Amanda Larson

December 12, 2021

IT FDN 110 – Foundations of Programming: Python

Assignment 09_B

Module 09_B – Assignment 09_B

Introduction

The last (I think) assignment of the CD_Inventory project was to put all the pieces together, or rather, disassemble them and create modules that are separate files but that work together by being called in by name. This proved to be quite the task. I spent well over 10 hours on completing the Lab A and most of what I could accomplish on LabB but was unable to complete the assignment due to time restrictions and personal circumstances (I've been out of work sick the last few days and now my husband and 1 year old are sick – not covid though). This assignment was especially hard and I feel defeated that I was unable to finish on time and was unable to complete all the tasks successfully.

What went well/not so well

I think I understood the concepts of modules and how it should all work together well. I understand how to set up object-based classes and properties and attributes (at least I think so!). I struggled with the complexity of the code, the multiple tabs, the circular calling of modules etc. and spent a lot of time just reading and trying to understand what the code was doing before I could really dive in to changing anything.

Summary

Overall, I wish I had had more time to spend on this assignment. I'm disappointed in myself for not being able to finish it successfully like I had with the other assignments. I was torn between turning it in late and losing points or turning it in on-time but incomplete. Ultimately, I went with on-time but incomplete because I don't have time on weeknights and this is the last week of class with another assignment coming up. I'm hoping I managed to get in enough changes to pass the course.

Appendix

I used a site called (Highlight your source code, 2021)[1] to format the code into the colored version. See below:

#---MAIN---#

```
#------------------------------------------#
# Title: CD Inventory.py
# Desc: The CD Inventory App main Module
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# DBiesinger, 2030-Jan-02, Extended functionality to add tracks
# ALarson, 2021-Dec-12, Added sub menu functionality and ability to manage CD tracks
#------------------------------------------#


import ProcessingClasses as PC
import IOClasses as IO
```

[1] https://highlight.hohli.com/index.php

```
13.
14.
15. lstFileNames = ['AlbumInventory.csv', 'TrackInventory.csv']
16. lstOfCDObjects = IO.FileIO.load_inventory(lstFileNames)
17.
18. while True:
19.     IO.ScreenIO.print_menu()
20.     strChoice = IO.ScreenIO.menu_choice()
21.
22.     if strChoice == 'x':
23.         break
24.     if strChoice == 'l':
25.         print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from
    file.')
26.         strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be
    canceled')
27.         if strYesNo.lower() == 'yes':
28.             print('reloading...')
29.             lstOfCDObjects = IO.FileIO.load_inventory(lstFileNames)
30.             IO.ScreenIO.show_inventory(lstOfCDObjects)
31.         else:
32.             input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
33.             IO.ScreenIO.show_inventory(lstOfCDObjects)
34.         continue  # start loop back at top.
35.     elif strChoice == 'a':
36.         tplCdInfo = IO.ScreenIO.get_CD_info()
37.         PC.DataProcessor.add_CD(tplCdInfo, lstOfCDObjects)
38.         IO.ScreenIO.show_inventory(lstOfCDObjects)
39.         continue  # start loop back at top.
40.     elif strChoice == 'd':
41.         IO.ScreenIO.show_inventory(lstOfCDObjects)
42.         continue  # start loop back at top.
43.     elif strChoice == 'c':
44.         IO.ScreenIO.show_inventory(lstOfCDObjects)
45.         cd_idx = input('Select the CD / Album index: ')
46.         cd = PC.DataProcessor.select_cd(lstOfCDObjects, cd_idx)
47.         while True:
48.             IO.ScreenIO.print_CD_menu()
49.             strChoice = IO.ScreenIO.menu_CD_choice()
50.             if strChoice == 'a':
51.                 trkId, trkTitle, trkLength = IO.ScreenIO.get_track_info()
52.                 PC.DataProcessor.add_track(trkId, trkTitle, trkLength)
53.                 IO.ScreenIO.show_tracks(cd)
54.             elif strChoice == 'd':
55.                 IO.ScreenIO.show_tracks(cd)
56.             elif strChoice == 'r':
57.                 IO.ScreenIO.show_tracks(cd)
58.                 trk_remv = input('Which track would you like to remove?')
59.                 cd.rmv_track(trk_remv)
60.             elif strChoice == 'x':
```

```
                break
            else:
                raise Exception('Please select a sub-menu selection (a, d, r or x).')
    elif strChoice == 's':
        IO.ScreenIO.show_inventory(lstOfCDObjects)
        strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
        if strYesNo == 'y':
            IO.FileIO.save_inventory(lstFileNames, lstOfCDObjects)
        else:
            input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
        continue  # start loop back at top.
    else:
        print('General Error')
```

#---Data Classes---#

```
#------------------------------------------#
# Title: Data Classes
# Desc: A Module for Data Classes
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# DBiesinger, 2030-Jan-02, Modified to add Track class, added methods to CD class to handle tracks
# ALarson, 2021-Dec-12, Modified to add class Track, attributes extended in CD Class, add_track and
#                       rmv_track added to CD Class
#------------------------------------------#

if __name__ == '__main__':
    raise Exception('This file is not meant to run by itself')

class Track():
    """Stores Data about a single Track:

    properties:
        position: (int) with Track position on CD / Album
        title: (str) with Track title
        length: (str) with length / playtime of Track
    methods:
        get_record() -> (str)

    """

       #---Fields---#
    trknum = None
    trktitle = ''
    trklen = ''

    #---Constructor---#
    def __init__(self, trknum, trktitle, trklen):
```

```
33.         #---Attributes---#
34.         self.__trackNum = trknum
35.         self.__trackTitle = trktitle
36.         self.__trackLen = trklen
37.
38.     #---Properties---#
39.
40.     #trackPosition  getter/setter
41.     @property
42.     def trackNum(self):
43.         return self.__trackNum
44.
45.     @trackNum.setter
46.     def trackNum(self, trknum):
47.         if str(trknum).isnumeric():
48.             return trknum
49.         else:
50.             raise Exception('Track number must be a number')
51.
52.     #trackTitle getter/setter
53.     @property
54.     def trackTitle(self):
55.         return self.__trackTitle
56.
57.     @trackTitle.setter
58.     def trackTitle(self, trktitle):
59.         if str(trktitle).isnumeric():
60.             raise Exception('Track title must be a string')
61.         else:
62.             return self.__trktitle
63.
64.     #trackLength getter/setter
65.     @property
66.     def trackLen(self):
67.         return self.__trackLen
68.
69.     @trackLen.setter
70.     def trackLen(self, trklen):
71.         if str(trklen).isnumeric():
72.             raise Exception('Track length must be a string')
73.         else:
74.             return self.__trklen
75.
76.
77.     #---Methods---#
78.     def __str__(self):
79.         """Returns Track details as formatted string"""
80.
81.         trkinfo = str(self.__trackNum) + ', ' + self.__trackTitle + ', ' + self.__trackLen +'\n'
82.         return trkinfo
```

```
83.
84.     def get_record(self) -> str:
85.         """Returns: Track record formatted for saving to file"""
86.         trkinfo = str(self.__trackNum) + ', ' + self.__trackTitle + ', ' + self.__trackLen +'\n'
87.         return trkinfo
88.
89.
90. class CD:
91.     """Stores data about a CD / Album:
92.
93.     properties:
94.         cd_id: (int) with CD  / Album ID
95.         cd_title: (string) with the title of the CD / Album
96.         cd_artist: (string) with the artist of the CD / Album
97.         cd_tracks: (list) with track objects of the CD / Album
98.     methods:
99.         get_record() -> (str)
100.                add_track(track) -> None
101.                rmv_track(int) -> None
102.                get_tracks() -> (str)
103.                get_long_record() -> (str)
104.
105.            """
106.
107.            # -- Constructor -- #
108.            def __init__(self, cd_id: int, cd_title: str, cd_artist: str) -> None:
109.                """Set ID, Title and Artist of a new CD Object"""
110.                #    -- Attributes  -- #
111.                try:
112.                    self.__cd_id = int(cd_id)
113.                    self.__cd_title = str(cd_title)
114.                    self.__cd_artist = str(cd_artist)
115.                    self.__tracks = []
116.                except Exception as e:
117.                    raise Exception('Error setting initial values:\n' + str(e))
118.
119.            # -- Properties -- #
120.            # CD ID
121.            @property
122.            def cd_id(self):
123.                return self.__cd_id
124.
125.            @cd_id.setter
126.            def cd_id(self, value):
127.                try:
128.                    self.__cd_id = int(value)
129.                except Exception:
130.                    raise Exception('ID needs to be Integer')
131.
132.            # CD title
```

```
133.            @property
134.            def cd_title(self):
135.                return self.__cd_title
136.
137.            @cd_title.setter
138.            def cd_title(self, value):
139.                try:
140.                    self.__cd_title = str(value)
141.                except Exception:
142.                    raise Exception('Title needs to be String!')
143.
144.            # CD artist
145.            @property
146.            def cd_artist(self):
147.                return self.__cd_artist
148.
149.            @cd_artist.setter
150.            def cd_artist(self, value):
151.                try:
152.                    self.__cd_artist = str(value)
153.                except Exception:
154.                    raise Exception('Artist needs to be String!')
155.
156.            # CD tracks
157.            @property
158.            def cd_tracks(self):
159.                return self.__tracks
160.
161.            # -- Methods -- #
162.            def __str__(self):
163.                """Returns: CD details as formatted string"""
164.                return '{:>2}\t{} (by: {})'.format(self.cd_id, self.cd_title, self.cd_artist)
165.
166.            def get_record(self):
167.                """Returns: CD record formatted for saving to file"""
168.                return '{},{},{}\n'.format(self.cd_id, self.cd_title, self.cd_artist)
169.
170.            def add_track(self, track: Track) -> None:
171.                """Adds a track to the CD / Album
172.
173.
174.                Args:
175.                    track (Track): Track object to be added to CD / Album.
176.
177.                Returns:
178.                    None.
179.
180.                """
181.                self.__tracks.append(track)
182.                self.__sort_tracks()
```

```
183.
184.         def rmv_track(self, trackNum: int) -> None:
185.             """Removes the track identified by track_id from Album
186.
187.
188.             Args:
189.                 track_id (int): ID of track to be removed.
190.
191.             Returns:
192.                 None.
193.
194.             """
195.             del self.__tracks[trackNum - 1]
196.             self.__sort_tracks()
197.
198.         def __sort_tracks(self):
199.             """Sorts the tracks using Track.trackNum. Fills blanks with None"""
200.             n = len(self.__tracks)
201.             for track in self.__tracks:
202.                 if (track is not None) and (n < track.trackNum):
203.                     n = track.trackNum
204.             tmp_tracks = [None] * n
205.             for track in self.__tracks:
206.                 if track is not None:
207.                     tmp_tracks[track.trackNum - 1] = track
208.             self.__tracks = tmp_tracks
209.
210.         def get_tracks(self) -> str:
211.             """Returns a string list of the tracks saved for the Album
212.
213.             Raises:
214.                 Exception: If no tracks are saved with album.
215.
216.             Returns:
217.                 result (string):formatted string of tracks.
218.
219.             """
220.             self.__sort_tracks()
221.             if len(self.__tracks) < 1:
222.                 raise Exception('No tracks saved for this Album')
223.             result = ''
224.             for track in self.__tracks:
225.                 if track is None:
226.                     result += 'No Information for this track\n'
227.                 else:
228.                     result += str(track) + '\n'
229.             return result
230.
231.         def get_long_record(self) -> str:
232.             """gets a formatted long record of the Album: Album information plus track details
```

```
233.
234.
235.                 Returns:
236.                     result (string): Formatted information about ablum and its tracks.
237.
238.                 """
239.                 result = self.get_record() + '\n'
240.                 result += self.get_tracks() + '\n'
241.                 return result
```

#---IO Classes---#

```
1.  #-----------------------------------------#
2.  # Title: IO Classes
3.  # Desc: A Module for IO Classes
4.  # Change Log: (Who, When, What)
5.  # DBiesinger, 2030-Jan-01, Created File
6.  # DBiesinger, 2030-Jan-02, Extended functionality to add tracks
7.  # ALarson, 2021-Dec-12, Modified save_inventory and load_inventory to accept a list of files
8.  #-----------------------------------------#
9.
10. if __name__ == '__main__':
11.     raise Exception('This file is not meant to run by itself')
12.
13. import DataClasses as DC
14. import ProcessingClasses as PC
15.
16.
17. lstFileNames = ['AlbumInventory.csv', 'TrackInventory.csv']
18.
19. class FileIO:
20.     """Processes data to and from file:
21.
22.     properties:
23.
24.     methods:
25.         save_inventory(file_name, lst_Inventory): -> None
26.         load_inventory(file_name): -> (a list of CD objects)
27.
28.     """
29.     @staticmethod
30.     def save_inventory(file_name: list, lst_Inventory: list) -> None:
31.         """
32.
33.
34.         Args:
35.             file_name (list): list of file names [CD Inventory, Track Inventory] that hold the data.
36.             lst_Inventory (list): list of CD objects.
```

```
37.
38.         Returns:
39.             None.
40.
41.         """
42.
43.         file_name_cd = file_name[0]
44.         file_name_track = file_name[1]
45.
46.
47.         try:
48.             with open(file_name_cd, 'w') as file:
49.                 for eachCD in lst_Inventory:
50.                     file.write(eachCD.get_record())
51.             with open(file_name_track, 'w') as file:
52.                 for eachCD in lst_Inventory:
53.                     tracks = eachCD.cd_tracks
54.                     cdid = eachCD.cd_id
55.                     for eachTrack in tracks:
56.                         strTrack = str(cdid) + ',' + eachTrack.get_record()
57.                         file.write(strTrack)
58.
59.         except Exception as e:
60.             print('There was a general error!', e, e.__doc__, type(e), sep='\n')
61.
62.
63.
64.     @staticmethod
65.     def load_inventory(file_name: list) -> list:
66.         """
67.
68.
69.         Args:
70.             file_name (list): list of file names [CD Inventory, Track Inventory] that hold the data.
71.
72.         Returns:
73.             list: list of CD objects.
74.
75.         """
76.
77.         lst_Inventory = []
78.         file_name_cd = file_name[0]
79.         file_name_track = file_name[1]
80.
81.         try:
82.             with open(file_name_cd, 'r') as file:
83.                 for line in file:
84.                     data = line.strip().split(',')
85.                     row = DC.CD(data[0], data[1], data[2])
86.                     lst_Inventory.append(row)
```

```
87.             with open(file_name_track, 'r') as file:
88.                 for line in file:
89.                     data = line.strip().split(',')
90.                     row = DC.Track(data[0], data[1], data[2])
91.                     lst_Inventory.append(row)
92.
93.         except Exception as e:
94.             print('There was a general error!', e, e.__doc__, type(e), sep='\n')
95.         return lst_Inventory
96.
97. class ScreenIO:
98.     """Handling Input / Output"""
99.
100.            @staticmethod
101.            def print_menu():
102.                """Displays a menu of choices to the user
103.
104.                Args:
105.                    None.
106.
107.                Returns:
108.                    None.
109.                """
110.
111.                print('Main Menu\n\n[l] load Inventory from file\n[a] Add CD / Album\n[d] Display Current Inventory')
112.                print('[c] Choose CD / Album\n[s] Save Inventory to file\n[x] exit\n')
113.
114.            @staticmethod
115.            def menu_choice():
116.                """Gets user input for menu selection
117.
118.                Args:
119.                    None.
120.
121.                Returns:
122.                    choice (string): a lower case sting of the users input out of the choices l, a, d, c, s or x
123.
124.                """
125.                choice = ' '
126.                while choice not in ['l', 'a', 'd', 'c', 's', 'x']:
127.                    choice = input('Which operation would you like to perform? [l, a, d, c, s or x]: ').lower().strip()
128.                print()  # Add extra space for layout
129.                return choice
130.
131.            @staticmethod
132.            def print_CD_menu():
133.                """Displays a sub menu of choices for CD / Album to the user
```

```
134.
135.                 Args:
136.                     None.
137.
138.                 Returns:
139.                     None.
140.                 """
141.
142.                 print('CD Sub Menu\n\n[a] Add track\n[d] Display cd / Album details\n[r] Remove
    track\n[x] exit to Main Menu')
143.
144.             @staticmethod
145.             def menu_CD_choice():
146.                 """Gets user input for CD sub menu selection
147.
148.                 Args:
149.                     None.
150.
151.                 Returns:
152.                     choice (string): a lower case sting of the users input out of the choices a, d, r or
    x
153.
154.                 """
155.                 choice = ' '
156.                 while choice not in ['a', 'd', 'r', 'x']:
157.                     choice = input('Which operation would you like to perform? [a, d, r or x]:
    ').lower().strip()
158.                 print()  # Add extra space for layout
159.                 return choice
160.
161.             @staticmethod
162.             def show_inventory(table):
163.                 """Displays current inventory table
164.
165.
166.                 Args:
167.                     table (list of dict): 2D data structure (list of dicts) that holds the data during
    runtime.
168.
169.                 Returns:
170.                     None.
171.
172.                 """
173.                 print('======= The Current Inventory: =======')
174.                 print('ID\tCD Title (by: Artist)\n')
175.                 for row in table:
176.                     print(row)
177.                 print('======================================')
178.
179.             @staticmethod
```

```
180.            def show_tracks(cd):
181.                """Displays the Tracks on a CD / Album
182.
183.                Args:
184.                    cd (CD): CD object.
185.
186.                Returns:
187.                    None.
188.
189.                """
190.                print('====== Current CD / Album: ======')
191.                print(cd)
192.                print('================================')
193.                print(cd.get_tracks(cd))
194.                print('================================')
195.
196.            @staticmethod
197.            def get_CD_info():
198.                """function to request CD information from User to add CD to inventory
199.
200.
201.                Returns:
202.                    cdId (string): Holds the ID of the CD dataset.
203.                    cdTitle (string): Holds the title of the CD.
204.                    cdArtist (string): Holds the artist of the CD.
205.
206.                """
207.
208.                cdId = input('Enter ID: ').strip()
209.                cdTitle = input('What is the CD\'s title? ').strip()
210.                cdArtist = input('What is the Artist\'s name? ').strip()
211.                return cdId, cdTitle, cdArtist
212.
213.            @staticmethod
214.            def get_track_info():
215.                """function to request Track information from User to add Track to CD / Album
216.
217.
218.                Returns:
219.                    trkId (string): Holds the ID of the Track dataset.
220.                    trkTitle (string): Holds the title of the Track.
221.                    trkLength (string): Holds the length (time) of the Track.
222.
223.                """
224.
225.                trkId = input('Enter Position on CD / Album: ').strip()
226.                trkTitle = input('What is the Track\'s title? ').strip()
227.                trkLength = input('What is the Track\'s length? ').strip()
228.                return trkId, trkTitle, trkLength
```

#---Processing Classes---#

```
#------------------------------------------#
# Title: Processing Classes
# Desc: A Module for processing Classes
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# DBiesinger, 2030-Jan-02, Extended functionality to add tracks
# ALarson, 2021-Dec-12, Attempted to add code to add track for functionality
#------------------------------------------#

if __name__ == '__main__':
    raise Exception('This file is not meant to ran by itself')

import DataClasses as DC

class DataProcessor:
    """Processing the data in the application"""
    @staticmethod
    def add_CD(CDInfo, table):
        """function to add CD info in CDinfo to the inventory table.


        Args:
            CDInfo (tuple): Holds information (ID, CD Title, CD Artist) to be added to inventory.
            table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.

        Returns:
            None.

        """

        cdId, title, artist = CDInfo
        try:
            cdId = int(cdId)
        except:
            raise Exception('ID must be an Integer!')
        row = DC.CD(cdId, title, artist)
        table.append(row)

    @staticmethod
    def select_cd(table: list, cd_idx: int) -> DC.CD:
        """selects a CD object out of table that has the ID cd_idx

        Args:
            table (list): Inventory list of CD objects.
            cd_idx (int): id of CD object to return

        Raises:
            Exception: If id is not in list.
```

```
49.
50.         Returns:
51.             row (DC.CD): CD object that matches cd_idx
52.
53.         """
54.         # TODO add code as required
55.         pass
56.
57.
58.     @staticmethod
59.     def add_track(track_info: tuple, cd: DC.CD) -> None:
60.         """adds a Track object with attributes in track_info to cd
61.
62.
63.         Args:
64.             track_info (tuple): Tuple containing track info (position, title, Length).
65.             cd (DC.CD): cd object the tarck gets added to.
66.
67.         Raises:
68.             Exception: DESCraised in case position is not an integer.
69.
70.         Returns:
71.             None: DESCRIPTION.
72.
73.         """
74.
75.         # TODO add code as required
76.         trkId, trkTitle, trkLength = track_info
77.         pass
```