<Hongyi Zhu>

<2021-12-04>

<IT FDN 110>

<Assignment\_Module\_08>

Object Oriented Programming: Classes and Objects

# Introduction

In this module, we started to learn programming in a different way of thinking, which is object oriented programming. In this way, we can do the programming in a more flexible manner and others can import our scripts easily. First is the concept of class. A class is a set of both data and functionality of all corresponding objects. Under the class, we have fields, constructors, attributes, properties, and methods. Fields are the data stores of a class and are created the same way as variables so far. Constructors are a kind of internal method which we use \_\_init\_\_() to call it. Attributes are internal fields or variables which can hold data in python. Properties are another kind of special methods which can make the attributes private. Methods are like functions which we can call them later when processing the data. The main difference between methods and functions is that the first attribute supplied to a method is the “self” reference.

# Topic 1

I used [Syntax Highlighter](https://highlight.hohli.com/index.php) (external reference)[[1]](#footnote-1) web page to highlight the script as shown in Listing 1.

Notes:

1. Class CD (Line 4-12):

In this class, the main purpose is to receive the user’s data and store it into dict as one CD. The method of \_\_init\_\_ is to save all the parameters into attributes. Decorator “@property” made the second method “oneCD” a property and returned a dict as CD.

1. Class FileIO (Line 14-43):

This part processes data to and from file. “@staticmethod” made this method does not need instantiation to call. In the first method “load\_inventory”, I used “try-except” block to handle ValueError and FileNotFoundError, letting the user to correct the errors. In the second method “save\_inventory”, I saved the inventory to the file “CDInventory.txt”.

1. Class IO (Line 46-84):

This class deals with the user’s input and output. “@staticmethod” made this method does not need instantiation to call. The method “show\_menu” showed user the menu. “menu\_choice” captured the user’s choice and returned the user’s input out of the choices d, a, s, l, or x. “show\_inventory” method displayed the current data on screen as a 2D data structure (dict) that held the data during runtime. “get\_cd” got the CD data from user and returned a dict as CD. “add\_inventory” added cd from IO.get\_cd() to the inventory.

1. Line 88-123:

This is the main body of the script. It used a loop to deal with the user’s different input and call the corresponding method.

1. # -- DATA -- #
2. strFileName = "CDInventory.txt"
3. lstOfCDObjects = []
4. class CD:
5. def \_\_init\_\_(self,cd\_id,cd\_title,cd\_artist):
6. self.\_\_cd\_id = cd\_id
7. self.\_\_cd\_title = cd\_title
8. self.\_\_cd\_artist = cd\_artist
10. @property
11. def oneCD(self):
12. return {'ID':self.\_\_cd\_id,'Title':self.\_\_cd\_title,'Artist':self.\_\_cd\_artist}
14. class FileIO:
15. @staticmethod
16. def load\_inventory(file\_name):
17. table = []
18. try:
19. objFile = open(file\_name, 'r')
20. for line in objFile:
21. data = line.strip().split(',')
22. dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
23. table.append(dicRow)
24. except ValueError:
25. print('The ID is not an integer, please correct it in txt file of '+ file\_name)
26. objFile.close()
27. return table
28. except FileNotFoundError:
29. print('This file ' + file\_name + ' does not exist,will create blank ' + file\_name)
30. objFile = open(file\_name, 'w')
31. objFile.close()
32. return table
33. objFile.close()
34. return table
36. @staticmethod
37. def save\_inventory(file\_name, table):
38. objFile = open(file\_name, 'w')
39. for row in table:
40. lstValues = list(row.values())
41. lstValues[0] = str(lstValues[0])
42. objFile.write(','.join(lstValues) + '\n')
43. objFile.close()
45. # -- PRESENTATION (Input/Output) -- #
46. class IO:
47. @staticmethod
48. def show\_menu():
49. print('\n\n---------------Menu-------------\n\n[d] show user current inventory\n[a] add data to the inventory\n[s] save inventory to file')
50. print('[l] load inventory from file\n[x] exit\n')
52. @staticmethod
53. def menu\_choice():
54. choice = ' '
55. while choice not in ['d', 'a', 's', 'l', 'x']:
56. choice = input('Which operation would you like to perform? [d, a, s, l or x]: ').lower().strip()
57. return choice
59. @staticmethod
60. def show\_inventory(table):
61. print('======= The Current Inventory: =======')
62. print('ID\tCD Title (by: Artist)\n')
63. for row in table:
64. print('{}\t{} (by:{})'.format(\*row.values()))
65. print('======================================')

68. @staticmethod
69. def get\_cd():
70. while True:
71. intID = input("input an integer as CD ID:")
72. if intID.isdigit():
73. intID = int(intID)
74. break
75. else:
76. print("please input an integer!")
77. strTitle = input("input CD title:").strip()
78. strArtist = input("input CD artist:").strip()
79. cd = CD(intID,strTitle,strArtist).oneCD
80. return cd
82. @staticmethod
83. def add\_inventory(lstCD,cd):
84. lstCD.append(cd)

87. # -- Main Body of Script -- #
88. lstOfCDObjects = FileIO.load\_inventory(strFileName)
89. while True:
90. IO.show\_menu()
91. strChoice = IO.menu\_choice()
92. if strChoice == 'x':
93. break
94. if strChoice == 'd':
95. IO.show\_inventory(lstOfCDObjects)
96. continue
97. if strChoice == 'a':
98. cd = IO.get\_cd()
99. IO.add\_inventory(lstOfCDObjects,cd)
100. IO.show\_inventory(lstOfCDObjects)
101. continue
102. if strChoice == 'l':
103. lstOfCDObjects = FileIO.load\_inventory(strFileName)
104. IO.show\_inventory(lstOfCDObjects)
105. continue
106. print("WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.")
107. strYesNo = input("type \'yes\' to continue and reload from file. otherwise reload will be canceled")
108. if strYesNo.lower() == "yes":
109. print("reloading...")
110. lstOfCDObjects = FileIO.load\_inventory(strFileName)
111. else:
112. input("canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.")
113. IO.show\_inventory(lstOfCDObjects)
114. continue
115. if strChoice == 's':
116. FileIO.save\_inventory(strFileName,lstOfCDObjects)
117. IO.show\_inventory(lstOfCDObjects)
118. strYesNo = input("Save this inventory to file? [y/n] ").strip().lower()
119. if strYesNo == 'y':
120. FileIO.save\_inventory(strFileName,lstOfCDObjects)
121. else:
122. input("The inventory was NOT saved to file. Press [ENTER] to return to the menu.")
123. continue

Listing - Script of Assignment 08

# Topic 2

In this topic, I ran the script and tested all options in spyder.

Figure 1 is the testing result on spyder of [a] add data to the inventory, [d] show user current inventory, [s] save inventory to file, and [x] exit. It’s also the testing result of “try-except” block where there was no txt file at first and it created a blank file automatically.

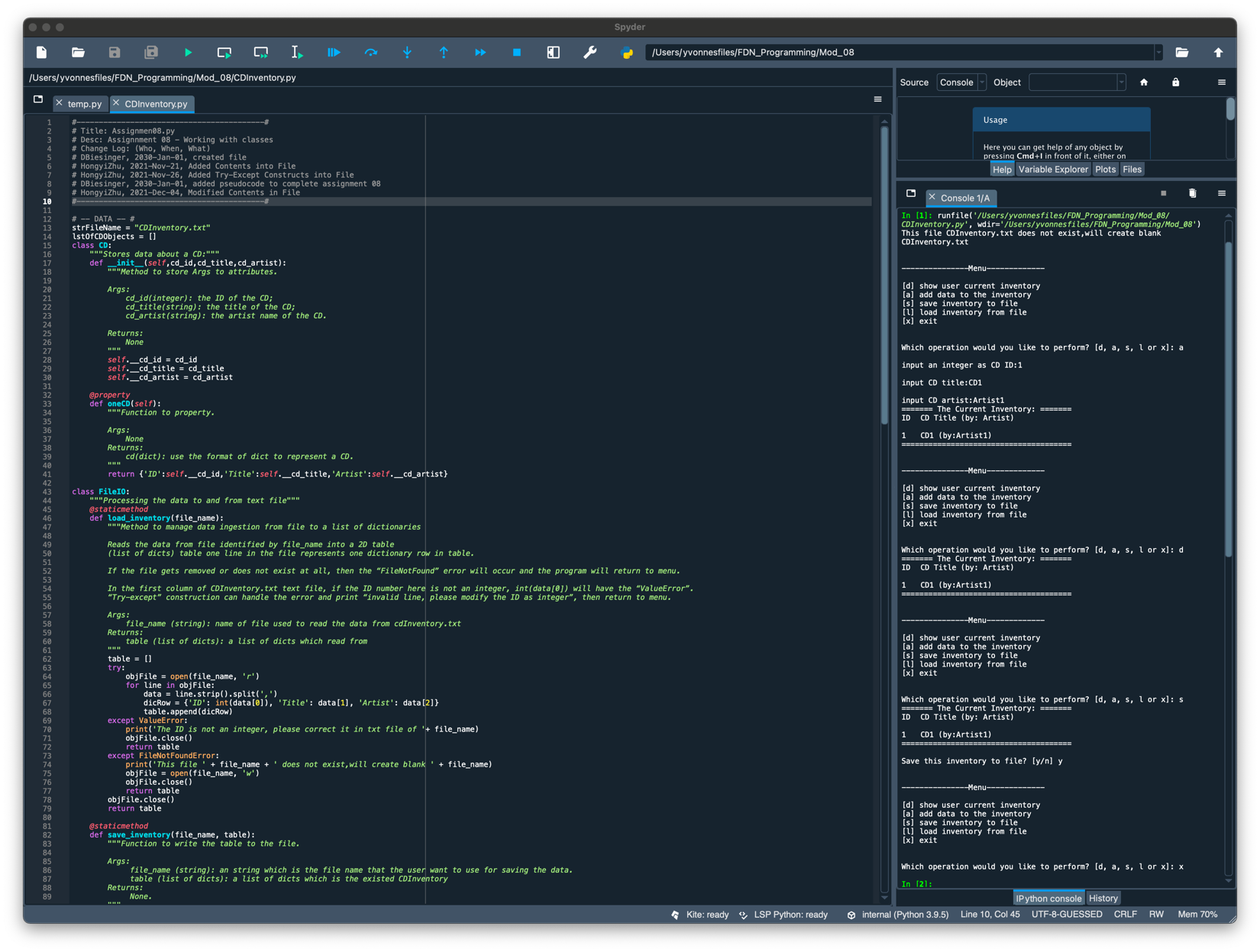


Figure - Running Results of Testing [a], [d], [s] and [x] in Spyder

Figure 2 is the testing result on spyder of [l] load inventory from file.

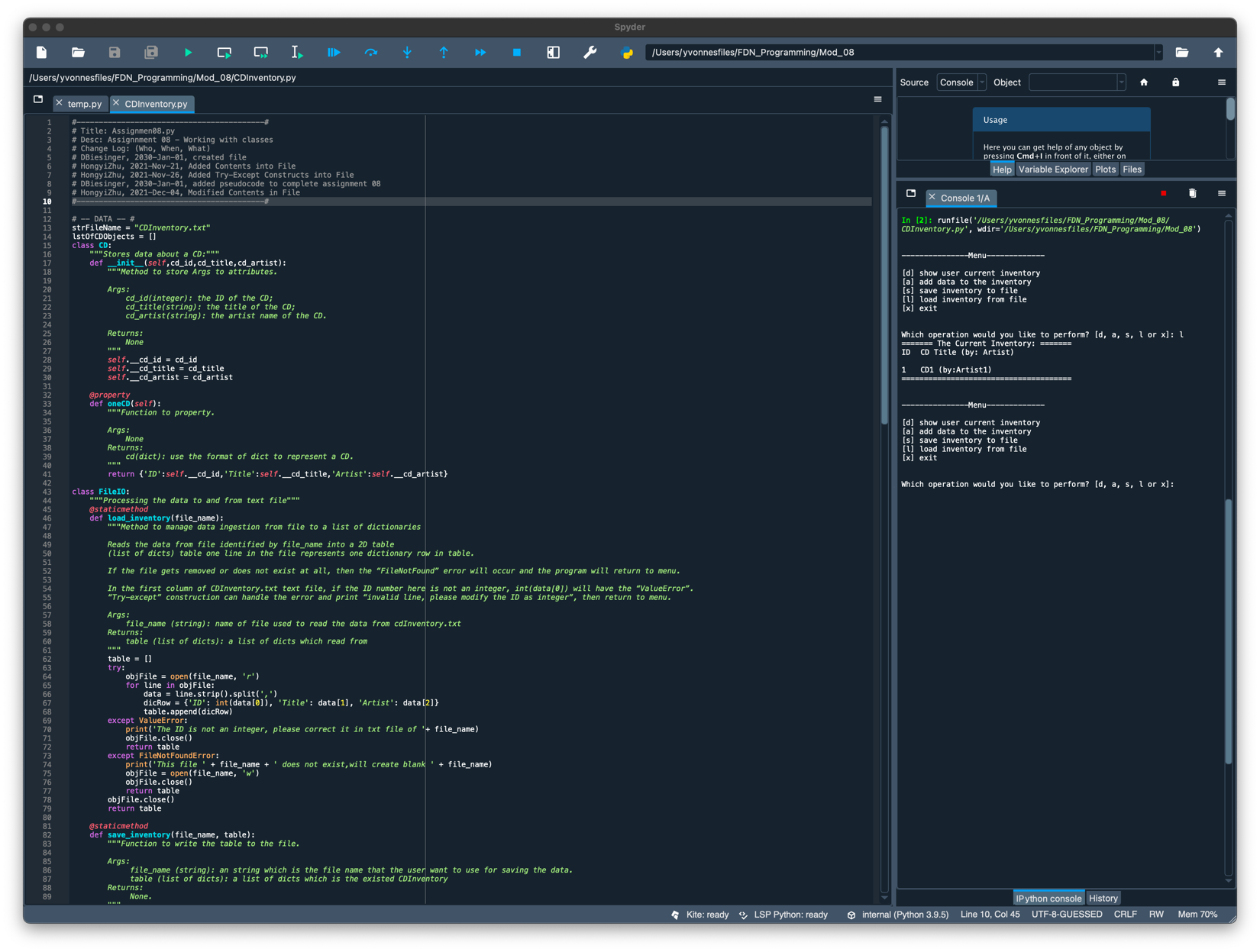


Figure - Running Results of Testing [l] in Spyder

Figure 3 below is the testing result in the text file of adding one line of CD and saving it to file.

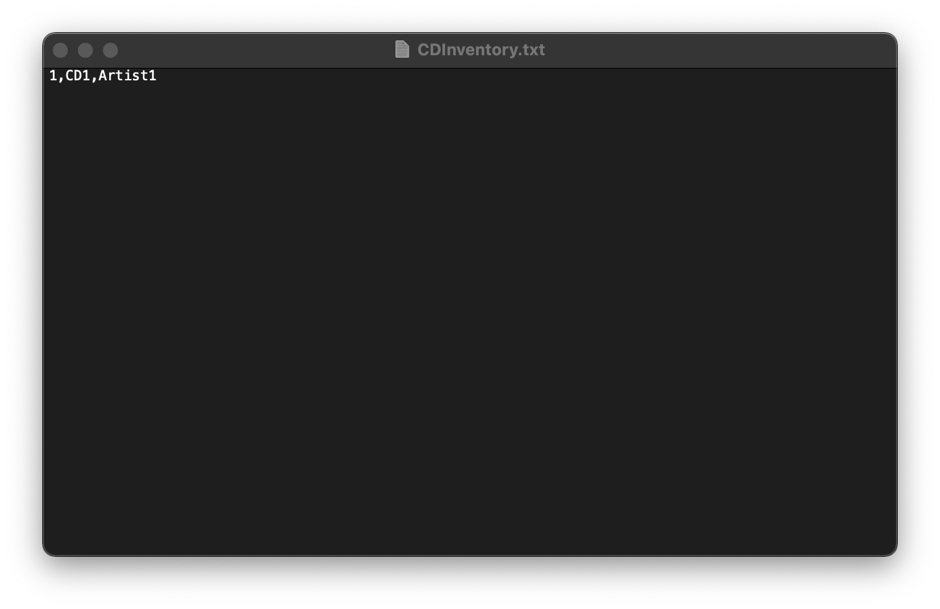


Figure - Testing Results of CDInventory.txt

# Topic 3

In this topic, I repeated all of the same steps in the terminal as what I did exactly in spyder.

Figure 4 is the testing result on terminal of [a] add data to the inventory, [d] show user current inventory, [s] save inventory to file, and [x] exit. It’s also the testing result of “try-except” block where there was no txt file at first and it created a blank file automatically.



Figure - Running Results of Testing [a], [d], [s] and [x] in Terminal

Figure 5 is the testing result on spyder of [l] load inventory from file.

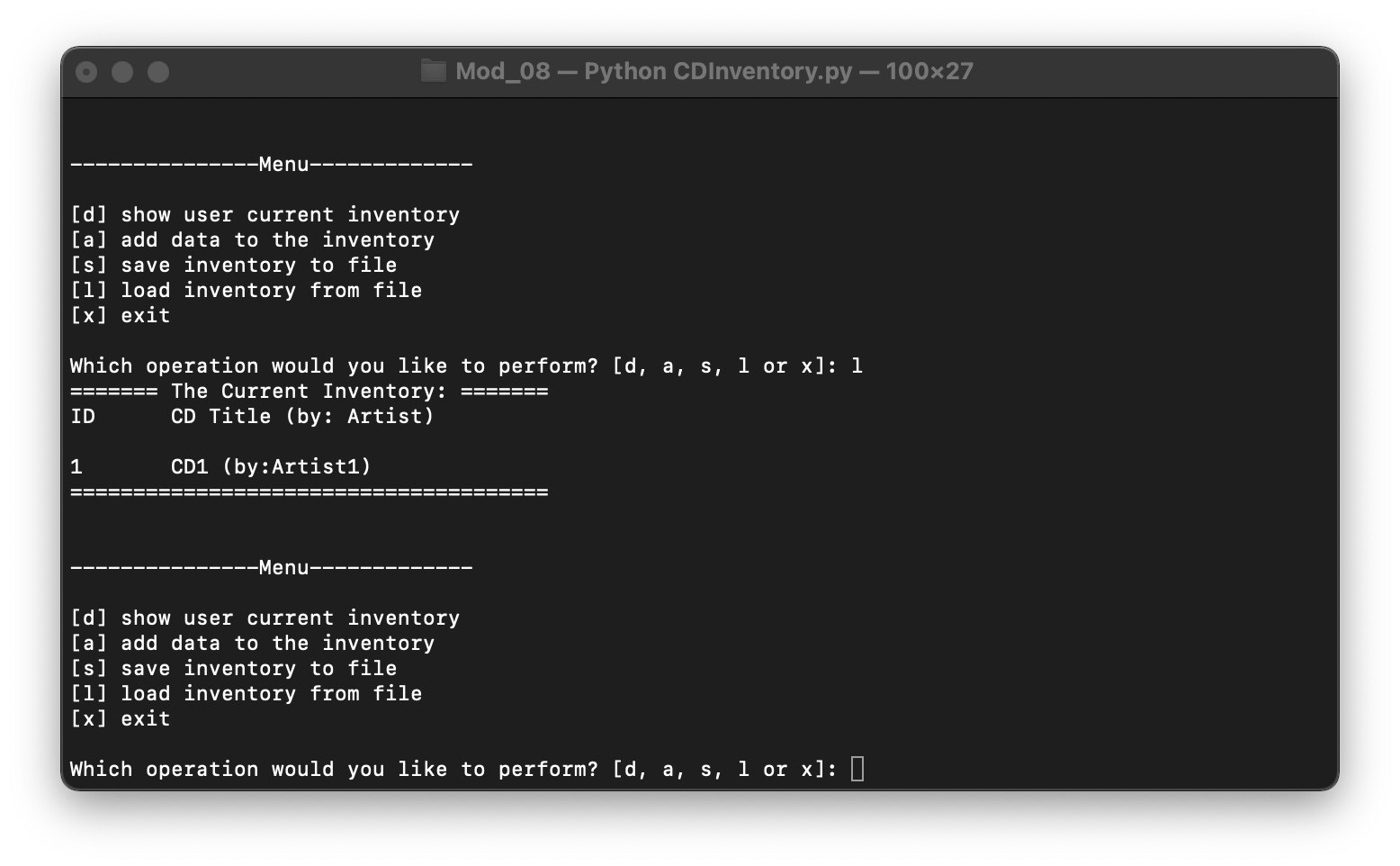


Figure - Running Results of Testing [l] in Terminal

Figure 6 below is the testing result in the text file of adding one line of CD and saving it to file.

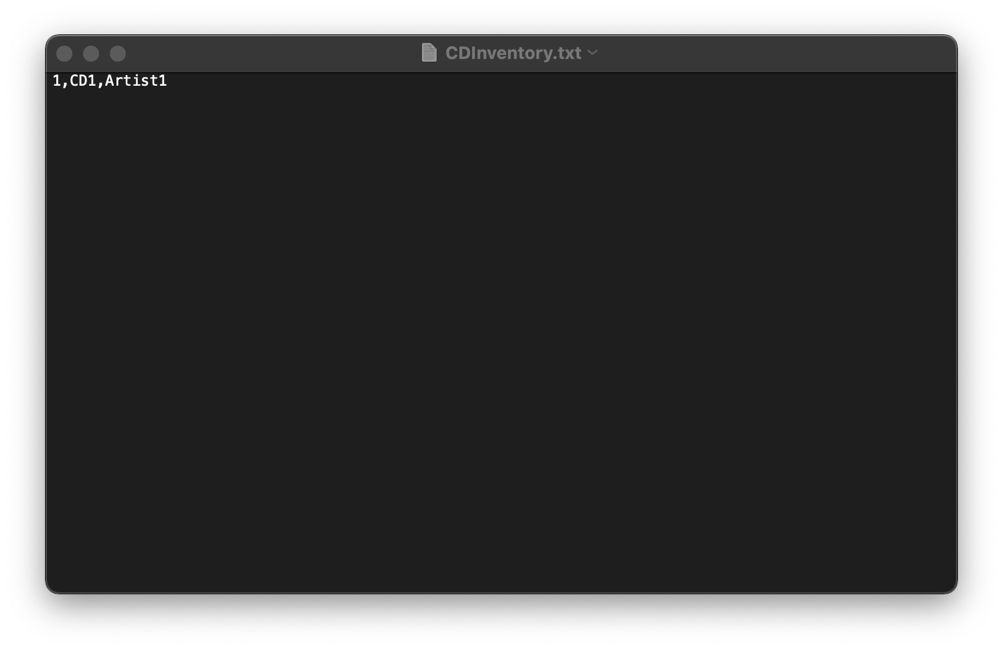


Figure - Testing Results of CDInventory.txt

# Summary

In this assignment, I applied the thought of object oriented programming and created several classes to do the tasks required. I created different class to define objects, write methods and work with properties. Aside from that, I restricted access to several objects’ attributes.

# Appendix

# Listing CDInventory.py

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. # HongyiZhu, 2021-Nov-21, Added Contents into File
7. # HongyiZhu, 2021-Nov-26, Added Try-Except Constructs into File
8. # DBiesinger, 2030-Jan-01, added pseudocode to complete assignment 08
9. # HongyiZhu, 2021-Dec-04, Modified Contents in File
10. #------------------------------------------#
12. # -- DATA -- #
13. strFileName = "CDInventory.txt"
14. lstOfCDObjects = []
15. class CD:
16. """Stores data about a CD:"""
17. def \_\_init\_\_(self,cd\_id,cd\_title,cd\_artist):
18. """Method to store Args to attributes.
20. Args:
21. cd\_id(integer): the ID of the CD;
22. cd\_title(string): the title of the CD;
23. cd\_artist(string): the artist name of the CD.
25. Returns:
26. None
27. """
28. self.\_\_cd\_id = cd\_id
29. self.\_\_cd\_title = cd\_title
30. self.\_\_cd\_artist = cd\_artist
32. @property
33. def oneCD(self):
34. """Function to property.
36. Args:
37. None
38. Returns:
39. cd(dict): use the format of dict to represent a CD.
40. """
41. return {'ID':self.\_\_cd\_id,'Title':self.\_\_cd\_title,'Artist':self.\_\_cd\_artist}
43. class FileIO:
44. """Processing the data to and from text file"""
45. @staticmethod
46. def load\_inventory(file\_name):
47. """Method to manage data ingestion from file to a list of dictionaries
49. Reads the data from file identified by file\_name into a 2D table
50. (list of dicts) table one line in the file represents one dictionary row in table.
52. If the file gets removed or does not exist at all, then the “FileNotFound” error will occur and the program will return to menu.
54. In the first column of CDInventory.txt text file, if the ID number here is not an integer, int(data[0]) will have the “ValueError”.
55. “Try-except” construction can handle the error and print “invalid line, please modify the ID as integer”, then return to menu.
57. Args:
58. file\_name (string): name of file used to read the data from cdInventory.txt
59. Returns:
60. table (list of dicts): a list of dicts which read from
61. """
62. table = []
63. try:
64. objFile = open(file\_name, 'r')
65. for line in objFile:
66. data = line.strip().split(',')
67. dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
68. table.append(dicRow)
69. except ValueError:
70. print('The ID is not an integer, please correct it in txt file of '+ file\_name)
71. objFile.close()
72. return table
73. except FileNotFoundError:
74. print('This file ' + file\_name + ' does not exist,will create blank ' + file\_name)
75. objFile = open(file\_name, 'w')
76. objFile.close()
77. return table
78. objFile.close()
79. return table
81. @staticmethod
82. def save\_inventory(file\_name, table):
83. """Function to write the table to the file.
85. Args:
86. file\_name (string): an string which is the file name that the user want to use for saving the data.
87. table (list of dicts): a list of dicts which is the existed CDInventory
88. Returns:
89. None.
90. """
92. objFile = open(file\_name, 'w')
93. for row in table:
94. lstValues = list(row.values())
95. lstValues[0] = str(lstValues[0])
96. objFile.write(','.join(lstValues) + '\n')
98. objFile.close()
100. # -- PRESENTATION (Input/Output) -- #
101. class IO:
102. @staticmethod
103. def show\_menu():
104. """show menu to user
106. Args:
107. None.
109. Returns:
110. None.
111. """
112. print('\n\n---------------Menu-------------\n\n[d] show user current inventory\n[a] add data to the inventory\n[s] save inventory to file')
113. print('[l] load inventory from file\n[x] exit\n')
115. @staticmethod
116. def menu\_choice():
117. """captures user's choice
119. Args:
120. None.
122. Returns:
123. choice (string): a lower case sting of the users input out of the choices d, a, s, l or x
125. """
126. choice = ' '
127. while choice not in ['d', 'a', 's', 'l', 'x']:
128. choice = input('Which operation would you like to perform? [d, a, s, l or x]: ').lower().strip()
129. return choice
131. @staticmethod
132. def show\_inventory(table):
133. """display the current data on screen

136. Args:
137. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
139. Returns:
140. None.
142. """
143. print('======= The Current Inventory: =======')
144. print('ID\tCD Title (by: Artist)\n')
145. for row in table:
146. print('{}\t{} (by:{})'.format(\*row.values()))
147. print('======================================')

150. @staticmethod
151. def get\_cd():
152. """get CD data from user

155. Args:
156. None
158. Returns:
159. cd (dict):a dict like {'ID':1,'Title':'CD title','Artist':'CD artist'}
161. """
162. while True:
163. intID = input("input an integer as CD ID:")
164. if intID.isdigit():
165. intID = int(intID)
166. break
167. else:
168. print("please input an integer!")
169. strTitle = input("input CD title:").strip()
170. strArtist = input("input CD artist:").strip()
171. cd = CD(intID,strTitle,strArtist).oneCD
172. return cd
174. @staticmethod
175. def add\_inventory(lstCD,cd):
176. """add data to the inventory

179. Args:
180. lstCD (list of dict):a list of dicts which is the existed CDInventory
181. cd (dict):a dict like {'ID':1,'Title':'CD title','Artist':'CD artist'} ,will add to lstCD
182. Returns:
183. None
185. """
186. lstCD.append(cd)

189. # -- Main Body of Script -- #
190. lstOfCDObjects = FileIO.load\_inventory(strFileName)
191. while True:
192. IO.show\_menu()
193. strChoice = IO.menu\_choice()
194. if strChoice == 'x':
195. break
196. if strChoice == 'd':
197. IO.show\_inventory(lstOfCDObjects)
198. continue
199. if strChoice == 'a':
200. cd = IO.get\_cd()
201. IO.add\_inventory(lstOfCDObjects,cd)
202. IO.show\_inventory(lstOfCDObjects)
203. continue
204. if strChoice == 'l':
205. lstOfCDObjects = FileIO.load\_inventory(strFileName)
206. IO.show\_inventory(lstOfCDObjects)
207. continue
208. print("WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.")
209. strYesNo = input("type \'yes\' to continue and reload from file. otherwise reload will be canceled")
210. if strYesNo.lower() == "yes":
211. print("reloading...")
212. lstOfCDObjects = FileIO.load\_inventory(strFileName)
213. else:
214. input("canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.")
215. IO.show\_inventory(lstOfCDObjects)
216. continue
217. if strChoice == 's':
218. FileIO.save\_inventory(strFileName,lstOfCDObjects)
219. IO.show\_inventory(lstOfCDObjects)
220. strYesNo = input("Save this inventory to file? [y/n] ").strip().lower()
221. if strYesNo == 'y':
222. FileIO.save\_inventory(strFileName,lstOfCDObjects)
223. else:
224. input("The inventory was NOT saved to file. Press [ENTER] to return to the menu.")
225. continue

1. Retrieved 2021-Dec-04 [↑](#footnote-ref-1)