Greg Marable

2020-Mar-02

IT FDN 100 B Wi 20: Foundations of Programming: Python

Assignment06

Python Assignment06 – CDInventory.py

# Introduction

This is another four-part assignment. As usual the questions and basic answers will lead the way. Second, overviews of all three of the labs. Third, the assignment and the troubles that I faced with working on it. Finally, the summary to wrap up all of my thoughts on this week.

# Topic 1 - Questions

• What is a function?

Function is a way of grouping statements and making the group available via a programmer defined name. This is helpful because lumping frequently used statements together can save space and clean up a script.

• What are parameters?

Parameters are what is defined within the function.

• What are arguments?

An argument is what you will eventually fill the function with.

• What is the difference between parameters and arguments?

Parameters allow you to pass in values for processing arguments are what are passed into the function.

• What are return values?

A return value does exactly what it says, it returns what was fed into it. Unlike print() it shouldn’t just display it. It will place the value into memory to be used when it is needed.

• What is the difference between a global and a local variable?

A global variable is a way of communicating our desire change a variable on the parent level to inside a child. A local variable on the other hand that is only accessible inside of a function.

• What is shadowing?

Shadowing is when a variable in a certain scope has the same name as a variable declared in an outer scope. [Variable Shadowing](https://en.wikipedia.org/wiki/Variable_shadowing)

• How do you use functions to organize your code?

Functions create a logical block of code that does something for your code. It lets you group code together and not have similar code pasted all over your script.

• What is the difference between a function and a class?

A function is a way of grouping statements together while a class is a way of grouping functions together.

• How do functions help you program using the “Separations of Concerns" pattern?

Creating functions allows the programmer to separate blocks of code into logical sections. This makes it easier to see where different things are happening in the code and you can adjust when needed.

# Topic 2 - Lab Results

# LAB 06-A

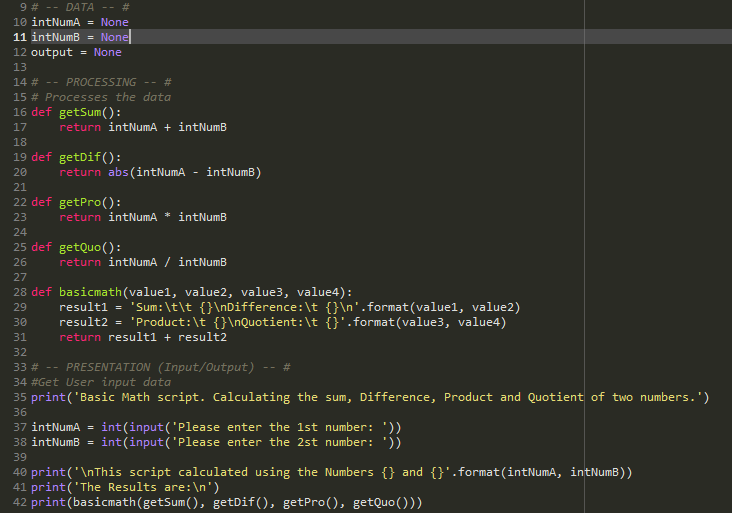


Figure - LAB06-A - Script

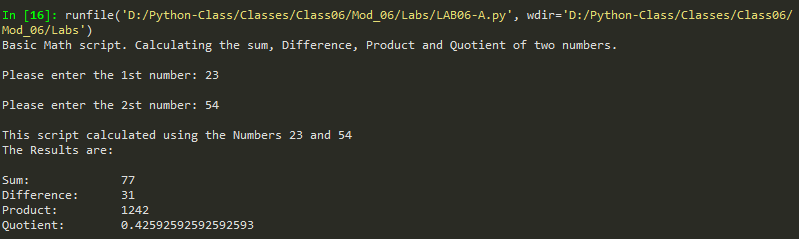


Figure - LAB06-A – Output

This lab works by turning all of the equations into functions. Sum, difference, Product, quotient are what we're using for our moth. We then create one final function for basic math. This function is asking to return the values of result1 and result2. Inside of both of our results we've listed two parameters that are going to be filled with our values. When the user inputs their numbers all of our math functions are calculated and returned and start filling in the parameters for our results. We then print our output and basicmath.

# LAB 06-B

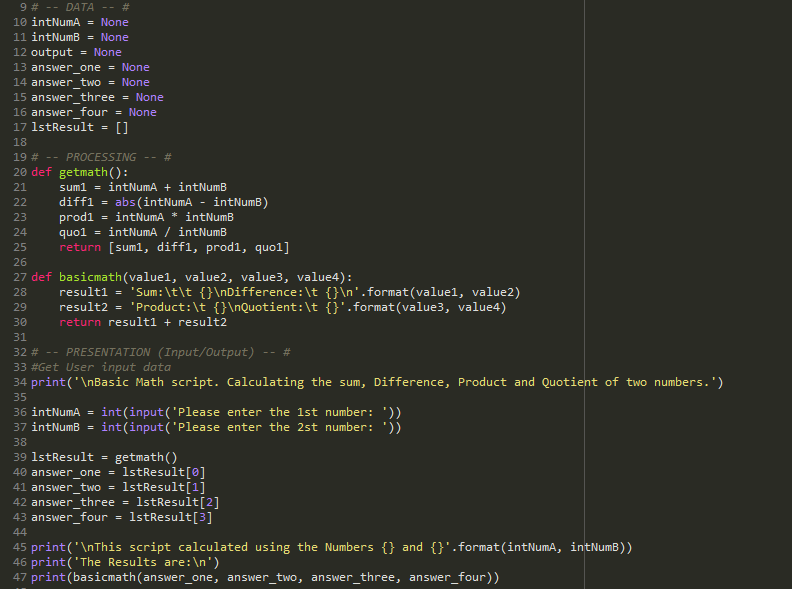


Figure - LAB06-B - Script

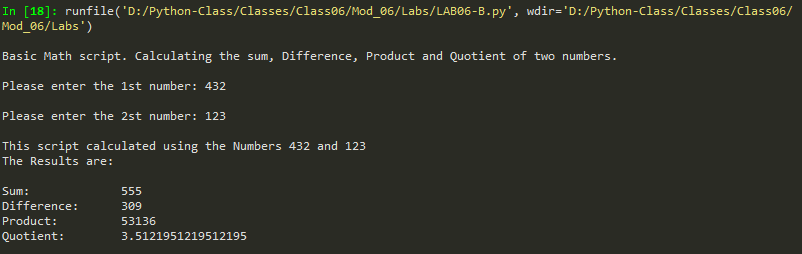


Figure - LAB06-B – Output

The way that I changed the code to consolidate functions was to add all of the calculations into one function and ask the function to return a list with those variables. I kept the basicmath function the exact same. I then defined lstResult as getmath() which was giving me a list of numbers associated with my calculations. From that point I assigned all of those numbers using the index from the list and assigned them an answer number. From there it was really easy to plug it into basicmath() and get the output that I was looking for.

# LAB 06-C

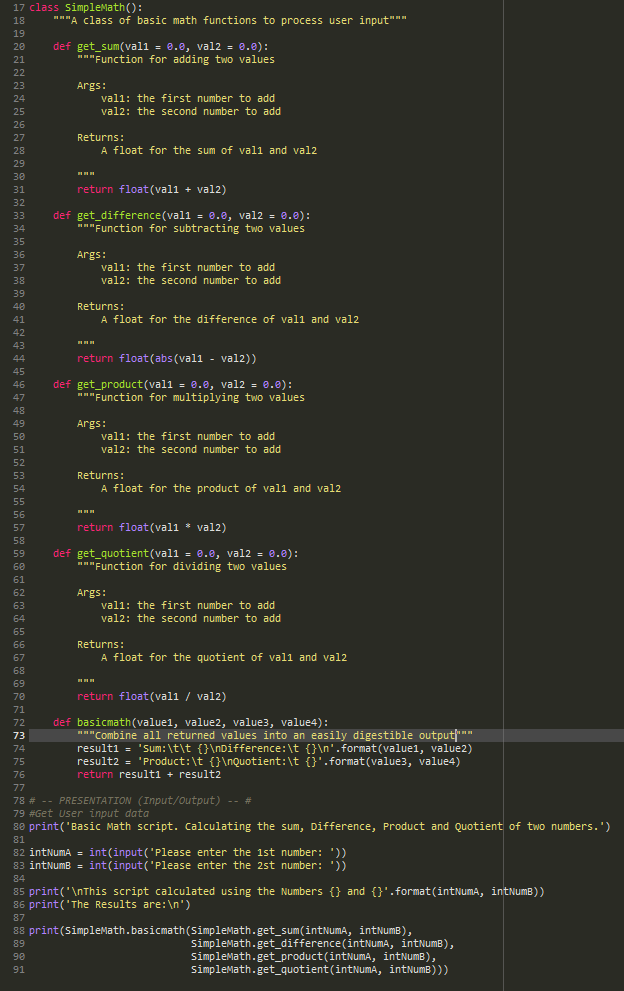


Figure - LAB06-C Script

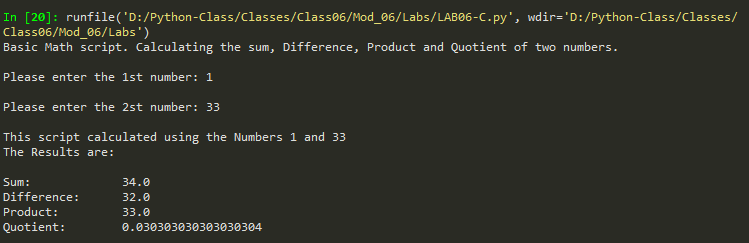


Figure - LAB06-C - Output

The code works essentially the same way that all the other labs work. This time all of the functions are assigned under a class. Each function is called with their class and function separated by a period. At this point its just a matter of calling out each function under print() and you have the result needed.

# Topic 3 – Assignment06 –



Figure - Assignment06 - FileProcessor Script



Figure - Assignment06 - IO Script

I really enjoyed learning about classes and functions in this assignment. I struggled on a few things but once I wrapped my head around them, I was able to make the ideas work pretty well. I want to talk about what I liked and what I found a little difficult.

First, what I liked; I didn’t make any menu changes this time around. After hearing what Dirk had to say I totally understood why we shouldn’t. It can confuse the user of the product. Honestly, if that user was me then there is no way that I would be happy trying to learn a whole new menu after using a product for so long. Another thing that I liked was the overall idea of functions and classes. I can totally see the power of them. They clean up a script. The clutter might not have totally disappeared but honestly there’s a heck of a lot less then there was. Organizing all of the functions into a class logically organizes and understand what each function might do. Amazing!

This section on what I found difficult will be short. Once again, my problems are coming down to thinking the same way as a computer. I’m still struggling with parameters. The idea of what a parameter is makes sense to me. I get it. The problem is implementing them and linking those parameters to functions is difficult. I think that it’s going to have to take a lot more time playing around with them and really trying to understand them to really make them work for me. Oh well, with enough playing around I eventually made them work but without Dirk listing out file\_name and table on the write\_file function I would have probably had to spend an insane amount of time trying to plug things in and fail.

# Summary

What a great lesson. We learned about functions much earlier in class but I’m glad we finally got a chance to really dive more deeply into them. They are great tools for cleaning up scripts and organizing them. I have a feeling that we will be using them a lot more in the upcoming assignments. I know that I have a lot more work on my end to fully understand how to work with functions but I’m looking forward to learning.

# Appendix

GitHub Link - <https://github.com/gamarable/Assignment06>

Variable Shadowing - <https://en.wikipedia.org/wiki/Variable_shadowing>

1. #------------------------------------------#
2. # Title: CDInventory.py
3. # Desc: Working with classes and functions.
4. # Change Log: (Who, When, What)
5. # GMarable, 2020-Feb-29, Created File
6. # GMarable, 202-Mar-02, Finished Assignment
7. #------------------------------------------#
9. # -- DATA -- #
10. strChoice = '' # User input
11. lstTbl = []  # list of lists to hold data
12. dicRow = {}  # list of data row
13. strFileName = 'CDInventory.txt'  # data storage file
14. objFile = None  # file object
15. strID = None
16. strTitle = None
17. stArtist = None

20. # -- PROCESSING -- #
21. **class** DataProcessor:
22. """Processes data in memory"""
24. **def** load\_file():
25. """Loads information currently on CDInventory.txt.
27. Will wipe all information from memory and pull whatever is on CDInventory.txt currently into memory.
29. Args:
30. None
32. Returns:
33. Formatted version of CDInventory.txt information
35. """
36. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
37. strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled: ')
38. **if** strYesNo.lower() == 'yes':
39. **print**('reloading...')
40. FileProcessor.read\_file(strFileName, lstTbl)
41. IO.show\_inventory(lstTbl)
42. **else**:
43. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
44. IO.show\_inventory(lstTbl)
46. **def** del\_inventory():
47. """Searches through table and removes user input ID number
49. Only deleted from memory. Does not write to file until saved with FileProcessor.write\_file.
51. Args:
52. None
54. Returns:
55. 'The CD was removed' or 'Could not find this CD!'
57. """
58. intRowNr = -1
59. blnCDRemoved = False
60. **for** row **in** lstTbl:
61. intRowNr += 1
62. **if** row['ID'] == intIDDel:
63. **del** lstTbl[intRowNr]
64. blnCDRemoved = True
65. **break**
66. **if** blnCDRemoved:
67. **print**('The CD was removed')
68. **else**:
69. **print**('Could not find this CD!')

72. **class** FileProcessor:
73. """Processing the data to and from text file"""
75. @staticmethod
76. **def** read\_file(file\_name, table):
77. """Function to manage data ingestion from file to a list of dictionaries
79. Reads the data from file identified by file\_name into a 2D table
80. (list of dicts) table one line in the file represents one dictionary row in table.
82. Args:
83. file\_name (string): name of file used to read the data from
84. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
86. Returns:
87. None.
88. """
89. table.clear()  # this clears existing data and allows to load data from file
90. with open(strFileName, 'r') as objFile:
91. **for** line **in** objFile:
92. data = line.strip().split(',')
93. dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
94. table.append(dicRow)
96. @staticmethod
97. **def** write\_file(file\_name, table):
98. """Pulls user input from IO.input\_add\_inventory and formats information to a dictionary.
100. Args:
101. file\_name (string): name of file used to read the data from
102. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
104. Returns:
105. IO.show\_inventory(lstTbl) displays inventory currently in table
107. """
108. lstResult = IO.input\_add\_inventory()
109. answer\_one = lstResult[0]
110. answer\_two = lstResult[1]
111. answer\_three = lstResult[2]
112. dicRow = {'ID': answer\_one, 'Title': answer\_two, 'Artist': answer\_three}
113. lstTbl.append(dicRow)
114. IO.show\_inventory(lstTbl)

117. **def** save\_file():
118. """Saves changes made to lstValues
120. Opens CDInventory.txt and specifies write. Formats data with commas
121. seperating individual values and at the end of the line adds a new
122. row.
124. Args:
125. None
127. Returns:
128. None
130. """
131. objFile = open(strFileName, 'w')
132. **for** row **in** lstTbl:
133. lstValues = list(row.values())
134. lstValues[0] = str(lstValues[0])
135. objFile.write(','.join(lstValues) + '\n')
136. objFile.close()
138. # -- PRESENTATION (Input/Output) -- #
140. **class** IO:
141. """Handling Input / Output"""
143. @staticmethod
144. **def** print\_menu():
145. """Displays a menu of choices to the user
147. Args:
148. None.
150. Returns:
151. None.
152. """
154. **print**('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
155. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')
157. @staticmethod
158. **def** menu\_choice():
159. """Gets user input for menu selection
161. Args:
162. None.
164. Returns:
165. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
167. """
168. choice = ' '
169. **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:
170. choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
171. **print**()  # Add extra space for layout
172. **return** choice
174. @staticmethod
175. **def** show\_inventory(table):
176. """Displays current inventory table
178. Args:
179. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
181. Returns:
182. None.
184. """
185. **print**('======= The Current Inventory: =======')
186. **print**('ID\tCD Title (by: Artist)\n')
187. **for** row **in** table:
188. **print**('{}\t{} (by:{})'.format(\*row.values()))
189. **print**('======================================')
191. **def** input\_add\_inventory():
192. """User input assigned to list
194. List will be fed to FileProcessor.write\_file to process data into a dictionary
196. Args:
197. None
199. Returns:
200. List of user inputs
202. """
203. strID = input('Enter ID: ').strip()
204. strTitle = input('What is the CD\'s title? ').strip()
205. stArtist = input('What is the Artist\'s name? ').strip()
206. **return** [strID, strTitle, stArtist]

209. # 1. When program starts, read in the currently saved Inventory
210. FileProcessor.read\_file(strFileName, lstTbl)
212. # 2. start main loop
213. **while** True:
214. # 2.1 Display Menu to user and get choice
215. IO.print\_menu()
216. strChoice = IO.menu\_choice()
217. # 3. Process menu selection
218. # 3.1 process exit first
219. **if** strChoice == 'x':
220. **break**
221. # 3.2 process load inventory
222. **if** strChoice == 'l':
223. **print**(DataProcessor.load\_file())
224. # 3.3 process add a CD
225. **elif** strChoice == 'a':
226. FileProcessor.write\_file(strFileName, lstTbl)
227. # 3.3.1 Ask user for new ID, CD Title and Artist
228. # 3.3.2 Add item to the table
229. # 3.4 process display current inventory
230. **elif** strChoice == 'i':
231. IO.show\_inventory(lstTbl)
232. **continue**  # start loop back at top.
233. # 3.5 process delete a CD
234. **elif** strChoice == 'd':
235. # 3.5.1 get Userinput for which CD to delete
236. # 3.5.1.1 display Inventory to user
237. IO.show\_inventory(lstTbl)
238. # 3.5.1.2 ask user which ID to remove
239. intIDDel = int(input('Which ID would you like to delete? ').strip())
240. # 3.5.2 search thru table and delete CD
241. DataProcessor.del\_inventory()
242. IO.show\_inventory(lstTbl)
243. **continue**  # start loop back at top.
244. # 3.6 process save inventory to file
245. **elif** strChoice == 's':
246. # 3.6.1 Display current inventory and ask user for confirmation to save
247. IO.show\_inventory(lstTbl)
248. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
249. # 3.6.2 Process choice
250. **if** strYesNo == 'y':
251. # 3.6.2.1 save data
252. FileProcessor.save\_file()
253. **else**:
254. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
255. **continue**  # start loop back at top.
256. # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
257. **else**:
258. **print**('General Error')