**Kasisi Harris**

**2022-Aug-28**

**IT FDN 110 A Su 22: Foundations of Programming**

**Assignment\_08**

**GitHub link:** [kasisi79/Assignment\_08 (github.com)](https://github.com/kasisi79/Assignment_08)

Constructors, Getters, and Setters

# Introduction

If there was ever a pit of despair, this is it. This week, we were to define objects and use Constructors, Getters, and Setters in order to recreate the functionality of our CDInventory scripts from previous weeks. The assignment also instructs us to test our code and add error proofing, when possible, to ensure the code is operating as intended. Each of lab\_08 exercises seemed very straight forward; I didn’t get them on the first try. However, after sticking with it, I was able to understand how to code the script. Unfortunately, I was not able to translate that knowledge into the objects involved with the CDInventory code. Though I’m very disappointed with the results, I will detail my journey below.

# Lab 08

As I mentioned earlier, implementing Getter and Setters seemed straight forward with some time. Below is both the Python and Terminal runs of the lab. The actual code can been found in the Appendix.

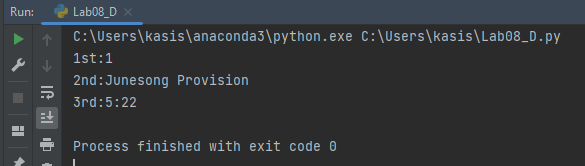


Figure - Lab08 - Python Run

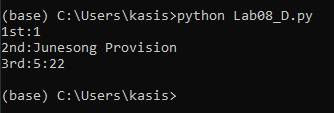
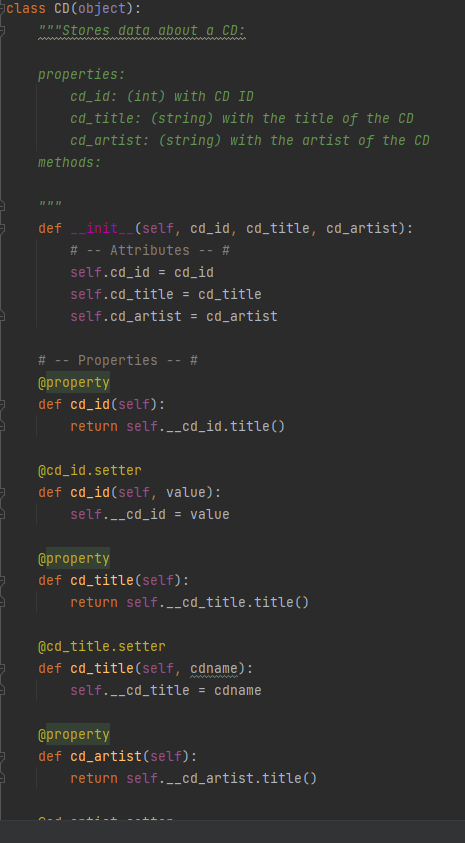


Figure - Lab08 - Terminal Run

# Assignment 08 Attempt.

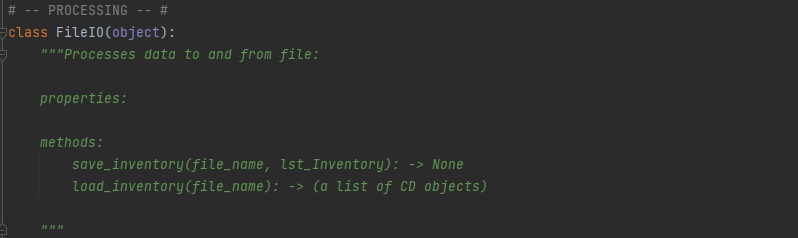
I took what I learned from the lab defining objects and using Getters and Setters in order to access, (or attempt to access), the objects from outside their local function. The lab focused on using a single object. However, the doc string instructs us to create three objects. Right away, I ran into an conceptual issue, I didn’t understand how to convert “cd\_id” into type integer.



Listing - Assignment 08 - Storage Getters & Setters

I tried several methods. However, all seemed to cause errors, so I left the value as a string for the moment to try to attempt the rest of the script.

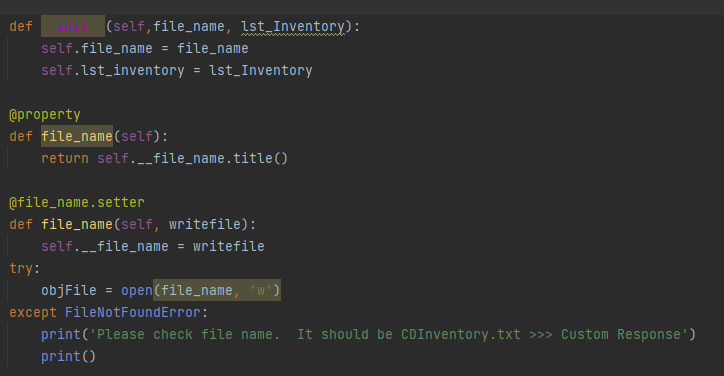
This is where I really began to bang my head against the wall. While attempting the functions within the class FileIO(), I was unable to truly understand how to incorporate the objects identified in order to execute the process of reading and writing to the file. Below is my attempt at reading and writing to file based on the identified properties.



Listing - Assignment 08 - Storage Parameters



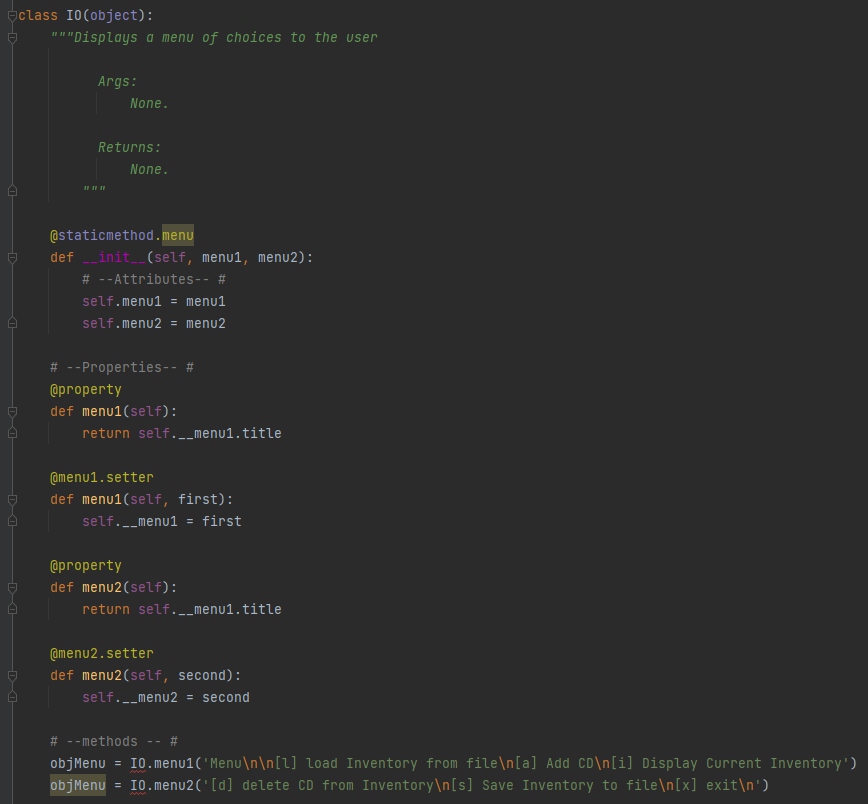
Listing - Assignment 08 - Read File



Listing - Assignment 08 - Write to File

Notice in listing 3 that the positional parameter “file\_name” isn’t recognized when I try to use “objFile” to open it. I considered moving this. However, that wouldn’t make sense because our assignment is to place this in the function. I’m not exactly sure what I’m missing. However, I wanted to keep pushing to perhaps complete the other section of the script and return to this portion once I completed the other sections.

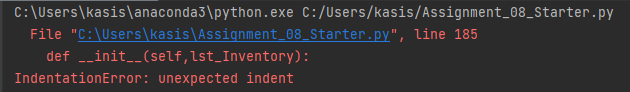
The other big concept I just couldn’t wrap my head around was how to call my objects into my variables. The lab made this seem very simple. However, my attempts at this proved quite frustrating. Below is an example of me trying display the prompt for the user to select a letter to complete an action.



Listing - Assignment 08 - Menu

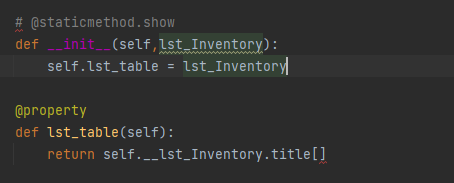
Looking at examples on-line, they typically apply the class combined with the object name in order to complete this. However, when I attempted to do this using “menu1” and “menu2” it wouldn’t recognize the class. I just don’t understand what I’m failing to do. I’m sure it’s simple. However, I’ve gone round and round for hours trying to change the lines in the hopes that I could successfully troubleshoot the problem.

Error Handling: I didn’t even know where to begin. None of my code would execute. I kept getting the following:



Listing - Assignment 08 - Indentation Error

An indent error sounds so simple to fix. I check the relationship in positioning between line 185, (below), and the other similar code and it seems to be correct.



Listing - Assignment 08 - line 185

No matter where I would move the code, it would still present the same error, or change the line based on the new position of the code.

I wasn’t sure what to even implement besides the handling implemented on the previous assignment. Adding the error checking presented in the lab won’t do any good if my script doesn’t have enough working code for it to be effective.

# Summary

So, what did I learn from this. I feel like I understand the concept of simple Setters, Getters, and class definition. However, my fear is that I truly don’t understand this well enough to complete Assignment 09. I’ll need to reach out for some virtual office hours in order to understand this well enough to correct the code and complete next week’s assignment.

# Appendix

## Listing of Lab08\_D

1. **class** TrackInfo(object):
3. *# --Fields--#*
4. *# -- Constructor -- #*
5. **def \_\_init\_\_(self, position, top, length):**
6. *# -- Attributes -- #*
7. self.position = position
8. self.top = top
9. self.length = length
11. *# -- Properties --#*
12. @property
13. **def** position(self):
14. **return** self.\_\_position.title()
16. @position.setter
17. **def** position(self, value):
18. self.\_\_position = value
20. **@property**
21. **def** top(self):
22. **return** self.\_\_top.title()
24. @top.setter
25. **def top(self, track):**
26. self.\_\_top = track
28. @property
29. **def** length(self):
30. **return self.\_\_length.title()**
32. @length.setter
33. **def** length(self, time):
34. self.\_\_length = time
36. *# -- Methods -- #*

39. objTrack1 = TrackInfo('1', 'Junesong Provision', '5:22')
41. **print**('1st:{}'.format(objTrack1.position))
42. **print**('2nd:{}'.format(objTrack1.top))
43. **print**('3rd:{}'.format(objTrack1.length))

## Listing of CDInventory.py

1. *#------------------------------------------#*
2. *# Title: CDInventory.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. *# KHarris, 2022-Aug-28, complete tasks in pseudocode*
8. *#------------------------------------------#*
10. ***# -- DATA -- #***
11. strFileName = 'cdInventory.txt'
12. lstOfCDObjects = []
14. **class** CD(object):
15. **"""Stores data about a CD:**
17. properties:
18. cd\_id: (int) with CD ID
19. cd\_title: (string) with the title of the CD
20. **cd\_artist: (string) with the artist of the CD**
21. methods:
23. """
24. **def** \_\_init\_\_(self, cd\_id, cd\_title, cd\_artist):
25. ***# -- Attributes -- #***
26. self.cd\_id = cd\_id
27. self.cd\_title = cd\_title
28. self.cd\_artist = cd\_artist
30. ***# -- Properties -- #***
31. @property
32. **def** cd\_id(self):
33. **return** self.\_\_cd\_id.title()
35. **@cd\_id.setter**
36. **def** cd\_id(self, value):
37. self.\_\_cd\_id = value
39. @property
40. **def cd\_title(self):**
41. **return** self.\_\_cd\_title.title()
43. @cd\_title.setter
44. **def** cd\_title(self, cdname):
45. **self.\_\_cd\_title = cdname**
47. @property
48. **def** cd\_artist(self):
49. **return** self.\_\_cd\_artist.title()
51. @cd\_artist.setter
52. **def** cd\_artist(self, artist):
53. self.\_\_cd\_artist = artist
55. ***# -- PROCESSING -- #***
56. **class** FileIO(object):
57. """Processes data to and from file:
59. properties:
61. methods:
62. save\_inventory(file\_name, lst\_Inventory): -> None
63. load\_inventory(file\_name): -> (a list of CD objects)
65. **"""**
66. **def** \_\_init\_\_(self,file\_name):
67. self.file\_name = file\_name
69. @property
70. **def file\_name(self):**
71. **return** self.\_\_file\_name.title
73. @file\_name.setter
74. **def** file\_name(self, readfile):
75. **self.\_\_file\_name = readfile**
76. **try**:
77. objFile = open(file\_name, 'r')
78. **except** FileNotFoundError:
79. **print**('Please check file name. It should be CDInventory.txt >>> Custom Response')
80. **print()**
81. **for** line **in** objFile:
82. data = line.strip().split(',')
83. dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
84. lst\_Inventory.append(dicRow)
85. **objFile.close()**
87. **def** \_\_init\_\_(self,file\_name, lst\_Inventory):
88. self.file\_name = file\_name
89. self.lst\_inventory = lst\_Inventory
91. @property
92. **def** file\_name(self):
93. **return** self.\_\_file\_name.title()
95. **@file\_name.setter**
96. **def** file\_name(self, writefile):
97. self.\_\_file\_name = writefile
98. **try**:
99. objFile = open(file\_name, 'w')
100. **except FileNotFoundError:**
101. **print**('Please check file name. It should be CDInventory.txt >>> Custom Response')
102. **print**()
104. @property
105. **def lst\_Inventory(self):**
106. **return** self.\_\_lst\_Inventory.title()
108. **def** lst\_Inventory(self, table):
110. **for row in lst\_Inventory:**
111. lstValues = list(row.values())
112. lstValues[0] = str(lstValues[0])
113. objFile.write(','.join(lstValues) + '**\n**')
114. objFile.close()


118. *# -- PRESENTATION (Input/Output) -- #*
120. **class IO(object):**
121. """Displays a menu of choices to the user
123. Args:
124. None.
126. Returns:
127. None.
128. """
130. **@staticmethod.menu**
131. **def** \_\_init\_\_(self, menu1, menu2):
132. *# --Attributes-- #*
133. self.menu1 = menu1
134. self.menu2 = menu2
136. *# --Properties-- #*
137. @property
138. **def** menu1(self):
139. **return** self.\_\_menu1.title
141. @menu1.setter
142. **def** menu1(self, first):
143. self.\_\_menu1 = first
145. **@property**
146. **def** menu2(self):
147. **return** self.\_\_menu1.title
149. @menu2.setter
150. **def menu2(self, second):**
151. self.\_\_menu2 = second
153. *# --methods -- #*
154. objMenu = IO.menu1('Menu**\n\n**[l] load Inventory from file**\n**[a] Add CD**\n**[i] Display Current Inventory')
155. **objMenu = IO.menu2('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')**
157. *# @staticmethod.choice*
158. **def** menu\_choice():
159. """Gets user input for menu selection
160. **Args:**
161. None.
162. Returns:
163. Return choice
164. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
165. **"""**
166. **def** \_\_init\_\_(self, choice):
167. *# --Attributes-- #*
168. self.choice = choice
169. *# choice = ' '*
170. **while choice not in ['l', 'a', 'i', 'd', 's', 'x']:**
171. choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
172. **print**() *# Add extra space for layout*
174. @property
175. **def choice(self):**
176. **return** self.\_\_choice.title()
178. @choice.setter
179. **def** choice(self, schoice):
180. **self.\_\_choice = schoice**
181. objChoice = input(IO.menu\_choice('Which operation would you like to perform? [l, a, i, d, s, or x]: ').lower().strip())

184. *# @staticmethod.show*
185. **def \_\_init\_\_(self,lst\_Inventory):**
186. self.lst\_table = lst\_Inventory
188. @property
189. **def** lst\_Inventory(self):
190. **return self.\_\_lst\_Inventory.title[]**
192. @lst\_Inventory.setter
193. **def** lst\_Inventory(self, inventory):
194. self.\_\_lst\_Inventory.title[] = inventory
195. **for row in lst\_Inventory:**
196. **print**('{}**\t**{} (by:{})'.format(\*row.values()))
197. """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**('======= The Current Inventory: =======')
207. **print**('ID**\t**CD Title (by: Artist)**\n**')
208. **for** row **in** lst\_Inventory:
209. **print**('{}**\t**{} (by:{})'.format(\*row.values()))
210. **print('======================================')**
212. @property
213. **def** lst\_Inventory(self):
214. **print**('======= The Current Inventory: =======')
215. **print('ID\tCD Title (by: Artist)\n')**
216. **for** row **in** lst\_Inventory:
217. **print**('{}**\t**{} (by:{})'.format(\*row.values()))
218. **print**('======================================')
220. ***# -- Main Body of Script -- #***
221. *# Load data from file into a list of CD objects on script start*
223. **while** True:
224. IO.print\_menu()
225. **user\_choice = IO.menu\_choice()**
226. **if** user\_choice == 'x':
227. **break**
228. **if** user\_choice == 'l':
229. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
230. **strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled')**
231. **if** strYesNo.lower() == 'yes':
232. **print**('reloading...')
233. FileIO.read\_file(strFileName, lst\_Inventory)
234. IO.show\_inventory(lst\_Inventory)
235. **else:**
236. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
237. IO.show\_inventory(lst\_Inventory)
238. **continue** *# start loop back at top.*
240. **elif user\_choice == 'a':**
242. IO.show\_inventory(lst\_Inventory)
244. **continue** *# start loop back at top.*
246. **elif** user\_choice == 'i':
247. IO.show\_inventory(lst\_Inventory)
248. **continue** *# start loop back at top.*
250. **elif user\_choice == 'd':**
252. IO.show\_inventory(lst\_Inventory)
254. **try**:
255. **intIDDel = int(input('Which ID would you like to delete? ').strip())**
256. **except** ValueError:
257. **print**('The value entered is not an integer. Please try again later. >>> Custom Response')
258. **break**
260. **DataProcessor.DeleteEntry(intIDDel)**
261. IO.show\_inventory(lst\_Inventory)
262. **continue** *# start loop back at top.*
263. **elif** user\_choice == 's':
265. **IO.show\_inventory(lst\_Inventory)**
266. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
267. *# 3.6.2 Process choice*
268. **if** strYesNo == 'y':
269. FileIO.write\_file(strFileName, lst\_Inventory)
270. **else:**
271. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
272. **continue** *# start loop back at top.*
274. **else**:
275. **print('General Error')**
276. *# let user load inventory from file*
277. *# let user exit program*