

Charles Hodges(hodges11@uw.edu)  
August 15, 2021  
IT FDN 110: Introduction to Programming (Python)  
Assignment 06

# Modify an Example Solution of Assignment 05 by Adding Functions

## Introduction

In this assignment, we were given a sample solution to assignment #05. We were asked to finish that assignment by adding functions into the already defined classes. Most of this assignment was simplifying the sample solution, adding a few functions, and a bit of formatting. The aspect that took the longest was modifying the variable names to match the introduced [Google PyGuide's Naming Guidelines](#)(external link)<sup>1</sup>.

---

<sup>1</sup> Retrieved 2021-August-14

## Non-String Variables & Strings

```
9. # Variables
10. dic_row = {} # list of data row
11. lst_input_options = ['l', 'a', 'i', 'd', 's', 'x']
12. lst_tbl = [] # list of lists to hold data
13. obj_file = None # file object
14.
15. # Strings
16. str_cancelling_reload = (
17.     'Canceling... Inventory data NOT reloaded. '
18.     'Press [ENTER] to continue to the menu. \n'
19. )
20. str_cd_removed = 'The CD was removed.'
21. str_choice = '' # User input
22. str_confirm_reload = (
23.     'Type \'yes\' to continue and reload from the file. '
24.     'Otherwise, the reload will be canceled. --> '
25. )
26. str_file_name = 'CDInventory.txt' # The data storage file
27. str_footer = '======'
28. str_general_error = '!General Error!'
29. str_header = '\n===== The Current Inventory: ====='
30. str_inventory_not_saved = (
31.     'The inventory was NOT saved to file.'
32.     'Press [ENTER] to return to the menu.'
33. )
```

```

34. str_menu = (
35.     '\n'
36.     'MENU\n\n'
37.     '[l] load Inventory from file\n'
38.     '[a] Add CD\n'
39.     '[i] Display Current Inventory\n'
40.     '[d] Delete CD from Inventory\n'
41.     '[s] Save Inventory to file\n'
42.     '[x] Exit\n'
43. )
44. str_not_find_cd = 'Could not find this CD!'
45. str_reloading = 'reloading...'
46. str_save_inventory = 'Save this inventory to file? [y/n] '
47. str_sub_header = 'ID\tCD Title \t(by: Artist)\n'
48. str_what_artist = 'What is the Artist\'s name? '
49. str_what_id = 'Enter ID: '
50. str_what_title = 'What is the CD\'s title? '
51. str_which_delete = 'Which CD would you like to delete? Please use ID: '
52. str_which_operation = (
53.     'Which operation would you like to perform?'
54.     '[l, a, i, d, s or x]: '
55. )
56. str_warning = (
57.     'WARNING: If you continue, all unsaved data will be lost and the '
58.     'Inventory will be re-loaded from the file.'
59. )

```

Figure 1 - Non-String Variables & Strings

Once again we have two alphabetized lists, non-string variables and strings. When I reviewed the sample solution which we were given, I noticed that the variable names did not match the Google PyGuide's naming convention, so I edited everything, as I abstracted the strings to a list at the top. I did take Laura's advice, and did not migrate all the tiny strings to that list, as readability was impacted by leaving some very tiny strings within the code.

I am not entirely sure if we were expected to modify all of the non-string variables, and strings, to this convention, or not. But after we used the Google PyGuide for style examples in class, I assumed that we were going to use it further. Maybe we could get some guidance here. Finally, I was able to continue to practice line-wrapping successfully, so that was a bonus.

## Args & Returns

There are 10 functions total, across the three classes. Of those 10 functions, only 2 return any values, and only 1 takes any arguments. Not sure if that was what was intended, but since the sample solution used `global_var_names`, and the current state of the Inventory whenever a menu selection was made, there were hardly any arguments or returns needed. I found that a bit strange, given the topic of the week. Maybe I am misunderstanding something.

```
319.     # Add a CD.
320.     elif str_choice == 'a':
321.         # Ask user for new ID, CD Title and Artist,
322.         int_id_input, str_title_input, str_artist_input = IO.input_cd_info()
323.         # Add CD information to the Inventory
324.         DataProcessor.add_cd(int_id_input, str_title_input, str_artist_input)
325.         continue # start loop back at top.
```

Figure 2 - Unpacking the tuple

The thing I did learn about returns, was that when returning multiple values, unpacking of the tuple is required. Initially, I had these two lines written like this:

```
DataProcessor.add_cd(IO.input_cd_info())
```

The issue I ran into was that the three returned values from the function “`IO.input_cd_info`” were arranged in a tuple, which was returned as the first value only. So I needed to unpack that tuple, immediately, into some local variables, and then pass them into the “`DataProcessor.add_cd()`” function. I was disappointed that I couldn’t do that in one simple line. There were some other complex solutions that were described on StackOverflow, but, in the end, two simple lines were preferred to unnecessary complexity.

## Starting the Program

```
296. # When the program starts, read in the currently saved Inventory, if it exists.
297. # Otherwise, create the inventory file.
298. try:
299.     FileProcessor.read_file()
300. except FileNotFoundError:
301.     FileProcessor.create_file()
```

Figure 3 - Starting the Program

To start the program, I used a Try:Except Block, which was, admittedly, a first for me. Since the sample solution had some text describing how they expected the program to start, we would

need a couple of lines to accommodate that description. In this case, we try to read the current text file containing the Inventory. If the text file does not yet exist, we receive a `FileNotFoundError`. So we handle that error, keep the program running, and just create the file. While it does look a bit odd, it honors the sample solutions intended start.

## The Main Loop

```
304. # Start main loop
305. while True:
306.     # Display Menu to user, and get choice
307.     IO.print_menu()
308.     str_choice = IO.menu_choice()
309.
310.     # Exit
311.     if str_choice == 'x':
312.         break
```

Figure 4 - Top of the Main Loop

Here is where we start the While loop, which allows the User to continue to interact with the script, until they are ready to exit. We print the menu of options, and call the function to request and accept the User's selection. Also, if the User wants to exit, we put that option at the top.

```

314.     # Load Inventory.
315.     if str_choice == 'l':
316.         FileProcessor.load_file()
317.         continue # start loop back at top.
318.
319.     # Add a CD.
320.     elif str_choice == 'a':
321.         # Ask user for new ID, CD Title and Artist,
322.         int_id_input, str_title_input, str_artist_input = IO.input_cd_info()
323.         # Add CD information to the Inventory
324.         DataProcessor.add_cd(int_id_input, str_title_input, str_artist_input)
325.         continue # start loop back at top.
326.
327.     # Display current inventory.
328.     elif str_choice == 'i':
329.         IO.show_inventory()
330.         continue # start loop back at top.
331.
332.     # Delete a CD.
333.     elif str_choice == 'd':
334.         DataProcessor.delete_cd()
335.         continue # start loop back at top.
336.
337.     # Save inventory to file.
338.     elif str_choice == 's':
339.         FileProcessor.save_file()
340.         continue # start loop back at top.
341.
342.     # A catch-all, which should not be possible, as user choice gets
343.     # vetted in IO, but to be safe.
344.     else:
345.         print(str_general_error)

```

Figure 5 - The rest of the Main Loop

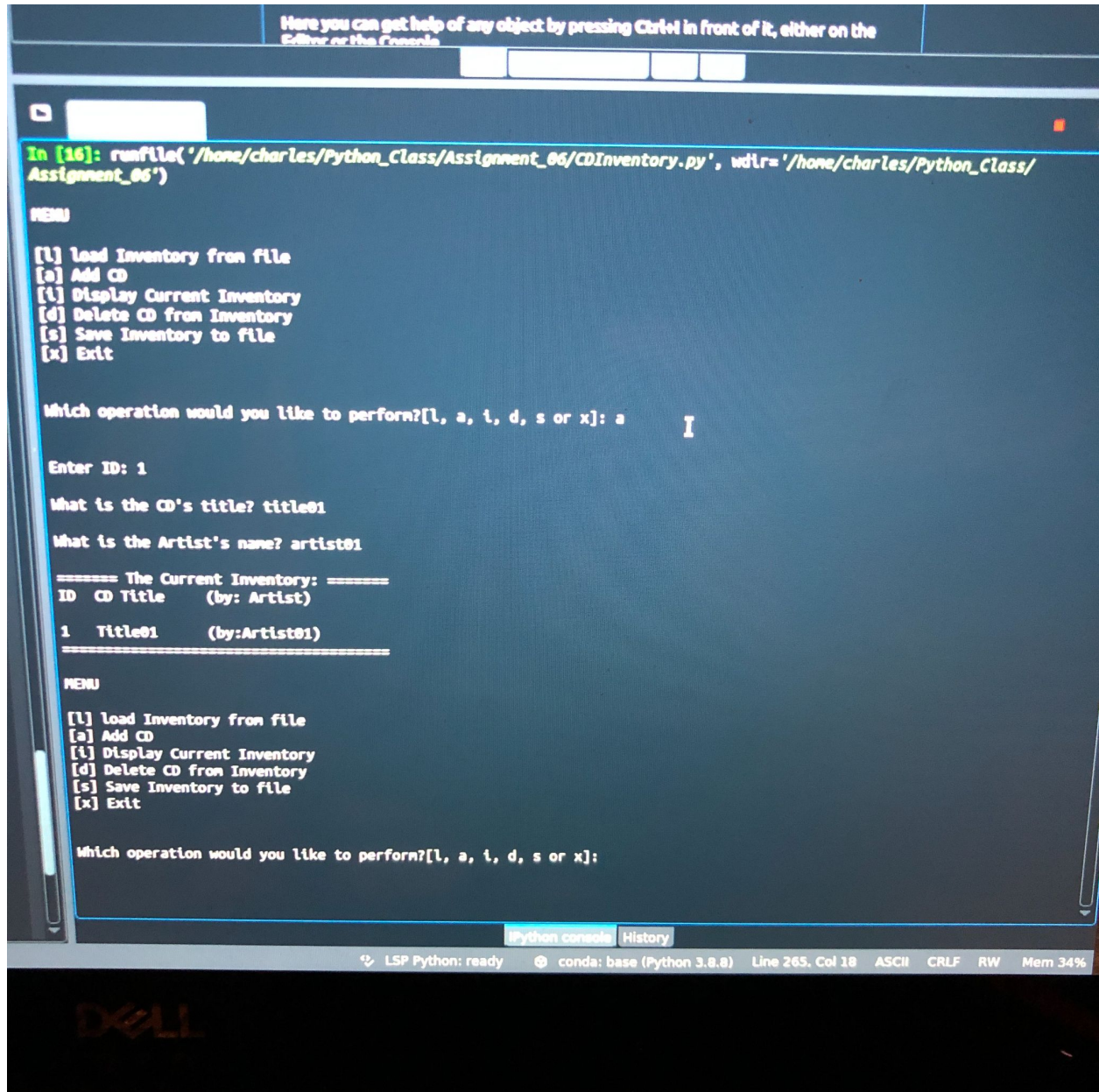
The rest of the loop is made up of specific function calls, based on the User's input. No matter what they select from the rest of the main loop's options, they may continue to interact with the Inventory, even after they've completed a task.

## Summary

In this program, we were given a sample solution to Assignment 05, and asked to amend it to include functions. We were asked to add some functions to the already defined classes, and format the script and output correctly. Most of the time that I spent on this assignment was on formatting variables to comply with the Google PyGuide. Hope that was correct.

# Script Running

## Spyder



```
In [16]: runfile('/home/charles/Python_Class/Assignment_06/CDInventory.py', wdir='/home/charles/Python_Class/Assignment_06')

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: 1

What is the CD's title? title01

What is the Artist's name? artist01

===== The Current Inventory: =====
ID  CD Title      (by: Artist)
1   Title01      (by:Artist01)
=====

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]:
```

Python console History

LSP Python: ready conda: base (Python 3.8.8) Line 265, Col 18 ASCII CRLF RW Mem 34%

Figure 6 - Running in Spyder



```
Enter at the console
Press Ctrl+C to stop the program by pressing Ctrl+C in front of it, either on the

1 Title01 (by:Artist01)
=====
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 Title01 (by:Artist01)
=====

Which CD would you like to delete? Please use ID: 1
The CD was removed.

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

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]: |

Python console History
LSP Python: ready conda: base (Python 3.8.8) Line 265, Col 18 ASCII CRLF RW M
```

Figure 7 - Running in Spyder

```
Editor or the Console
Python Package
File
Help

What is the Artist's name? artist02

===== The Current Inventory: =====
ID  CD Title      (by: Artist)
1   Title01      (by:Artist01)
2   Title02      (by:Artist02)
=====

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   Title01      (by:Artist01)
2   Title02      (by:Artist02)
=====

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

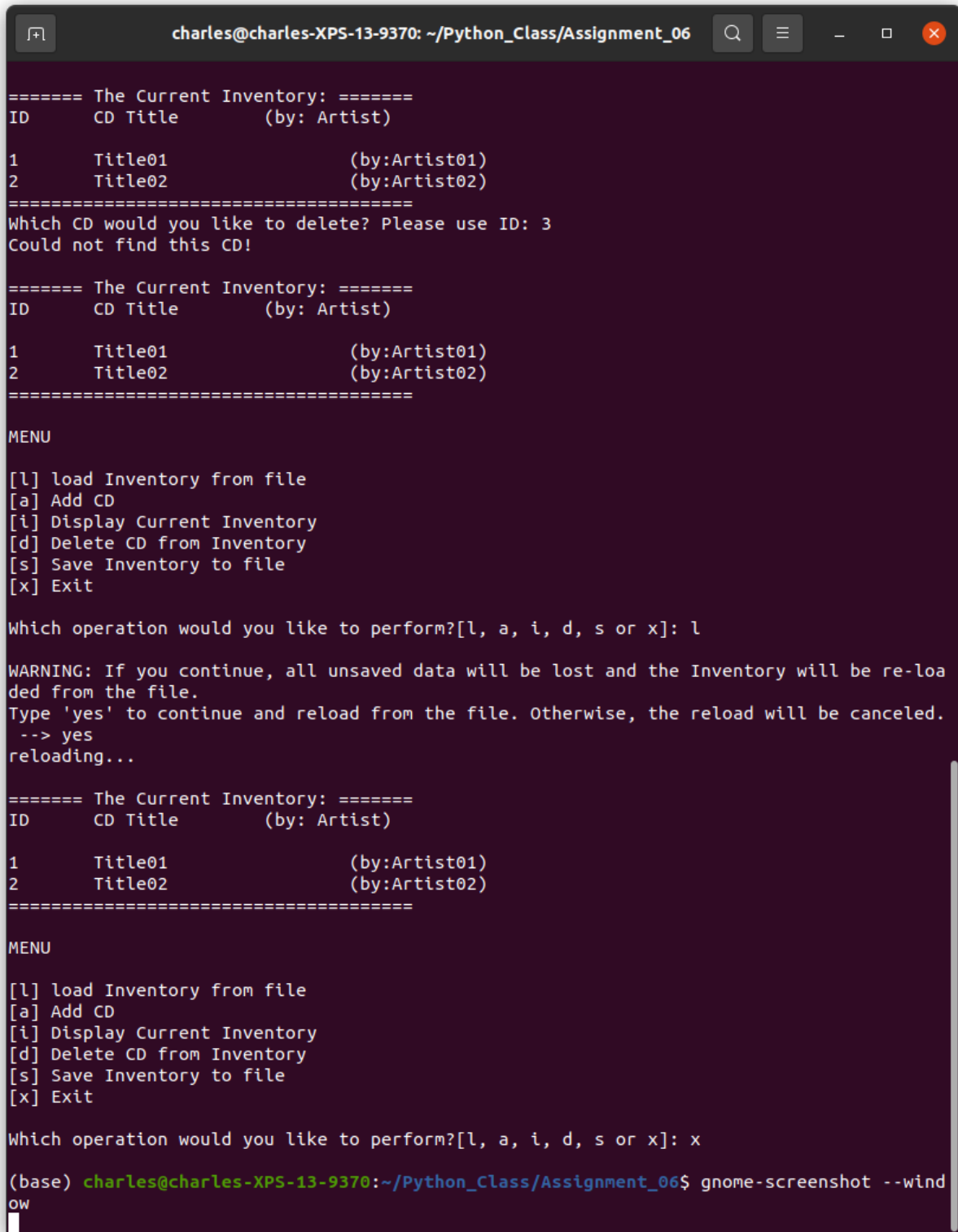
Which operation would you like to perform?[l, a, i, d, s or x]:

IPython console History
LSP Python: ready  conda: base (Python 3.8.8)  Line 265, Col 18  ASCII  CRLF
```

Figure 8 - Running in Spyder



## Terminal



```
charles@charles-XPS-13-9370: ~/Python_Class/Assignment_06

===== The Current Inventory: =====
ID      CD Title      (by: Artist)
1       Title01       (by:Artist01)
2       Title02       (by:Artist02)
=====
Which CD would you like to delete? Please use ID: 3
Could not find this CD!

===== The Current Inventory: =====
ID      CD Title      (by: Artist)
1       Title01       (by:Artist01)
2       Title02       (by:Artist02)
=====

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 will be re-loaded from the file.
Type 'yes' to continue and reload from the file. Otherwise, the reload will be canceled.
--> yes
reloading...

===== The Current Inventory: =====
ID      CD Title      (by: Artist)
1       Title01       (by:Artist01)
2       Title02       (by:Artist02)
=====

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

(base) charles@charles-XPS-13-9370:~/Python_Class/Assignment_06$ gnome-screenshot --window
```

Figure 9 - Running in the Terminal

```
charles@charles-XPS-13-9370: ~/Python_Class/Assignment_06
Enter ID: 1
What is the CD's title? title01
What is the Artist's name? artist01

===== The Current Inventory: =====
ID      CD Title      (by: Artist)
1       Title01       (by:Artist01)
2       Title02       (by:Artist02)
1       Title01       (by:Artist01)
=====

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       Title01       (by:Artist01)
2       Title02       (by:Artist02)
1       Title01       (by:Artist01)
=====
Which CD would you like to delete? Please use ID: 1
The CD was removed.

===== The Current Inventory: =====
ID      CD Title      (by: Artist)
2       Title02       (by:Artist02)
1       Title01       (by:Artist01)
=====

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

(base) charles@charles-XPS-13-9370:~/Python_Class/Assignment_06$ gnome-screenshot --wind
OW
```

Figure 10 - Running in the Terminal

```
charles@charles-XPS-13-9370: ~/Python_Class/Assignment_06

[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 will be re-loaded from the file.
Type 'yes' to continue and reload from the file. Otherwise, the reload will be canceled.
--> yes
reloading...

===== The Current Inventory: =====
ID      CD Title          (by: Artist)
1       Title01           (by:Artist01)
2       Title02           (by:Artist02)
=====

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       Title01           (by:Artist01)
2       Title02           (by:Artist02)
=====

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

(base) charles@charles-XPS-13-9370:~/Python_Class/Assignment_06$ gnome-screenshot --wind
```

Figure 11 - Running in the Terminal

# Appendix

```
1. # -----#
2. # Title: CDInventory.py
3. # Desc: Working with classes and functions.
4. # Change Log: (Who, When, What)
5. # Charles Hodges(hodges11@uw.edu), 2021-Aug-15, Created File
6. # -----#
7.
8.
9. # Variables
10. dic_row = {} # List of data row
11. lst_input_options = ['l', 'a', 'i', 'd', 's', 'x']
12. lst_tbl = [] # List of lists to hold data
13. obj_file = None # file object
14.
15. # Strings
16. str_cancelling_reload = (
17.     'Canceling... Inventory data NOT reloaded. '
18.     'Press [ENTER] to continue to the menu. \n'
19. )
20. str_cd_removed = 'The CD was removed.'
21. str_choice = '' # User input
22. str_confirm_reload = (
23.     'Type \'yes\' to continue and reload from the file. '
24.     'Otherwise, the reload will be canceled. --> '
25. )
26. str_file_name = 'CDInventory.txt' # The data storage file
27. str_footer = '===== '
28. str_general_error = '!General Error!'
29. str_header = '\n===== The Current Inventory: ====='
30. str_inventory_not_saved = (
31.     'The inventory was NOT saved to file.'
32.     'Press [ENTER] to return to the menu.'
33. )
34. str_menu = (
35.     '\n'
36.     'MENU\n\n'
37.     '[l] load Inventory from file\n'
38.     '[a] Add CD\n'
39.     '[i] Display Current Inventory\n'
40.     '[d] Delete CD from Inventory\n'
41.     '[s] Save Inventory to file\n'
42.     '[x] Exit\n'
43. )
44. str_not_find_cd = 'Could not find this CD!'
45. str_reloading = 'reloading...'
46. str_save_inventory = 'Save this inventory to file? [y/n] '
47. str_sub_header = 'ID\tCD Title \t(by: Artist)\n'
48. str_what_artist = 'What is the Artist\'s name? '
49. str_what_id = 'Enter ID: '
50. str_what_title = 'What is the CD\'s title? '
```

```

51. str_which_delete = 'Which CD would you like to delete? Please use ID: '
52. str_which_operation = (
53.     'Which operation would you like to perform?'
54.     '[1, a, i, d, s or x]: '
55. )
56. str_warning = (
57.     'WARNING: If you continue, all unsaved data will be lost and the '
58.     'Inventory will be re-loaded from the file.'
59. )
60.
61.
62. # -- PROCESSING -- #
63. class DataProcessor:
64.     """Processing the data in the table, before file interaction"""
65.
66.     @staticmethod
67.     def add_cd(int_id_input, str_title_input, str_artist_input):
68.         """Function to manage data ingestion from User input of CD info.
69.
70.         Accepts the User input of new CD information, and creates a dictionary
71.         object, which is appended to the list table which makes up the
72.         Inventory.
73.
74.         Args:
75.             str_id_input (int):
76.             str_title_input (string):
77.             str_artist_input (string):
78.
79.         Returns:
80.             None.
81.         """
82.         dic_row = {
83.             'ID': int_id_input,
84.             'Title': str_title_input.title(),
85.             'Artist': str_artist_input.title()
86.         }
87.         lst_tbl.append(dic_row)
88.         IO.show_inventory()
89.
90.     @staticmethod
91.     def delete_cd():
92.         """Function to identify and then delete a CD from the Inventory.
93.
94.         When the User selects a CD to delete, by ID, that CD is deleted from
95.         the Inventory.
96.
97.         Args:
98.             None.
99.
100.        Returns:
101.            None.
102.        """
103.        # Display Inventory to user

```

```

104.         IO.show_inventory()
105.         # Ask user which ID to remove
106.         int_id_del = int(input(str_which_delete).strip())
107.         # Search thru table and delete CD
108.         int_row_nr = -1
109.         bln_cd_removed = False
110.         for row in lst_tbl:
111.             int_row_nr += 1
112.             if row['ID'] == int_id_del:
113.                 del lst_tbl[int_row_nr]
114.                 bln_cd_removed = True
115.                 break
116.         if bln_cd_removed:
117.             print(str_cd_removed)
118.         else:
119.             print(str_not_find_cd)
120.         # Display Inventory to user again
121.         IO.show_inventory()
122.
123.
124.     class FileProcessor:
125.         """Processing the data to and from text file"""
126.
127.         @staticmethod
128.         def read_file():
129.             """Function to manage data ingestion from file to a list of
130.             dictionaries.
131.
132.             Reads the data from file identified by file_name into a 2D table
133.             (list of dicts) table one line in the file represents one dictionary
134.             row in table.
135.
136.             Args:
137.                 None.
138.
139.             Returns:
140.                 None.
141.             """
142.             # This code clears existing data, and loads data from file
143.             lst_tbl.clear()
144.             with open(str_file_name, 'r') as obj_file:
145.                 for line in obj_file:
146.                     data = line.strip().split(',')
147.                     dic_row = {
148.                         'ID': int(data[0]),
149.                         'Title': data[1],
150.                         'Artist': data[2]
151.                     }
152.                     lst_tbl.append(dic_row)
153.
154.         @staticmethod
155.         def load_file():
156.             """Function to manage data ingestion from file to a list of

```



```

157.         dictionaries, when initiated by the User, from the menu.
158.
159.     Reads the data from file identified by file_name into a 2D table
160.     (list of dicts) table one line in the file represents one dictionary
161.     row in table.
162.
163.     Args:
164.         None.
165.
166.     Returns:
167.         None.
168.     """
169.     print(str_warning)
170.     str_yes_no = input(str_confirm_reload)
171.     if str_yes_no.lower() == 'yes':
172.         print(str_reloading)
173.         FileProcessor.read_file()
174.         IO.show_inventory()
175.     else:
176.         input(str_cancelling_reload)
177.
178. @staticmethod
179. def save_file():
180.     """Function to save a file.
181.
182.     When the User decides to write the current Inventory to a file, after
183.     any edits, this function is used.
184.
185.     Args:
186.         None.
187.
188.     Returns:
189.         None.
190.     """
191.     # Display current inventory and ask user for confirmation to save
192.     IO.show_inventory()
193.     str_yes_no = input(str_save_inventory).strip().lower()
194.     # Process choice
195.     if str_yes_no == 'y':
196.         # Save data
197.         obj_file = open(str_file_name, 'w')
198.         for row in lst_tbl:
199.             lst_values = list(row.values())
200.             lst_values[0] = str(lst_values[0])
201.             obj_file.write(','.join(lst_values) + '\n')
202.         obj_file.close()
203.     else:
204.         input(str_inventory_not_saved)
205.
206. @staticmethod
207. def create_file():
208.     """Function to create a file if there is none, already.
209.

```

```

210.         Since Write or Append are the only two ways to open/create a file
211.         if it has not yet been created, we use Append, as it will not
212.         overwrite any data, if it has already been created. This function
213.         creates and closes, or merely opens and closes the text file.
214.
215.         Args:
216.             None.
217.
218.         Returns:
219.             None.
220.         """
221.         obj_file = open(str_file_name, 'a')
222.         obj_file.close()
223.
224.
225. # -- PRESENTATION (Input/Output) -- #
226.
227. class IO:
228.     """Handling Input / Output"""
229.
230.     @staticmethod
231.     def print_menu():
232.         """Displays a menu of choices to the user
233.
234.         Args:
235.             None.
236.
237.         Returns:
238.             None.
239.         """
240.         print(str_menu)
241.
242.     @staticmethod
243.     def menu_choice():
244.         """Gets user input for menu selection
245.
246.         Args:
247.             None.
248.
249.         Returns:
250.             None.
251.         """
252.         choice = ' '
253.         while choice not in lst_input_options:
254.             choice = input(str_which_operation).lower().strip()
255.             print() # Add extra space for layout
256.             return choice
257.
258.     @staticmethod
259.     def show_inventory():
260.         """Displays current inventory table
261.
262.         Args:

```

```

263.         None.
264.
265.     Returns:
266.         None.
267.
268.     """
269.     print(str_header)
270.     print(str_sub_header)
271.     for row in lst_tbl:
272.         print('{}\t{} \t\t(by:{})'.format(*row.values()))
273.     print(str_footer)
274.
275. @staticmethod
276. def input_cd_info():
277.     """Requests and receives CD information from the User.
278.
279.     Args:
280.         None.
281.
282.     Returns:
283.         int_id_input(int): ID Number
284.         str_title_input(string): CD Title
285.         str_artist_input(string): Artist Name
286.     """
287.     int_id_input = int(input(str_what_id).strip())
288.     str_title_input = input(str_what_title).strip()
289.     str_artist_input = input(str_what_artist).strip()
290.     return int_id_input, str_title_input, str_artist_input
291.
292.
293. # When the program starts, read in the currently saved Inventory, if it exists.
294. # Otherwise, create the inventory file.
295. try:
296.     FileProcessor.read_file()
297. except FileNotFoundError:
298.     FileProcessor.create_file()
299.
300.
301. # Start main loop
302. while True:
303.     # Display Menu to user, and get choice
304.     IO.print_menu()
305.     str_choice = IO.menu_choice()
306.
307.     # Exit
308.     if str_choice == 'x':
309.         break
310.
311.     # Load Inventory.
312.     if str_choice == 'l':
313.         FileProcessor.load_file()
314.         continue # start loop back at top.
315.

```

```

316.     # Add a CD.
317.     elif str_choice == 'a':
318.         # Ask user for new ID, CD Title and Artist,
319.         int_id_input, str_title_input, str_artist_input = IO.input_cd_info()
320.         # Add CD information to the Inventory
321.         DataProcessor.add_cd(int_id_input, str_title_input, str_artist_input)
322.         continue # start loop back at top.
323.
324.     # Display current inventory.
325.     elif str_choice == 'i':
326.         IO.show_inventory()
327.         continue # start loop back at top.
328.
329.     # Delete a CD.
330.     elif str_choice == 'd':
331.         DataProcessor.delete_cd()
332.         continue # start loop back at top.
333.
334.     # Save inventory to file.
335.     elif str_choice == 's':
336.         FileProcessor.save_file()
337.         continue # start loop back at top.
338.
339.     # A catch-all, which should not be possible, as user choice gets
340.     # vetted in IO, but to be safe.
341.     else:
342.         print(str_general_error)

```