iterable, not CD" for the following lines of code (line 209 and 210 in Listing #1):

for row in table:

print('{}\t{} (by: {})'.format(\*row))

I wondered if the way I was unpacking the list of CD objects was incorrect, so I tried a different method of unpacking:

for row in table:

ID, CDTitle, ArtistName = row

print('{}\t{} (by: {})'.format(row))

I still got an unpacking error, “TypeError: cannot unpack non-iterable CD object”. I needed a different solution to work with the CD objects. I looked through the Module 08 PDF and textbook, and tried a few things, but I did not find anything that fixed the unpacking problem. I decided to call my dad to see if he could help. I talked through my problem and what I had tried so far. Together my dad and I tried some other possible solutions before ultimately fixing the unpacking problem. Listing #5 in the Module 08 PDF helped us realize that I needed to use the getters I had set up at the very beginning of this assignment to access the objects in the list. There was no need to unpack anything, just list the getters for each object inside the .format method. These new lines of code are below:

for row in table:

print('{}\t{} (by: {})'.format(str(row.cd\_id()), row.cd\_title(), row.cd\_artist()))

Python now complained of a new error, “ ‘in’ object is not callable”. Upon further review of Listing #5 in the Module 08 PDF, I noticed that I needed to remove the extra parenthesis after each getter. I could now successfully display the CD inventory. The FileIO.save\_inventory ( ) function needed a similar rework as FileIO.load\_inventory ( ). The original code is found in lines 101-124 of Listing #1. See the following updated code below:

for row in table:

objFile.write(str(row.cd\_id) + ',' + row.cd\_title + ',' + row.cd\_artist + '\n')

My last task was to update how my code added the user inputted CD data into lstofCDObjects. This part of my code is in lines 272-273 of Listing #1. Before, I was calling a function, CD.create\_table, to append the list, but Python could not add non-objects to a list of objects. The solution here was to create an object, cd\_info, and use the user inputs as arguments for CD.\_\_int\_\_. This caused the user inputs to become part of the object class, CD, and thus be allowed to be added to lstofCDObjects. Below are the new lines of code for lines 272-273 of Listing #1:

cd\_info = CD(int(strID), strTitle, strArtist)

lstOfCDObjects.append(cd\_info)

Throughout this assignment, it really helped to talk about my code with another person (i.e. Doug and my dad). When I try to solve problems on my own I can sometimes easily get lost in all the examples and text in the textbook, Module PDFs, and internet articles. With Doug and my dad’s help I was able to develop a clearer understanding of objects, but I still need a lot more practice.

# Topic 3: Final Products

Here is the link to the final code for Assignment 08 on Github:

<https://github.com/pythongal6295/Assignment_08.git>

Below, Figure 3, 4, and 5 show my code working in Spyder. Figure 6 and 7 show my code working in the Anaconda Prompt.

A screenshot of a cell phone

Description automatically generated

Figure Assignment 08 working in Spyder part 1

A screenshot of a cell phone

Description automatically generated

Figure Assignment 08 working in Spyder part 2

A screenshot of a cell phone

Description automatically generated

Figure Assignment 08 working in Spyder part 3

A screenshot of a cell phone

Description automatically generated

Figure Assignment 08 working in the Anaconda Prompt part 1

A screenshot of a cell phone

Description automatically generated

Figure Assignment 08 working in the Anaconda Prompt part 2

# Summary

The focus of Module 08 was Object Orientated Programming. I learned how to create objects; and set up attributes, properties, and methods that are associated with an object class. The labs taught me the syntax for OOP and Assignment 08 taught me how to integrate objects within a program. Assignment 08 asked for me to use a starter code and complete the TODOs that were in the psuedocode. The TODOs centered around using objects and an object class to accomplish the desired functionality of the CD Inventory program. In addition, I needed to make sure I updated the docstrings and included error handling.

# Appendix

Listing #1: First Draft of Assignment 08

1. #------------------------------------------#
2. # Title: Assignmen08.py
3. # Desc: Assignnment 08 - Working with classes
4. # Change Log: (Who, When, What)
5. # DBiesinger, 2030-Jan-01, created file
6. # DBiesinger, 2030-Jan-01, added pseudocode to complete assignment 08
7. # KKauffman, 2020-Aug-30, added code to class CD and FileIO, began Main code
8. # KKauffman, 2020-Sept-01, continued building Main code
9. #------------------------------------------#
11. # -- DATA -- #
12. strFileName = 'cdInventory.txt'
13. lstOfCDObjects = []
15. **class** CD:
16. """Stores data about a CD:
18. properties:
19. cd\_id: (int) with CD ID
20. cd\_title: (string) with the title of the CD
21. cd\_artist: (string) with the artist of the CD
22. methods:
24. """
25. # TODO Add Code to the CD class
26. #Fields
27. #Constructor (do I need this?)
28. **def** \_\_init\_\_(self, cd\_id, cd\_title, cd\_artist):
29. **pass**
31. #Attributes
32. self.cd\_id = cd\_id
33. self.cd\_title = cd\_title
34. self.cd\_artist = cd\_artist
36. #Properties
37. @property
38. **def** cd\_position(self):
39. **return** self.\_\_cd\_id
41. @property
42. **def** cd\_title(self):
43. **return** self.\_\_cd\_title
45. @property
46. **def** cd\_artist(self):
47. **return** self.\_\_cd\_artist
49. @cd\_id.setter
50. **def** cd\_id(self, new\_cd\_id):
51. **if** type (new\_cd\_id) == int:
52. self.\_\_cd\_id == new\_cd\_id
53. **else**:
54. **raise** Exception ('CD ID needs to be an integer.')
56. @cd\_title.setter
57. **def** cd\_title(self, new\_cd\_title):
58. self.\_\_cd\_title == new\_cd\_title
60. @cd\_artist.setter
61. **def** cd\_artist(self, new\_cd\_artist):
62. self.\_\_cd\_artist == new\_cd\_artist
64. #methods
65. #Could add something about counting the number of Cds in the inventory
66. #Might need a create table method
67. @staticmethod
68. **def** create\_table(strID, strTitle, strArtist, table):
69. """Function to take list of user inputs and put them in a dictionary (dicRow)
71. Arg:
72. table/list to store user inputs
73. Taken from unpacked tuple returned by IO.user\_inputs()
74. strID: User's inputted CD ID
75. strTitle: User's inputted CD title
76. strArtist: User's inputted CD Artist
78. Return:
79. None
81. """
83. dicRow = {'ID': strID, 'Title': strTitle, 'Artist': strArtist}
84. table.append(dicRow)
86. # -- PROCESSING -- #
87. **class** FileIO:
88. """Processes data to and from file:
90. properties:
92. methods:
93. save\_inventory(file\_name, lst\_Inventory): -> None
94. load\_inventory(file\_name): -> (a list of CD objects)
96. """
97. #Constructor (do I need this?)
98. **def** \_\_init\_\_(self, file\_name, table):
99. **pass**
100. #Properties
102. #Methods
103. # TODO Add code to process data from a file
104. @staticmethod
105. **def** save\_inventory(file\_name, table):
106. """Function to write data to file
108. Takes each row from table and separates the items in it by a comma
109. and adds a \n at the end
111. Args:
112. file\_name (string): name of file data is saved to
113. table (lists of dicts): 2D data structure that holds the data
115. Returns: print statement confirming save is complete
117. """
118. objFile = open(strFileName, 'w')
119. **for** row **in** table:
120. lstValues = list(row)
121. lstValues[0] = str(lstValues[0])
122. objFile.write(','.join(lstValues) + '\n')
123. objFile.close()
124. **return** **print**('\nYour data has been saved.')
126. # TODO Add code to process data to a file
127. @staticmethod
128. **def** load\_inventory(file\_name, table):
129. """Method to load data from file to a list of dictionaries
131. Reads the data from file identified by file\_name into a 2D table
132. (list of dicts) table one line in the file represents one dictionary row in table.
134. Args:
135. file\_name (string): name of file used to read the data from
136. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
138. Returns:
139. table
140. """
141. table.clear()  # this clears existing data and allows to load data from file
142. **try**:
143. objFile = open(file\_name, 'r')
144. **for** line **in** objFile:
145. data = line.strip().split(',')
146. dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
147. table.append(dicRow)
148. objFile.close()
149. **except** IOError:
150. #If function can't find the file, then this code will execute
151. **print**('\nCD Inventory file does not yet exist. Creating a new file now.')
152. objFile = open(file\_name, 'r')
153. objFile.close()
154. **return** table
156. # -- PRESENTATION (Input/Output) -- #
157. **class** IO:
158. # TODO add docstring
159. """Handling Input / Output"""
161. # TODO add code to show menu to user
162. @staticmethod
163. **def** print\_menu():
164. """Displays a menu of choices to the user
166. Args:
167. None.
169. Returns:
170. None.
171. """
172. **print**('\nCD Inventory Menu\n\n[l] Load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
173. **print**('[s] Save Inventory to file\n[x] Exit\n')
175. # TODO add code to captures user's choice
176. @staticmethod
177. **def** menu\_choice():
178. """Gets user input for menu selection
180. Args:
181. None.
183. Returns:
184. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
186. """
187. choice = ' '
188. **while** choice **not** **in** ['l', 'a', 'i', 's', 'x']:
189. choice = input('Which operation would you like to perform? [l, a, i, s or x]: ').lower().strip()
190. **print**()  # Add extra space for layout
191. **return** choice
193. # TODO add code to display the current data on screen
194. @staticmethod
195. **def** show\_inventory(table):
196. """Displays current inventory table

199. Args:
200. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
202. Returns:
203. None.
205. """
206. **print**()
207. **print**('------- The Current Inventory: -------')
208. **print**('ID\tCD Title (by: Artist)\n')
209. **for** row **in** table:
210. **print**('{}\t{} (by:{})'.format(\*row.values()))
211. **print**('--------------------------------------')
212. **print**()
214. # TODO add code to get CD data from user
215. @staticmethod
216. **def** user\_inputs():
217. """Function to gather the user's inputs for CD ID, CD Title, and CD Artist
219. Arg: none
221. Return: a tuple of the three user inputs (entryID, entryTitle, entryArtist)
223. """
224. #Will catch if the user enters an noninteger for the ID. Loop will continue to prompt them.
225. entryID = ''
226. **while** True:
227. **try**:
228. entryID = int(input('Enter ID: ').strip())
229. **break**
230. **except** ValueError:
231. **print**('\nOops! You must enter an integer for the CD ID.')
232. **continue**
234. entryTitle = input('What is the CD\'s title? ').strip()
235. entryArtist = input('What is the Artist\'s name? ').strip()
236. **return** (entryID, entryTitle, entryArtist)
238. # -- Main Body of Script -- #
239. # TODO Add Code to the main body
241. #Create CD Object
242. #objCD = CD()
243. #Create File Object
244. #objFile = FileIO()
246. # Load data from file into a list of CD objects on script start
247. # 1. When program starts, read in the currently saved Inventory
249. **print**('\nWelcome to your CD Inventory!')

252. lstOfCDObjects = FileIO.load\_inventory(strFileName, lstOfCDObjects)
254. # 2. start main loop
256. **while** True:
257. # 2.1 Display Menu to user
258. IO.print\_menu()
259. strChoice = IO.menu\_choice()
260. # show user current inventory
261. # 3.4 process display current inventory
262. **if** strChoice == 'i':
263. IO.show\_inventory(lstOfCDObjects)
264. **continue**  # start loop back at top.
265. # let user add data to the inventory
266. # 3.3 process add a CD
267. **elif** strChoice == 'a':
269. # 3.3.1 Ask user for new ID, CD Title and Artist
270. strID, strTitle, strArtist = IO.user\_inputs() #Assigned return to variables and unpacked this tuple
272. # 3.3.2 Add item to the table
273. CD.create\_table(strID, strTitle, strArtist, lstOfCDObjects) #Arguments are unpacked tuple from IO.user\_inputs()
274. **continue**  # start loop back at top.
276. # let user save inventory to file
277. # 3.6 process save inventory to file
278. **elif** strChoice == 's':
280. # 3.6.1 Display current inventory and ask user for confirmation to save
281. IO.show\_inventory(lstOfCDObjects)
282. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
284. # 3.6.2 Process choice
285. **if** strYesNo == 'y':
287. # 3.6.2.1 save data
288. FileIO.save\_inventory(strFileName, lstOfCDObjects)
290. **else**:
291. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
292. **continue**  # start loop back at top.
293. # let user load inventory from file
294. # 3.2 process load inventory
295. **elif** strChoice == 'l':
296. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
297. strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled\n')
298. **if** strYesNo.lower() == 'yes':
299. **print**('\nreloading...')
300. FileIO.load\_inventory(strFileName, lstOfCDObjects)
301. IO.show\_inventory(lstOfCDObjects)
302. **else**:
303. input('\ncanceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.\n')
304. IO.show\_inventory(lstOfCDObjects)
305. **continue**  # start loop back at top.
306. # let user exit program
307. # 3.1 process exit first
308. **elif** strChoice == 'x':
309. **break**
310. **else**:
311. **print**('General Error')