Bill McG… (truncated for public sharing)

March 6, 2022

Foundations of Programming: Python

Assignment 06

GitHub Link: https://github.com/wwm787/Assignment\_06

# Introduction to Classes and Functions in Python

# Introduction

In this assignment and lab, I learned how to use functions to move individual sequential portions of IO and Processing Code into functions. Some functions pass data and other functions return data such as the function for inventory. Additional items learned while working on this assignment include the difference between parameters and arguments, definition of a return value, global vs local variables, shadowing and how functions and classes differ. Details are contained in the summary below.

# Assignment 06 – CD Inventory Program Using Functions and Classes

In assignment 06, the script allows the user to inventory a CD collection. The program asks the user to either load data from a text file, add item to memory, view inventory, delete an item from memory, save to text file or exit the program. It is like assignment 05, except the IO and data processing are transferred to functions.

* Line 1: Import sys library to gain access to exit() function
* Lines 4 – 8: Define variables, list of lists, dictionary and text file name
* Lines 11 – 55: Group data processing functions into a class
  + Functions
    - Add a record containing ID, Title and Artist parameter and loads it into table
    - Delete a record based on ID parameter, called when user selects ‘d’ in menu
* Lines 57 – 106: Group text file read and write functions into a class
  + Functions
    - Read text file that is called at program startup and when inventory is selected in menu
    - Write to text file that is called when save is selected from the menu
* Lines 111 – 176: Group IO functions into a class
  + Functions
    - Display menu at startup
    - Get user menu choice
    - Show current inventory in memory (not saved to text file)
    - Add function when user selects ‘a’ in program menu that allows the user to enter a new CD ID, Title and Artist via input command
* Line 179: Load currently saved inventory from text file
* Lines 182 – 230: Series of if, elif and else statements to process menu items and call the specified data processing, read/write or IO functions
  + Choices are the exit the program, load from text file, view inventory, add a record, delete a record or save to text file.

1. **import** sys
3. # -- DATA -- #
4. strChoice **=** "" # User input
5. lstTbl **=** []  # list of lists to hold data
6. dicRow **=** {}  # list of data row
7. strFileName **=** "CDInventory.txt"  # data storage file
8. objFile **=** None  # file object
10. # -- PROCESSING -- #
11. **class** DataProcessor:
13. @staticmethod
14. **def** myAddProcCode(myID, myTitle, myArtist):
15. """ Function to process ID, Title and Artist
17. Args:
18. myID (string): ID of CD.
19. myTitle (string): Title of CD.
20. myArtist (string): Artist name.
22. Returns:
23. None.
25. """
26. # Add item to the table
27. intID **=** int(myID)
28. dicRow **=** {"ID": intID, "Title": myTitle, "Artist": myArtist}
29. lstTbl.append(dicRow)
30. IO.show\_inventory(lstTbl)
31. **def** myDeleteDataProcFunc(intIDDelReceived):
32. """ Function to delete CD based on ID passed to function
34. Args:
35. intIDDelReceived (int): ID of CD to delete.
37. Returns:
38. None.
40. """
41. # search thru table and delete CD
42. intRowNr **=** **-**1
43. blnCDRemoved **=** False
44. **for** row **in** lstTbl:
45. intRowNr **+=** 1
46. **if** row["ID"] **==** intIDDelReceived:
47. **del** lstTbl[intRowNr]
48. blnCDRemoved **=** True
49. **break**
50. **if** blnCDRemoved:
51. print("The CD was removed")
52. **else**:
53. print("Could not find this CD!")
54. IO.show\_inventory(lstTbl) # display inventory
55. **return**
57. **class** FileProcessor:
58. """Processing the data to and from text file"""
60. @staticmethod
61. **def** read\_file(file\_name, startTbl):
62. """Function to manage data ingestion from file to a list of dictionaries
64. Reads the data from file identified by file\_name into a 2D table
65. (list of dicts) table one line in the file represents one dictionary row in table.
67. Args:
68. file\_name (string): name of file used to read the data from
69. table (list of dictionary): 2D data structure (list of dicts) that holds the data during runtime
71. Returns:
72. None.
73. """
74. **try**:
75. startTbl.clear()  # this clears existing data and allows to load data from file
76. objFile **=** open(file\_name, "r")
77. **for** line **in** objFile:
78. data **=** line.strip().split(",")
79. dicRow **=** {"ID": int(data[0]), "Title": data[1], "Artist": data[2]}
80. startTbl.append(dicRow)
81. objFile.close()
82. **except** Exception as e:
83. print("\nYou need to create a CDInventory.txt file first. \n")
84. print(e)
85. print("Exiting Program\n")
86. sys.exit()
88. @staticmethod
89. **def** write\_file(file\_name, recTbl): #save data
90. """ Function to save table data to text file
92. Args:
93. file\_name (string): name of the file used to write data to.
94. recTbl (list of dictionary): 2D data structure (list of dicts) that holds the data during runtime.
96. Returns:
97. None.
99. """
100. # Save to text file
101. objFile **=** open(file\_name, "w")
102. **for** row **in** recTbl: # Parse each row
103. lstValues **=** list(row.values())
104. lstValues[0] **=** str(lstValues[0])
105. objFile.write(",".join(lstValues) **+** "\n")
106. objFile.close()

109. # -- PRESENTATION (Input/Output) -- #
111. **class** IO:
112. """Handling Input / Output"""
114. @staticmethod
115. **def** print\_menu():
116. """Displays a menu of choices to the user
118. Args:
119. None.
121. Returns:
122. None.
123. """
125. print("Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory")
126. print("[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n")
128. @staticmethod
129. **def** menu\_choice():
130. """Gets user input for menu selection
132. Args:
133. None.
135. Returns:
136. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
138. """
139. choice **=** " "
140. **while** choice **not** **in** ["l", "a", "i", "d", "s", "x"]:
141. choice **=** input("Which operation would you like to perform? [l, a, i, d, s or x]: ").lower().strip()
142. print()  # Add extra space for layout
143. **return** choice
145. @staticmethod
146. **def** show\_inventory(invTbl):
147. """Displays current inventory of table invTbl

150. Args:
151. invTbl (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
153. Returns:
154. None.
156. """
157. print("======= The Current Inventory: =======")
158. print("ID\tCD Title (by: Artist)\n")
159. **for** row **in** invTbl:
160. print("{}\t{} (by:{})".format(**\***row.values()))
161. print("======================================")
162. @staticmethod
163. **def** myAddIOFunc():
164. """ Function for input / ouput
165. Ask the user CD ID, Title and Artist
167. Returns:
168. strID1 (string): User inputted CD ID.
169. strTitle1 (string): User inputted CD Title.
170. strArtist1 (string): User inputted CD Artist.
172. """
173. strID1 **=** input("Enter ID: ").strip()
174. strTitle1 **=** input("What is the CD\"s title? ").strip()
175. strArtist1 **=** input("What is the Artist\"s name? ").strip()
176. **return** strID1, strTitle1, strArtist1
178. # When program starts, read in the currently saved Inventory
179. FileProcessor.read\_file(strFileName, lstTbl)
181. # Start main loop
182. **while** True:
183. # Display Menu to user and get choice
184. IO.print\_menu()
185. strChoice **=** IO.menu\_choice()
186. # Process menu selection
187. # Process exit first
188. **if** strChoice **==** "x":
189. **break**
190. # Process load inventory
191. **if** strChoice **==** "l":
192. print("WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.")
193. strYesNo **=** input("type \"yes\" to continue and reload from file. otherwise reload will be canceled: ")
194. **if** strYesNo.lower() **==** "yes":
195. print("reloading...")
196. FileProcessor.read\_file(strFileName, lstTbl)
197. IO.show\_inventory(lstTbl)
198. **else**:
199. input("canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.")
200. IO.show\_inventory(lstTbl)
201. **continue**  # start loop back at top.
202. # Process add a CD
203. **elif** strChoice **==** "a":
204. # Ask user for new ID, CD Title and Artist
205. strID, strTitle, strArtist **=** IO.myAddIOFunc()
206. DataProcessor.myAddProcCode(strID, strTitle, strArtist)
207. **continue**  # start loop back at top.
208. # Process display current inventory
209. **elif** strChoice **==** "i":
210. IO.show\_inventory(lstTbl)
211. **continue**  # start loop back at top.
212. **elif** strChoice **==** "d": # process delete a CD
213. IO.show\_inventory(lstTbl) # display inventory
214. # Get Userinput for which CD to delete
215. intIDDelInput **=** int(input("Which ID would you like to delete? ").strip())
216. DataProcessor.myDeleteDataProcFunc(intIDDelInput)
217. **continue**  # start loop back at top.
218. **elif** strChoice **==** "s": # process save inventory to file
219. # Display current inventory and ask user for confirmation to save
220. IO.show\_inventory(lstTbl)
221. strYesNo **=** input("Save this inventory to file? [y/n] ").strip().lower()
222. # Process choice
223. **if** strYesNo **==** "y":
224. FileProcessor.write\_file(strFileName, lstTbl)
225. **else**:
226. input("The inventory was NOT saved to file. Press [ENTER] to return to the menu.")
227. **continue**  # start loop back at top.
228. # Catch-all should not be possible, as user choice gets vetted in IO, but to be save:
229. **else**:
230. print("General Error")

Listing - CDInventory.py code

|  |  |
| --- | --- |
| Text  Description automatically generated  Figure - CDInventory.py Spyder Demo | Text  Description automatically generated  Figure - CDInventory.py Spyder Demo |

|  |  |
| --- | --- |
| Text  Description automatically generated  Figure - CDInventory.py Terminal Window Demo | Text  Description automatically generated  Figure - CDInventory.py Terminal Window Demo |

# LAB 06-A: Working with Files and Lists:

Lab 06-A works definition four functions for calculating Sum, Difference, Product and Quotient of two numbers. For each task a function is created with two parameters corresponding to the two numbers. The two numbers are passed as arguments from with the code via a function call. Then functions them process the simple math and return the results.

# LAB 06-B: Working with Files and Dictionaries:

To modify Lab 06-A so all four calculations are done in one function. Move all calculations under one function, assign each a variable. Then return each variable as a list.

# Summary

Regarding function paraments vs arguments, parameters are the variables inside the function definition, while arguments are the variables passed when the function is called from within the code. The return value is a variable returned by a function after it completes a task such as a calculation. For example, a function testBill() has a task to calculate 4 divided by 2 with the result assigned to variable testBillResult. When function testBill() is called within the code in a print statement, then 2 would be returned.

Global variables are variables defined outside of the function who value can be called anywhere within the code. Whereas a local variable is defined within a function and only accessible from within the function it is defined in. Shadowing is when a local variable has the same name as a global variable. This is not advised as it is confusing to the user.

Finally, for this module, classes are a way to group several functions with a similar purpose together.

# Appendix

## Listing CDInventory.py

1. #------------------------------------------#
2. # Title: CDInventory.py
3. # Desc: Working with classes and functions.
4. # Change Log: (Who, When, What)
5. # DBiesinger, 2030-Jan-01, Created File
6. # Bill McG…, 2022-Mar-06, Modified File to add functionality
7. #------------------------------------------#
9. **import** sys
11. # -- DATA -- #
12. strChoice **=** "" # User input
13. lstTbl **=** []  # list of lists to hold data
14. dicRow **=** {}  # list of data row
15. strFileName **=** "CDInventory.txt"  # data storage file
16. objFile **=** None  # file object
18. # -- PROCESSING -- #
19. **class** DataProcessor:
21. @staticmethod
22. **def** myAddProcCode(myID, myTitle, myArtist):
23. """ Function to process ID, Title and Artist
25. Args:
26. myID (string): ID of CD.
27. myTitle (string): Title of CD.
28. myArtist (string): Artist name.
30. Returns:
31. None.
33. """
34. # Add item to the table
35. intID **=** int(myID)
36. dicRow **=** {"ID": intID, "Title": myTitle, "Artist": myArtist}
37. lstTbl.append(dicRow)
38. IO.show\_inventory(lstTbl)
39. **def** myDeleteDataProcFunc(intIDDelReceived):
40. """ Function to delete CD based on ID passed to function
42. Args:
43. intIDDelReceived (int): ID of CD to delete.
45. Returns:
46. None.
48. """
49. # search thru table and delete CD
50. intRowNr **=** **-**1
51. blnCDRemoved **=** False
52. **for** row **in** lstTbl:
53. intRowNr **+=** 1
54. **if** row["ID"] **==** intIDDelReceived:
55. **del** lstTbl[intRowNr]
56. blnCDRemoved **=** True
57. **break**
58. **if** blnCDRemoved:
59. print("The CD was removed")
60. **else**:
61. print("Could not find this CD!")
62. IO.show\_inventory(lstTbl) # display inventory
63. **return**
65. **class** FileProcessor:
66. """Processing the data to and from text file"""
68. @staticmethod
69. **def** read\_file(file\_name, startTbl):
70. """Function to manage data ingestion from file to a list of dictionaries
72. Reads the data from file identified by file\_name into a 2D table
73. (list of dicts) table one line in the file represents one dictionary row in table.
75. Args:
76. file\_name (string): name of file used to read the data from
77. table (list of dictionary): 2D data structure (list of dicts) that holds the data during runtime
79. Returns:
80. None.
81. """
82. **try**:
83. startTbl.clear()  # this clears existing data and allows to load data from file
84. objFile **=** open(file\_name, "r")
85. **for** line **in** objFile:
86. data **=** line.strip().split(",")
87. dicRow **=** {"ID": int(data[0]), "Title": data[1], "Artist": data[2]}
88. startTbl.append(dicRow)
89. objFile.close()
90. **except** Exception as e:
91. print("\nYou need to create a CDInventory.txt file first. \n")
92. print(e)
93. print("Exiting Program\n")
94. sys.exit()
96. @staticmethod
97. **def** write\_file(file\_name, recTbl): #save data
98. """ Function to save table data to text file
100. Args:
101. file\_name (string): name of the file used to write data to.
102. recTbl (list of dictionary): 2D data structure (list of dicts) that holds the data during runtime.
104. Returns:
105. None.
107. """
108. # Save to text file
109. objFile **=** open(file\_name, "w")
110. **for** row **in** recTbl: # Parse each row
111. lstValues **=** list(row.values())
112. lstValues[0] **=** str(lstValues[0])
113. objFile.write(",".join(lstValues) **+** "\n")
114. objFile.close()

117. # -- PRESENTATION (Input/Output) -- #
119. **class** IO:
120. """Handling Input / Output"""
122. @staticmethod
123. **def** print\_menu():
124. """Displays a menu of choices to the user
126. Args:
127. None.
129. Returns:
130. None.
131. """
133. print("Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory")
134. print("[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n")
136. @staticmethod
137. **def** menu\_choice():
138. """Gets user input for menu selection
140. Args:
141. None.
143. Returns:
144. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
146. """
147. choice **=** " "
148. **while** choice **not** **in** ["l", "a", "i", "d", "s", "x"]:
149. choice **=** input("Which operation would you like to perform? [l, a, i, d, s or x]: ").lower().strip()
150. print()  # Add extra space for layout
151. **return** choice
153. @staticmethod
154. **def** show\_inventory(invTbl):
155. """Displays current inventory of table invTbl

158. Args:
159. invTbl (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
161. Returns:
162. None.
164. """
165. print("======= The Current Inventory: =======")
166. print("ID\tCD Title (by: Artist)\n")
167. **for** row **in** invTbl:
168. print("{}\t{} (by:{})".format(**\***row.values()))
169. print("======================================")
170. @staticmethod
171. **def** myAddIOFunc():
172. """ Function for input / ouput
173. Ask the user CD ID, Title and Artist
175. Returns:
176. strID1 (string): User inputted CD ID.
177. strTitle1 (string): User inputted CD Title.
178. strArtist1 (string): User inputted CD Artist.
180. """
181. strID1 **=** input("Enter ID: ").strip()
182. strTitle1 **=** input("What is the CD\"s title? ").strip()
183. strArtist1 **=** input("What is the Artist\"s name? ").strip()
184. **return** strID1, strTitle1, strArtist1
186. # When program starts, read in the currently saved Inventory
187. FileProcessor.read\_file(strFileName, lstTbl)
189. # Start main loop
190. **while** True:
191. # Display Menu to user and get choice
192. IO.print\_menu()
193. strChoice **=** IO.menu\_choice()
194. # Process menu selection
195. # Process exit first
196. **if** strChoice **==** "x":
197. **break**
198. # Process load inventory
199. **if** strChoice **==** "l":
200. print("WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.")
201. strYesNo **=** input("type \"yes\" to continue and reload from file. otherwise reload will be canceled: ")
202. **if** strYesNo.lower() **==** "yes":
203. print("reloading...")
204. FileProcessor.read\_file(strFileName, lstTbl)
205. IO.show\_inventory(lstTbl)
206. **else**:
207. input("canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.")
208. IO.show\_inventory(lstTbl)
209. **continue**  # start loop back at top.
210. # Process add a CD
211. **elif** strChoice **==** "a":
212. # Ask user for new ID, CD Title and Artist
213. strID, strTitle, strArtist **=** IO.myAddIOFunc()
214. DataProcessor.myAddProcCode(strID, strTitle, strArtist)
215. **continue**  # start loop back at top.
216. # Process display current inventory
217. **elif** strChoice **==** "i":
218. IO.show\_inventory(lstTbl)
219. **continue**  # start loop back at top.
220. **elif** strChoice **==** "d": # process delete a CD
221. IO.show\_inventory(lstTbl) # display inventory
222. # Get Userinput for which CD to delete
223. intIDDelInput **=** int(input("Which ID would you like to delete? ").strip())
224. DataProcessor.myDeleteDataProcFunc(intIDDelInput)
225. **continue**  # start loop back at top.
226. **elif** strChoice **==** "s": # process save inventory to file
227. # Display current inventory and ask user for confirmation to save
228. IO.show\_inventory(lstTbl)
229. strYesNo **=** input("Save this inventory to file? [y/n] ").strip().lower()
230. # Process choice
231. **if** strYesNo **==** "y":
232. FileProcessor.write\_file(strFileName, lstTbl)
233. **else**:
234. input("The inventory was NOT saved to file. Press [ENTER] to return to the menu.")
235. **continue**  # start loop back at top.
236. # Catch-all should not be possible, as user choice gets vetted in IO, but to be save:
237. **else**:
238. print("General Error")