Gavin Chan

02/28/2020

IT FDN 100

Assignment 06

# Introduction

In this week’s module and assignment, functions were introduced as a very efficient way to build upon provided code. Class of functions were also utilized to organize functions by their objectives.

# Lab 06-A: Working with Functions

Lab 06-A has the student take a piece of sample code provided by the instructor, but modify it to work with passing values into the function and returning the result instead. Figure 1 is a snippet of the code, and Figure 2 is a snippet of the output.

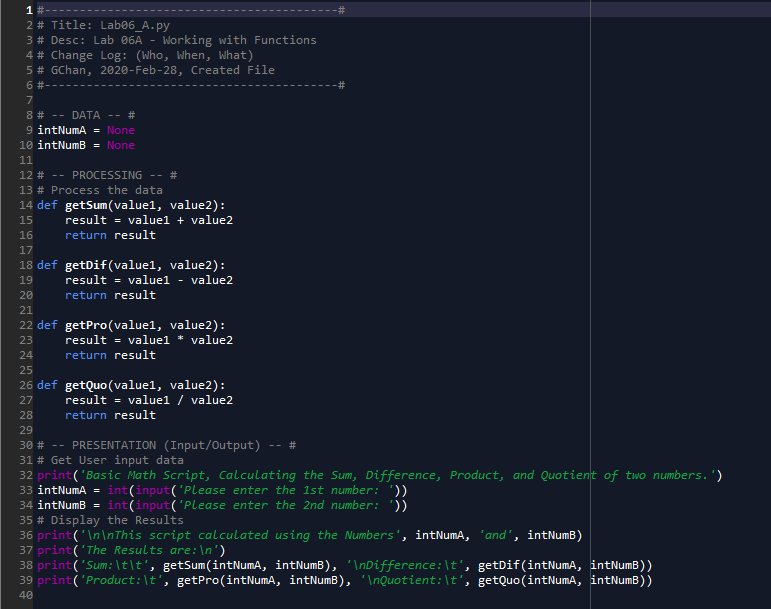


Figure 1 - Snippet of Code for Lab 06A

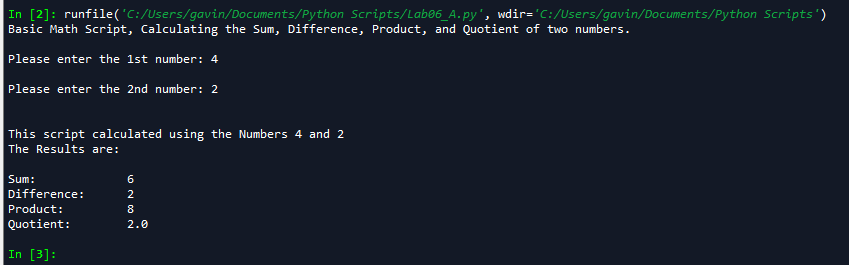


Figure 2 - Output for Code for Lab 06A

The code is modified so the function calls for two variables, initially named as value1 and value 2 but later identified as intNumA and intNumB. The functions take the two variables, apply the corresponding mathematical function, and returns the result. The advantage with this method is that the same functions can be applied later in the code, but could use different variables rather than intNumA and intNumB as defined in the pre-modified code.

## Lab 06-B: Returning Tuples

The code in Lab 06-A is then modified so one function does all of the math, rather than four separate functions. The results are put into a list, and the individual parts of the list are called upon when printing results. Figure 3 is a snippet of the code, and Figure 4 is the output.

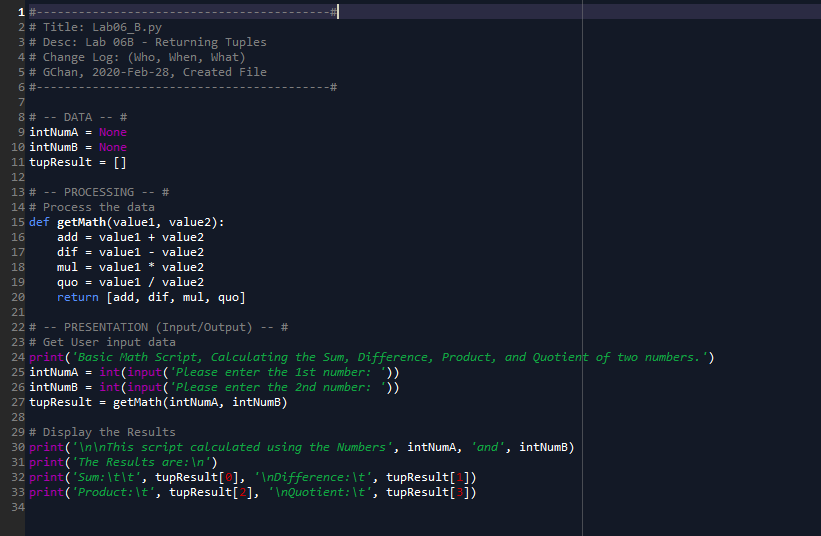


Figure 3 - Lab 06-B Code

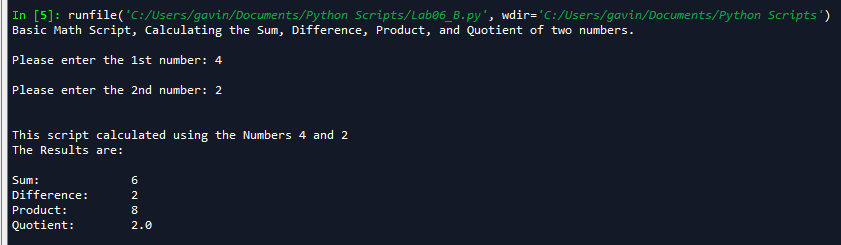
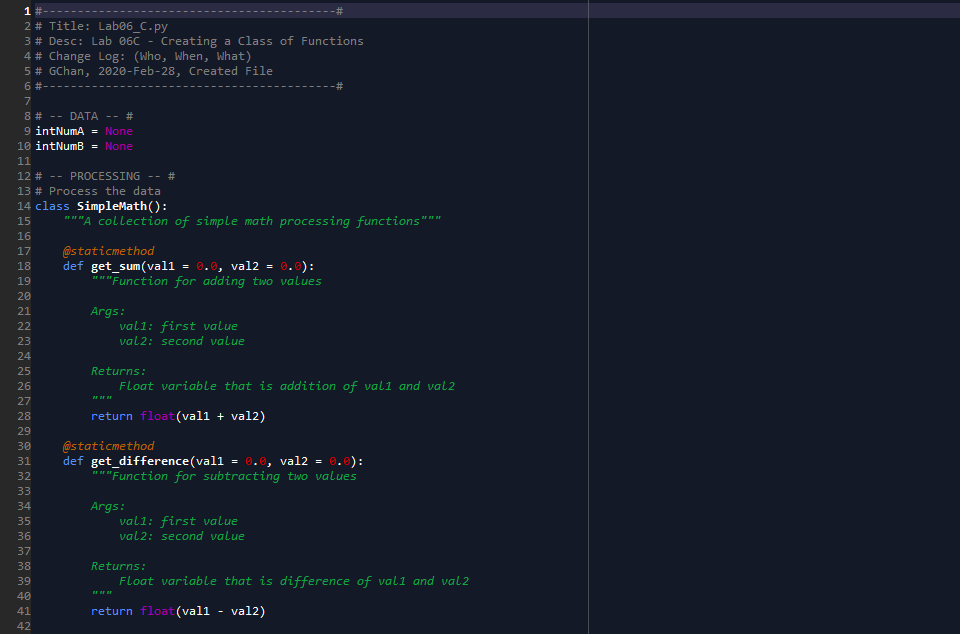


Figure 4 - Output for Lab 06-B'S Code

The functions are condensed into one function, as seen in Lines #15-20. Line #27 takes the results and puts them into a list, which can be easily accessed as seen in Lines #32-33.

## Lab 06-C: Creating a Class of Functions

For Lab 06-C, the code from Lab 06-A is modified to include docstring, as well as include the four math functions within one class, called “SimpleMath”. Docstring is a useful way to organize one’s code and make it easier for a peer to understand. Figure 5 is a snippet of the code, and Figure 6 is the output.



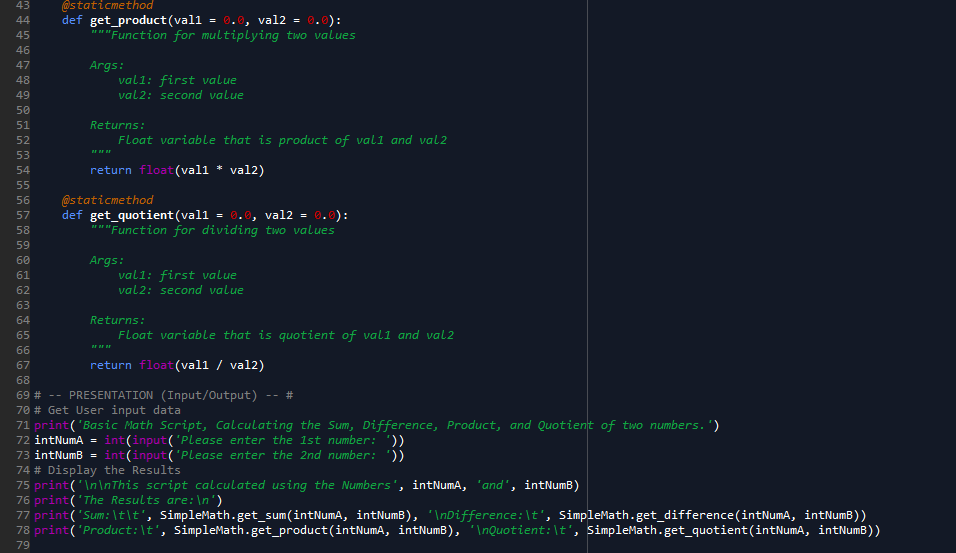


Figure 5 - Code for Lab 06-C

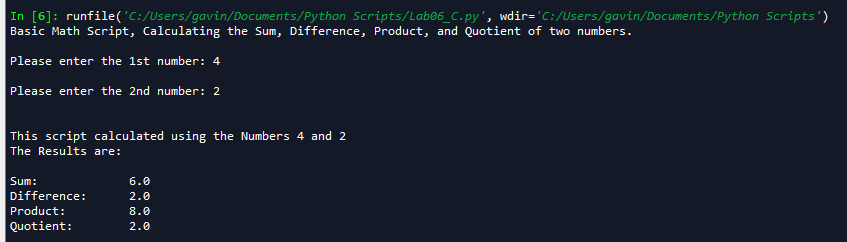


Figure 6 - Output for Code for Lab 06-C

The code begins by creating a class for all of the functions (Line #14). From there, different functions are created, with docstring below each header as an explanation for what is going on within the code. In Lines #77-78, the way that the function is called upon is different than the other two labs, due to the functions being part of a class.

# Assignment 06: Modifying a Starter Code

For Assignment 6, we were given a piece of code with several “TODO’s” within the script. The objective was to modify the code in order to utilize functions to perform much of the input/output and processing. Because the script is too long, the actual code can be found in the Appendix at the end of this document. Figures 7-9 shows a few snippets of some of the example functions run in the code.

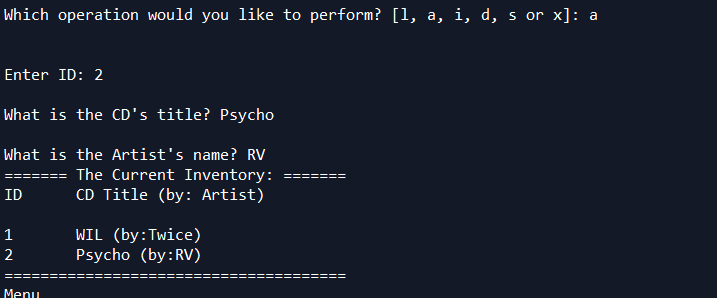


Figure 7 - Adding a CD

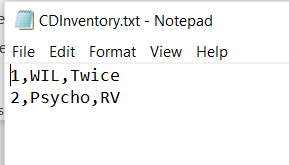


Figure 8 - Saving the CD's to a .txt File

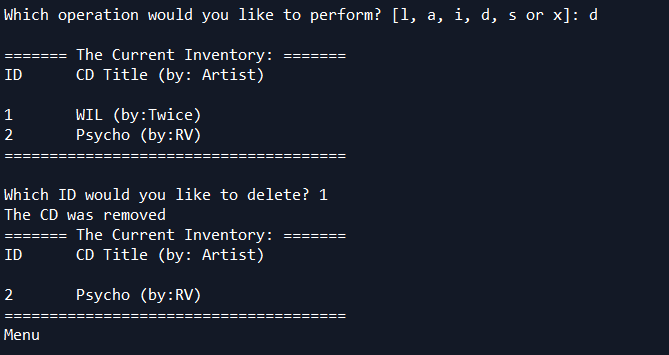


Figure 9 - Deleting a CD

One flaw with the code is that the txt file, which is “CDInventory.txt” in this case, needs to be created before the code can be run. Although I did not personally fix this flaw, one possible way to approach this issue could be to use a “try.. except” loop. Overall, this assignment involved a lot of moving around lines of code in order to improve efficiency, if the script were to be expanded upon.

# Saving the Python File

The python script was saved in the following directory:

C:\\_FDNProgramming\Assignment06

The code itself was named CDInventory.py. The text file with the results is named CDInventory.txt, as named by the script.

The link to the Github is as follows:

<https://github.com/NotGavin/Assignment_05/blob/master/CDInventory.py>

# Summary

In this module, we learned about the importance of functions as well as classes of functions. Functions are a very useful way to make code look a lot cleaner and more organized. The student was again tasked with modifying a starter code; the assignment involved a lot of copy/pasting in order to move lines of code around to build the functions.

# Appendix

Lab 06-A Script:

1. #------------------------------------------#
2. # Title: Lab06\_A.py
3. # Desc: Lab 06A - Working with Functions
4. # Change Log: (Who, When, What)
5. # GChan, 2020-Feb-28, Created File
6. #------------------------------------------#
8. # -- DATA -- #
9. intNumA = None
10. intNumB = None
12. # -- PROCESSING -- #
13. # Process the data
14. **def** getSum(value1, value2):
15. result = value1 + value2
16. **return** result
18. **def** getDif(value1, value2):
19. result = value1 - value2
20. **return** result
22. **def** getPro(value1, value2):
23. result = value1 \* value2
24. **return** result
26. **def** getQuo(value1, value2):
27. result = value1 / value2
28. **return** result
30. # -- PRESENTATION (Input/Output) -- #
31. # Get User input data
32. **print**('Basic Math Script, Calculating the Sum, Difference, Product, and Quotient of two numbers.')
33. intNumA = int(input('Please enter the 1st number: '))
34. intNumB = int(input('Please enter the 2nd number: '))
35. # Display the Results
36. **print**('\n\nThis script calculated using the Numbers', intNumA, 'and', intNumB)
37. **print**('The Results are:\n')
38. **print**('Sum:\t\t', getSum(intNumA, intNumB), '\nDifference:\t', getDif(intNumA, intNumB))
39. **print**('Product:\t', getPro(intNumA, intNumB), '\nQuotient:\t', getQuo(intNumA, intNumB))

Lab 06-B Script:

1. #------------------------------------------#
2. # Title: Lab06\_B.py
3. # Desc: Lab 06B - Returning Tuples
4. # Change Log: (Who, When, What)
5. # GChan, 2020-Feb-28, Created File
6. #------------------------------------------#
8. # -- DATA -- #
9. intNumA = None
10. intNumB = None
11. tupResult = []
13. # -- PROCESSING -- #
14. # Process the data
15. **def** getMath(value1, value2):
16. add = value1 + value2
17. dif = value1 - value2
18. mul = value1 \* value2
19. quo = value1 / value2
20. **return** [add, dif, mul, quo]
22. # -- PRESENTATION (Input/Output) -- #
23. # Get User input data
24. **print**('Basic Math Script, Calculating the Sum, Difference, Product, and Quotient of two numbers.')
25. intNumA = int(input('Please enter the 1st number: '))
26. intNumB = int(input('Please enter the 2nd number: '))
27. tupResult = getMath(intNumA, intNumB)
29. # Display the Results
30. **print**('\n\nThis script calculated using the Numbers', intNumA, 'and', intNumB)
31. **print**('The Results are:\n')
32. **print**('Sum:\t\t', tupResult[0], '\nDifference:\t', tupResult[1])
33. **print**('Product:\t', tupResult[2], '\nQuotient:\t', tupResult[3])

Lab 06-C’s Code

1. #------------------------------------------#
2. # Title: Lab06\_C.py
3. # Desc: Lab 06C - Creating a Class of Functions
4. # Change Log: (Who, When, What)
5. # GChan, 2020-Feb-28, Created File
6. #------------------------------------------#
8. # -- DATA -- #
9. intNumA = None
10. intNumB = None
12. # -- PROCESSING -- #
13. # Process the data
14. **class** SimpleMath():
15. """A collection of simple math processing functions"""
17. @staticmethod
18. **def** get\_sum(val1 = 0.0, val2 = 0.0):
19. """Function for adding two values
21. Args:
22. val1: first value
23. val2: second value
25. Returns:
26. Float variable that is addition of val1 and val2
27. """
28. **return** float(val1 + val2)
30. @staticmethod
31. **def** get\_difference(val1 = 0.0, val2 = 0.0):
32. """Function for subtracting two values
34. Args:
35. val1: first value
36. val2: second value
38. Returns:
39. Float variable that is difference of val1 and val2
40. """
41. **return** float(val1 - val2)
43. @staticmethod
44. **def** get\_product(val1 = 0.0, val2 = 0.0):
45. """Function for multiplying two values
47. Args:
48. val1: first value
49. val2: second value
51. Returns:
52. Float variable that is product of val1 and val2
53. """
54. **return** float(val1 \* val2)
56. @staticmethod
57. **def** get\_quotient(val1 = 0.0, val2 = 0.0):
58. """Function for dividing two values
60. Args:
61. val1: first value
62. val2: second value
64. Returns:
65. Float variable that is quotient of val1 and val2
66. """
67. **return** float(val1 / val2)
69. # -- PRESENTATION (Input/Output) -- #
70. # Get User input data
71. **print**('Basic Math Script, Calculating the Sum, Difference, Product, and Quotient of two numbers.')
72. intNumA = int(input('Please enter the 1st number: '))
73. intNumB = int(input('Please enter the 2nd number: '))
74. # Display the Results
75. **print**('\n\nThis script calculated using the Numbers', intNumA, 'and', intNumB)
76. **print**('The Results are:\n')
77. **print**('Sum:\t\t', SimpleMath.get\_sum(intNumA, intNumB), '\nDifference:\t', SimpleMath.get\_difference(intNumA, intNumB))
78. **print**('Product:\t', SimpleMath.get\_product(intNumA, intNumB), '\nQuotient:\t', SimpleMath.get\_quotient(intNumA, intNumB))

Assignment 06’s Code

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. #------------------------------------------#
8. # -- DATA -- #
9. strChoice = '' # User input
10. lstTbl = []  # list of lists to hold data
11. dicRow = {}  # list of data row
12. strFileName = 'CDInventory.txt'  # data storage file
13. objFile = None  # file object

16. # -- PROCESSING -- #
17. **class** DataProcessor:
18. @staticmethod
19. **def** add\_CD(ID, Title, Artist, Tbl):
20. """
21. Function to add CD to list
23. Args:
24. ID: CD ID that will get converted into int
25. Title: CD Title
26. Artist: CD Artist
27. Tbl: list to hold data
29. Returns:
30. None.
31. """
32. intID = int(ID)
33. dicRow = {'ID': intID, 'Title': Title, 'Artist': Artist}
34. Tbl.append(dicRow)
36. @staticmethod
37. **def** del\_CD(ID, Tbl):
38. """
39. Function to delete CD from existing table, if specified
41. Args:
42. ID: CD ID that will be deleted
43. Tbl: list to hold data
45. Returns:
46. None.
47. """
48. intRowNr = -1
49. blnCDRemoved = False
50. **for** row **in** Tbl:
51. intRowNr += 1
52. **if** row['ID'] == intIDDel:
53. **del** lstTbl[intRowNr]
54. blnCDRemoved = True
55. **break**
56. **if** blnCDRemoved:
57. **print**('The CD was removed')
58. **else**:
59. **print**('Could not find this CD!')

62. **class** FileProcessor:
63. """Processing the data to and from text file"""
65. @staticmethod
66. **def** read\_file(file\_name, table):
67. """Function to manage data ingestion from file to a list of dictionaries
69. Reads the data from file identified by file\_name into a 2D table
70. (list of dicts) table one line in the file represents one dictionary row in table.
72. Args:
73. file\_name (string): name of file used to read the data from
74. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
76. Returns:
77. None.
78. """
79. table.clear()  # this clears existing data and allows to load data from file
80. objFile = open(file\_name, 'r')
81. **for** line **in** objFile:
82. data = line.strip().split(',')
83. dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
84. table.append(dicRow)
85. objFile.close()
87. @staticmethod
88. **def** write\_file(file\_name, table):
89. """
90. Function to write data from table into a .txt file
92. Args:
93. file\_name (string): name of file used to write the data to
94. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
96. Returns:
97. None.
98. """
99. objFile = open(file\_name, 'w')
100. **for** row **in** table:
101. lstValues = list(row.values())
102. lstValues[0] = str(lstValues[0])
103. objFile.write(','.join(lstValues) + '\n')
104. objFile.close()
106. # -- PRESENTATION (Input/Output) -- #
108. **class** IO:
109. """Handling Input / Output"""
111. @staticmethod
112. **def** print\_menu():
113. """Displays a menu of choices to the user
115. Args:
116. None.
118. Returns:
119. None.
120. """
122. **print**('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
123. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')
125. @staticmethod
126. **def** menu\_choice():
127. """Gets user input for menu selection
129. Args:
130. None.
132. Returns:
133. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
135. """
136. choice = ' '
137. **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:
138. choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
139. **print**()  # Add extra space for layout
140. **return** choice
142. @staticmethod
143. **def** show\_inventory(table):
144. """Displays current inventory table

147. Args:
148. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
150. Returns:
151. None.
153. """
154. **print**('======= The Current Inventory: =======')
155. **print**('ID\tCD Title (by: Artist)\n')
156. **for** row **in** table:
157. **print**('{}\t{} (by:{})'.format(\*row.values()))
158. **print**('======================================')
160. @staticmethod
161. **def** input\_CDinfo():
162. """Gets CD information as input from user
164. Args:
165. none
167. Returns:
168. strID, StrTitle, strArtist
170. """
171. strID = input('Enter ID: ').strip()
172. strTitle = input('What is the CD\'s title? ').strip()
173. strArtist = input('What is the Artist\'s name? ').strip()
174. **return** strID, strTitle, strArtist
176. # 1. When program starts, read in the currently saved Inventory
177. FileProcessor.read\_file(strFileName, lstTbl)
179. # 2. start main loop
180. **while** True:
181. # 2.1 Display Menu to user and get choice
182. IO.print\_menu()
183. strChoice = IO.menu\_choice()
185. # 3. Process menu selection
186. # 3.1 process exit first
187. **if** strChoice == 'x':
188. **break**
189. # 3.2 process load inventory
190. **if** strChoice == 'l':
191. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
192. strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled')
193. **if** strYesNo.lower() == 'yes':
194. **print**('reloading...')
195. FileProcessor.read\_file(strFileName, lstTbl)
196. IO.show\_inventory(lstTbl)
197. **else**:
198. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
199. IO.show\_inventory(lstTbl)
200. **continue**  # start loop back at top.
201. # 3.3 process add a CD
202. **elif** strChoice == 'a':
203. # 3.3.1 Ask user for new ID, CD Title and Artist
204. strID, strTitle, strArtist = IO.input\_CDinfo()
205. # 3.3.2 Add item to the table
206. DataProcessor.add\_CD(strID, strTitle, strArtist, lstTbl)
207. IO.show\_inventory(lstTbl)
208. **continue**  # start loop back at top.
209. # 3.4 process display current inventory
210. **elif** strChoice == 'i':
211. IO.show\_inventory(lstTbl)
212. **continue**  # start loop back at top.
213. # 3.5 process delete a CD
214. **elif** strChoice == 'd':
215. # 3.5.1 get Userinput for which CD to delete
216. # 3.5.1.1 display Inventory to user
217. IO.show\_inventory(lstTbl)
218. # 3.5.1.2 ask user which ID to remove
219. intIDDel = int(input('Which ID would you like to delete? ').strip())
220. # 3.5.2 search thru table and delete CD
221. DataProcessor.del\_CD(intIDDel, lstTbl)
222. IO.show\_inventory(lstTbl)
223. **continue**  # start loop back at top.
224. # 3.6 process save inventory to file
225. **elif** strChoice == 's':
226. # 3.6.1 Display current inventory and ask user for confirmation to save
227. IO.show\_inventory(lstTbl)
228. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
229. # 3.6.2 Process choice
230. **if** strYesNo == 'y':
231. # 3.6.2.1 save data
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. # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
237. **else**:
238. **print**('General Error')