

How to create a script using Functions and Classes

Introduction

For the 6th Module, we learned about functions and classes which were applied to modify CDInventory.py script. The purpose of this document is to explore some of those learnings and demonstrate the use of functions and classes in a script.

Functions

In the previous module, we learned about Separation of concerns which is a design principle for separating a computer program into distinct sections such that each section addresses a separate concern.¹ The main takeaway with the application of separation of concerns is to intentionally break up the code in tiny sections instead of writing the code in one solid block. When the problem is broken up it is easier for the programmer to address the individual tasks within the script. One way of breaking up the code into sections is through the use of functions. A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.² In the declaration of function, an action is defined but not executed until the defined name is called upon. Within a function the parameters may be defined which identifies the values to be processed within the function. A return value is also defined which is the result when the function is executed. For the assignment, a number of codes in the main loop were moved in to the classes section by utilizing functions.

Classes

From the class videos, we learned that classes are a way of grouping functions, variables and constraints. A class is a user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.³ For the class assignment, 3 classes were identified: Data Processor to process data within the script, File Process to process data to and from the text file, and I/O for handling input and out.

Modifying the CDInventory.py script.

For this assignment, a CDInventory_Starter.py script was provided as a starting to implement changes that included the moving code from the main loop into the classes while using functions.

¹ https://en.wikipedia.org/wiki/Separation_of_concerns, retrieved 2021-Aug-08

² https://www.tutorialspoint.com/python/python_functions.htm, retrieved 2021-Aug-15

³ https://www.tutorialspoint.com/python/python_classes_objects.htm, retrieved 2021-Aug-15

Task 3.3.1 asks the user for information on new ID, CD Title and Artist. To move this to the I/O class, I first declared the function to add the cd information as `cd_data()`. I also added comments to describe that the function of the `cd_data()` to collect user information and to return the multiple values including `strID`, `strTitle`, and `strArtist`. Figure 1 below is the code for declaring the `cd_data()` function.

```
178     @staticmethod
179     def cd_data():
180
181         """Function to collect CD Data from the user: CD ID, Album, Artist
182
183         Args:
184             None.
185
186         Returns:
187             strID (string): this is the CD ID entered by the user
188             strTitle (string): this is the CD's title
189             strArtist (string): this is the Artist of the CD
190
191         """
192         strID = input('Enter ID: ').strip()
193         strTitle = input('What is the CD\'s title? ').strip()
194         strArtist = input('What is the Artist\'s name? ').strip()
195         return strID, strTitle, strArtist
```

Figure 1 – `cd_data` function

Next the I updated the task in 3.3.1 to call the function for `cd_data()`. Figure 2 below displays the code for calling the `cd_data()` function to collect user data if the user chooses to add a CD.

```
225     # 3.3.1 Ask user for new ID, CD Title and Artist
226     # TODOOne move IO code into function
227     strID, strTitle, strArtist = IO.cd_data()
228
```

Figure 2 – calling the `cd_data()` function to collect CD information.

Task 3.3.2 requires moving the code for adding the user input into the `IstTbl` from the main loop to data processing section. To move the code to the `DataProcessor` class, I first declared the function to add the cd info as `cd_addition` referencing the parameters - `strID`, `strTitle`, `strArtist` from the `cd_data` function, and an additional table that holds the dictionary list data during this execution of the `cd_data` function. Figure 3 below is the code for declaring the `cd_addition (strID, strTitle, strArtist, table)` function.

```

21     @staticmethod
22     def cd_addition(strID, strTitle, strArtist, table):
23
24         """Function to manage user input and add to the dictionary list
25
26         Processes user data and formats it into a 2D table
27         (list of dicts) table one line in the file represents one dictionary row in table.
28         User data of CD information is collected and added as a row in the dictionary list.
29         The table is appended to include the additional row in the dictionary list.
30
31         Args:
32             ID (string): this is the CD ID entered by the user
33             Title (string): this is the CD's title
34             Artist (string): this is the Artist of the CD
35             table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
36
37         Returns:
38             None.
39         """
40         intID = int(strID)
41         dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}
42         table.append(dicRow)
43

```

Figure 3 – cd_additon function

Next the I updated the task in 3.3.2 to call the function for cd_addition. Figure 4 below displays the code for calling the cd_data function to append the collected cd data into the lstTbl when the user chooses to add a CD.

```

229         # 3.3.2 Add item to the table
230         # TODOOne move processing code into function
231         DataProcessor.cd_addition(strID, strTitle, strArtist, lstTbl)
232         IO.show_inventory(lstTbl)
233         continue # start loop back at top.

```

Figure 4 – calling the cd_addition function to append the lstTbl.

Task 3.5.2 requires moving the code for searching thru the lstTbl and deleting a CD from the main loop to data processing section. To move the code to the DataProcessor class, I first declared the function to delete the cd as cd_deletion referencing the parameter 'table' - which is a temporary table holding the data provided by the user to identify the cd to be deleted. If the ID matches, the entire row from the lstTbl will be removed. Figure 5 below is the code for declaring the cd_deletion function.

```

45     @staticmethod
46     def cd_deletion(table):
47
48         """Function to manage user input and delete items from the dictionary list
49
50         Processes user data that identifies the CD to be deleted and deletes from the 2D table
51         (list of dicts). The user identifies the CD ID to be deleted, and all data associated with the ID
52         is deleted.
53
54         Args:
55             table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
56
57         Returns:
58             None.
59         """
60         intRowNr = -1
61         blnCDRemoved = False
62         for row in lstTbl:
63             intRowNr += 1
64             if row['ID'] == intIDDel:
65                 del lstTbl[intRowNr]
66                 blnCDRemoved = True
67                 break
68         if blnCDRemoved:
69             print('The CD was removed')
70         else:
71             print('Could not find this CD!')

```

Figure 5 – cd_deletion function

Next the I updated the task in 3.5.2 to call the function for cd_deletion. Figure 6 below displays the code for calling the cd_deletion function to the cd information from the lstTbl when the user chooses to delete a CD and have provided an ID that is currently available from the inventory.

```

245         # 3.5.2 search thru table and delete CD
246         # TODOOne move processing code into function
247         DataProcessor.cd_deletion(lstTbl)
248         IO.show_inventory(lstTbl)
249         continue # start loop back at top.
250         # 3.6 process user inventory to file

```

Figure 6 – calling the cd_deletion function to delete from the lstTbl

The final task is to be modified is Task 3.6.2.1 requires moving the code for saving current inventory list from the main loop to file processing section. To move the code to the FileProcessor class, I first declared the function write_file referencing the parameter for the file_name, and 'table' - which is a temporary table holding CD inventory list. Figure 7 below is the code for declaring the write_file function.

```

100     @staticmethod
101     def write_file(file_name, table):
102         # TODOOne Add code here
103         """Function to save current CD Inventory list into text file
104         Saves dictionary list (list of dicts) table one line in the file
105         represents one dictionary row in table. The collection of dictionary rows is the CD Inventory list.
106
107         Args:
108             file_name (string): name of file to write data to
109             table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
110
111         Returns:
112             None.
113         """
114         objFile = open(strFileName, 'w')
115         for row in table:
116             lstValues = list(row.values())
117             lstValues[0] = str(lstValues[0])
118             objFile.write(','.join(lstValues) + '\n')
119         objFile.close()

```

Figure 7 – write_file function

Next the I updated the task in 3.6.2.1 o call the function for write_file. Figure 8 below displays the code for calling the write_file function to save the lstTbl to the text file when the user chooses to save the data.

```

257         # 3.6.2.1 save data
258         # TODOOne move processing code into function
259         FileProcessor.write_file(strFileName, lstTbl)

```

Figure 8 – calling the write_file function to save data to the text file

Challenges

One of the main challenges of this assignment is getting through the code and figuring the ask for modifying the script. It was quite overwhelming trying to figure out where to start especially since changes to the main loop need to be reflected in the class section as functions. Following that, the code in the main loop needs to be updated to tie back to the function declared in the classes. From this assignment, I learned to slow down and stepped back to look at the big picture while looking for a pattern in the way I tackled the modification. It was also very helpful to carefully read the To Do task as it provided clues as to where in the class section I needed to make an update. When calling for the function in the main loop, I also learned that I needed connect the function name to the class ie: Fileprocessor.write_file in order to reference where the function is located. Searching for all the To Do tasks using the Ctrl + F to search for To Do was also very helpful to ensure that all the tasks have been completed while marking those as to TODOOne so as not to loose the original ask while marking the task complete.

Running the Python Script

After saving the file, I ran the CDInventory script in both Spyder and the Terminal. Figures 9 and 10 below display the script working on the computer ensuring that all options are running correctly.

```

In [23]: runfile('C:/_FDProgramming/Mod_06/Assignment06/Assignment06.py', wdir='C:/_FDProgramming/
Mod_06/Assignment06')
Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: l

WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.

type 'yes' to continue and reload from file. otherwise reload will be canceledyes
reloading...
===== The Current Inventory: =====
ID  CD Title (by: Artist)

1   The Big Wheel (by:Runrig)
2   Bad (by:Michael Jackson)
=====
Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: a

Enter ID: 3

What is the CD's title? Abbey Road

What is the Artist's name? The Beattles
===== The Current Inventory: =====
ID  CD Title (by: Artist)

1   The Big Wheel (by:Runrig)
2   Bad (by:Michael Jackson)
3   Abbey Road (by:The Beattles)
=====

```

Figure 9 – Running CDInventory.py in Spyder

```

=====
Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: i

===== The Current Inventory: =====
ID  CD Title (by: Artist)

1   The Big Wheel (by:Runrig)
2   Bad (by:Michael Jackson)
3   Abbey Road (by:The Beattles)
=====

Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: d

===== The Current Inventory: =====
ID  CD Title (by: Artist)

1   The Big Wheel (by:Runrig)
2   Bad (by:Michael Jackson)
3   Abbey Road (by:The Beattles)
=====

Which ID would you like to delete? 3
The CD was removed
===== The Current Inventory: =====
ID  CD Title (by: Artist)

1   The Big Wheel (by:Runrig)
2   Bad (by:Michael Jackson)
=====

```

Figure 9a – Running CDInventory.py in Spyder – continued

```
=====
Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: s

===== The Current Inventory: =====
ID  CD Title (by: Artist)

1   The Big Wheel (by:Runrig)
2   Bad (by:Michael Jackson)
=====

Save this inventory to file? [y/n] y
Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: x

In [24]: |
```

Figure 9b – Running CDInventory.py in Spyder – continued


```

Anaconda Prompt (anaconda3)

(base) C:\> cd C:\_FDProgramming\Mod_06\Assignment06

(base) C:\_FDProgramming\Mod_06\Assignment06> python CDInventory.py
Menu

[1] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [1, a, i, d, s or x]: 1

WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.
type 'yes' to continue and reload from file. otherwise reload will be canceledyes
reloading...
===== The Current Inventory: =====
ID      CD Title (by: Artist)
1       The Big Wheel (by:Runrig)
2       Bad (by:Michael Jackson)
=====
Menu

[1] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [1, a, i, d, s or x]: a

Enter ID: 3
What is the CD's title? Abbey Road
What is the Artist's name? The Beattles
===== The Current Inventory: =====
ID      CD Title (by: Artist)
1       The Big Wheel (by:Runrig)
2       Bad (by:Michael Jackson)
3       Abbey Road (by:The Beattles)
=====
Menu

[1] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

```

Figure 10 – Running CDInventory.py in Anaconda

Which operation would you like to perform? [l, a, i, d, s or x]: i

===== The Current Inventory: =====

ID CD Title (by: Artist)

1 The Big Wheel (by:Runrig)
 2 Bad (by:Michael Jackson)
 3 Abbey Road (by:The Beattles)

=====

Menu

[l] load Inventory from file
 [a] Add CD
 [i] Display Current Inventory
 [d] delete CD from Inventory
 [s] Save Inventory to file
 [x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: d

===== The Current Inventory: =====

ID CD Title (by: Artist)

1 The Big Wheel (by:Runrig)
 2 Bad (by:Michael Jackson)
 3 Abbey Road (by:The Beattles)

=====

Which ID would you like to delete? 3

The CD was removed

===== The Current Inventory: =====

ID CD Title (by: Artist)

1 The Big Wheel (by:Runrig)
 2 Bad (by:Michael Jackson)

=====

Menu

[l] load Inventory from file
 [a] Add CD
 [i] Display Current Inventory
 [d] delete CD from Inventory
 [s] Save Inventory to file
 [x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: s

===== The Current Inventory: =====

ID CD Title (by: Artist)

1 The Big Wheel (by:Runrig)
 2 Bad (by:Michael Jackson)

=====

Save this inventory to file? [y/n] y

^

Figure 10a – Running CDInventory.py in Anaconda – continued

```
Anaconda Prompt (anaconda3)

=====
Save this inventory to file? [y/n] y
Menu

[1] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [1, a, i, d, s or x]: x

(base) C:\_FDProgramming\Mod_06\Assignment06>
```

Figure 10b – Running CDInventory.py in Anaconda – continued

Figure 11 below displays the .txt file where the user data is stored.

```
*CDInventory.txt - Notepad
File Edit Format View Help
1,The Big Wheel,Runrig
2,Bad,Michael Jackson
```

Figure 11 – Stored data in CDInventory.txt

GitHub

The assignment files including CDInventory.py, CDInventory.txt, and the Know_Doc_05.pdf have been upload to GitHub - https://github.com/jlmagat/Assignment_06 .

Summary

Module 6 continues to stretch my knowledge and my application of various python programming concepts including the use of functions and classes to break the code into more digestible sections. Specific to this assignment, I found a pattern that was most helpful to me which was declaring a function in the classes, and updating the code in the main loop to reference the class and call the function in order to execute code. Overall, I am excited as I continue to develop my programming skills.

Appendix

Listing CDInventory.py – part 1

```
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  # JMagat, 2021-Aug-15, Modified file to use functions
7  # moving code from the main loop to the class as functions
8  # cd_addition, cd_deletion, write_file and cd_data
9  #-----#
10
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
17
18
19 # -- PROCESSING -- #
20 class DataProcessor:
21     # TODOOne add functions for processing here
22
23     @staticmethod
24     def cd_addition(strID, strTitle, strArtist, table):
25
26         """Function to manage user input and add to the dictionary list
27
28         Processes user data and formats it into a 2D table
29         (list of dicts) table one line in the file represents one dictionary row in table.
30         User data of CD information is collected and added as a row in the dictionary list.
31         The table is appended to include the additional row in the dictionary list.
32
33         Args:
34             ID (string): this is the CD ID entered by the user
35             Title (string): this is the CD's title
36             Artist (string): this is the Artist of the CD
37             table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
38
39         Returns:
40             None.
41         """
42         intID = int(strID)
43         dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}
44         table.append(dicRow)
45
46
47     @staticmethod
48     def cd_deletion(table):
49
50         """Function to manage user input and delete items from the dictionary list
51
52         Processes user data that identifies the CD to be deleted and deletes from the 2D table
```

Listing CDInventory.py – part 2

```
52 Processes user data that identifies the CD to be deleted and deletes from the 2D table
53 (list of dicts). The user identifies the CD ID to be deleted, and all data associated with the ID
54 is deleted.
55
56 Args:
57     table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
58
59 Returns:
60     None.
61 """
62 intRowNr = -1
63 blnCDRemoved = False
64 for row in lstTbl:
65     intRowNr += 1
66     if row['ID'] == intIDDel:
67         del lstTbl[intRowNr]
68         blnCDRemoved = True
69         break
70 if blnCDRemoved:
71     print('The CD was removed')
72 else:
73     print('Could not find this CD!')
74
75
76
77 class FileProcessor:
78     """Processing the data to and from text file"""
79
80     @staticmethod
81     def read_file(file_name, table):
82         """Function to manage data ingestion from file to a list of dictionaries
83
84         Reads the data from file identified by file_name into a 2D table
85         (list of dicts) table one line in the file represents one dictionary row in table.
86
87         Args:
88             file_name (string): name of file used to read the data from
89             table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
90
91         Returns:
92             None.
93         """
94         table.clear() # this clears existing data and allows to load data from file
95         objFile = open(file_name, 'r')
96         for line in objFile:
97             data = line.strip().split(',')
98             dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
99             table.append(dicRow)
100         objFile.close()
101
102     @staticmethod
103     def write_file(file_name, table):
```

Listing CDInventory.py – part 3

```
103 def write_file(file_name, table):
104     # TODOOne Add code here
105     """Function to save current CD Inventory list into text file
106     Saves dictionary list (list of dicts) table one line in the file
107     represents one dictionary row in table. The collection of dictionary rows is the CD Inventory list.
108
109     Args:
110         file_name (string): name of file to write data to
111         table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
112
113     Returns:
114         None.
115     """
116     objFile = open(strFileName, 'w')
117     for row in table:
118         lstValues = list(row.values())
119         lstValues[0] = str(lstValues[0])
120         objFile.write(','.join(lstValues) + '\n')
121     objFile.close()
122
123
124 # -- PRESENTATION (Input/Output) -- #
125
126 class IO:
127     """Handling Input / Output"""
128
129     @staticmethod
130     def print_menu():
131         """Displays a menu of choices to the user
132
133         Args:
134             None.
135
136         Returns:
137             None.
138         """
139
140         print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
141         print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')
142
143     @staticmethod
144     def menu_choice():
145         """Gets user input for menu selection
146
147         Args:
148             None.
149
150         Returns:
151             choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
152
153         """
154         choice = ' '
```

Listing CDInventory.py – part 4

```
154     choice = ''
155     while choice not in ['l', 'a', 'i', 'd', 's', 'x']:
156         choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
157     print() # Add extra space for layout
158     return choice
159
160     @staticmethod
161     def show_inventory(table):
162         """Displays current inventory table
163
164
165         Args:
166             table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
167
168         Returns:
169             None.
170
171         """
172         print('===== The Current Inventory: =====')
173         print('ID\tCD Title (by: Artist)\n')
174         for row in table:
175             print('{}\t{} (by: {})'.format(*row.values()))
176         print('=====')
177
178     # TODOOne add I/O functions as needed
179
180     @staticmethod
181     def cd_data():
182
183         """Function to collect CD Data from the user: CD ID, Album, Artist
184
185
186         Args:
187             None.
188
189         Returns:
190             strID (string): this is the CD ID entered by the user
191             strTitle (string): this is the CD's title
192             strArtist (string): this is the Artist of the CD
193
194         """
195         strID = input('Enter ID: ').strip()
196         strTitle = input('What is the CD\'s title? ').strip()
197         strArtist = input('What is the Artist\'s name? ').strip()
198         return strID, strTitle, strArtist
199
200     # 1. When program starts, read in the currently saved Inventory
201     FileProcessor.read_file(strFileName, lstTbl)
202
203     # 2. start main loop
204     while True:
205         # 2.1 Display Menu to user and get choice
```

Listing CDInventory.py – part 5

```
205 # 2.1 Display Menu to user and get choice
206 IO.print_menu()
207 strChoice = IO.menu_choice()
208
209 # 3. Process menu selection
210 # 3.1 process exit first
211 if strChoice == 'x':
212     break
213 # 3.2 process load inventory
214 if strChoice == 'l':
215     print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
216     strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled')
217     if strYesNo.lower() == 'yes':
218         print('reloading...')
219         FileProcessor.read_file(strFileName, lstTbl)
220         IO.show_inventory(lstTbl)
221     else:
222         input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
223         IO.show_inventory(lstTbl)
224     continue # start loop back at top.
225 # 3.3 process add a CD
226 elif strChoice == 'a':
227     # 3.3.1 Ask user for new ID, CD Title and Artist
228     # TODOOne move IO code into function
229     strID, strTitle, strArtist = IO.cd_data()
230
231     # 3.3.2 Add item to the table
232     # TODOOne move processing code into function
233     DataProcessor.cd_addition(strID, strTitle, strArtist, lstTbl)
234     IO.show_inventory(lstTbl)
235     continue # start loop back at top.
236 # 3.4 process display current inventory
237 elif strChoice == 'i':
238     IO.show_inventory(lstTbl)
239     continue # start loop back at top.
240 # 3.5 process delete a CD
241 elif strChoice == 'd':
242     # 3.5.1 get Userinput for which CD to delete
243     # 3.5.1.1 display Inventory to user
244     IO.show_inventory(lstTbl)
245     # 3.5.1.2 ask user which ID to remove
246     intIDDel = int(input('Which ID would you like to delete? ').strip())
247     # 3.5.2 search thru table and delete CD
248     # TODOOne move processing code into function
249     DataProcessor.cd_deletion(lstTbl)
250     IO.show_inventory(lstTbl)
251     continue # start loop back at top.
252 # 3.6 process save inventory to file
253 elif strChoice == 's':
254     # 3.6.1 Display current inventory and ask user for confirmation to save
255     IO.show_inventory(lstTbl)
256     strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
```

Listing CDInventory.py – part 6

```
256 strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
257 # 3.6.2 Process choice
258 if strYesNo == 'y':
259     # 3.6.2.1 save data
260     # TODOOne move processing code into function
261     FileProcessor.write_file(strFileName, lstTbl)
262 else:
263     input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
264     continue # start loop back at top.
265 # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
266 else:
267     print('General Error')
```