Andrew Yeo

March 11, 2020

Foundations of Programming: Python

Github Repository: <a href="https://github.com/andyhyeo/assignment\_08">https://github.com/andyhyeo/assignment\_08</a>

## Module 08

## Knowledge Document

#### Intro

In this week's assignment, we go over Object Oriented Programming. Object Oriented Programming makes use of classes which are the blueprints for objects. The class maintains then data and the operations of the object. The object is an "instantiation" of the class. And the "features" of the objects are inherited from its class (attributes, properties, and methods)

#### **Details**

In this week's assignment we are instructed to build the CD inventory script from scratch. I largely already used what the script that we have been using in order to focus on the parts that were the topics of this weeks modules: object oriented programming, defining classes, methods, properties, and attributes.

1

#### Class CD

```
11 # -- DATA -- #
12 # DBiesinger, 2030-Jan-01, created file
13 from os import path
14 strFileName = 'cdInventory.txt'
15 lstOfCDObjects = []
16
17 class CD:
       """Stores data about a CD:
18
19
20
       properties:
21
            cd_id: (int) with CD ID
22
            cd_title: (string) with the title of the CD
23
            cd_artist: (string) with the artist of the CD
24
       methods:
25
       .....
26
       # -- Contructor / Initializer -- #
def __init__(self, cd_id, cd_title, cd_artist):
27
28
29
            self.id = cd_id
            self.title = cd_title
30
31
            self.artist = cd_artist
```

Figure 1: Building the CD class with attributes of id, title, and artist.

The CD class was built simply by adding parameters. Even though I have gotten a little bit of purchase on Object Oriented Programming concepts, I still had to build this largely by emulating the script within the module notes. I tested it within the console to make sure that it probably stored data when called.

```
58
      @staticmethod
59
      def load_inventory(file_name, lst0fCD0bjects):
           """Function to manage data ingestion from file to a list of dictionaries
61
62
           Reads the data from file identified by file_name into a 2D table
           (list of dicts) table one line in the file represents one dictionary row in table.
63
64
65
               file_name (string): name of file used to read the data from
66
               table (list of dict): 2D data structure (list of dicts) that holds the data during
67
68
69
           Returns:
70
               None.
71
72
73
               if path.exists('cdInventory.txt'): #Corrects for a bug in the original file whereb
74
                    lstOfCDObjects.clear() # this clears existing data and allows to load data from
75
                   objFile = open(file_name, 'r')
76
                   for line in objFile:
                        data = line.strip().split(',')
dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
77
78
79
                        lstOfCDObjects.append(dicRow)
80
                   objFile.close()
81
                   return lstOfCDObjects
           except FileNotFoundError:
82
```

Figure 2: Building the FileIO class with the method of load inventory.

#### Class File IO with Method load\_inventory

Figure 2 shows the part of the script where I built the load\_inventory method of within the class FileIO. The module goes over concepts and best practices of writing class such as writing constructors, attributes, and properties but I was not precisely sure how to apply that and if I should have applied it. It was largely a copy and paste from the previous scripts. For simplicity, I opted to use the non-pickling loading strategy. And as shown, I wrote in the structured error handling to continue with the script if the file is not found.

#### Class File IO with Method save\_inventory

```
@staticmethod
87
       def save_inventory(file_name, table):
 88
            """Writes the inventory of IDs, CD Names, and Artists to a text file
 89
 90
 91
                file_name (string): The name of the file that it will write to
 92
                table (list of dict): 2D data structure (list of dicts) that holds the data during
 93
 94
 95
                None but saves a file in the directory of the python script
 96
 97
98
           objFile = open(file_name, 'w')
99
           for row in table:
100
                strRow =
101
                for item in row.values():
102
                    strRow += str(item) +
                strRow = strRow[:-1] + '\n'
103
104
                objFile.write(strRow)
105
           objFile.close()
```

Figure 3: Building the FileIO class with the method of save\_inventory.

The load\_inventory script was an amalgamation of previous scripts and so to was the the save\_inventory.

## Try-Except Error Structure Handling for ValueError in IO.ask() function

I also successfully substituted the try-except structure within the IO.ask() function which took a fair amount of trial error. I am learning that I can use all of the Spyder console in order to help me debug a code. For instance, I can run just line by line of the code. I can use the variable explorer as feedback if a variable was defined or changed the way that I wanted to.

```
181
       @staticmethod
182
       def ask():
            """Ask user for new ID, CD Title and Artist
183
184
           Args:
               None
186
187
           Returns:
188
189
                dicRow (dictionary): A dictionary entry with ID (int): integer that holds
                the ID tag, title (string): string that holds the name of the CD
190
191
                    and an artist (string): string that holds the name of the Artist.
192
193
           while True:
194
                strID = input('Enter ID: ').strip()
195
                try:
196
                    strID = int(strID)
197
                    break
198
                except ValueError:
                    print('That is not an integer')
            strTitle = input('What is the CD\'s title? ').strip()
200
            stArtist = input('What is the Artist\'s name? ').strip()
201
            dicRow = {'ID': strID, 'CD Title': strTitle, 'Artist': stArtist}
202
203
204
            return dicRow
```

Figure 4: Substituting Structured Error Handling in the Ask Function

# Try-Except Error Structure Handling to handle not-integers when deleting

Figure 5 shows the part of the CD Inventory script which prompts the user to that to put in an integer when they have not done so successfully.

```
elif strChoice == 'd':
247
248
           # 3.5.1 get Userinput for which CD to delete
249
           # 3.5.1.1 display Inventory to user
250
           IO.show_inventory(lstTbl)
251
           # 3.5.1.2 ask user which ID to remove
252
           while True:
253
               try:
254
                    intIDDel = int(input('Which ID would you like to delete? ').strip())
255
                    break
               except ValueError:
256
257
                    print('That is not an integer')
           # 3.5.2 search thru table and delete CD
258
259
           DataProcessor.delete_cd(intIDDel,lstTbl)
260
           IO.show_inventory(lstTbl)
261
           continue # start loop back at top.
```

Figure 4: Error Structure Handling to handle not-integers when prompted to delete

### FileIO.load\_inventory

Figure 5: Reading Files into the Script

```
85 # -- PROCESSING -- #
 86 class FileIO:
 87
        """Processes data to and from file:
 88
 89
        properties:
 90
 91
        methods:
            save_inventory(file_name, lst_Inventory): -> None
 92
 93
            load_inventory(file_name): -> (a list of CD objects)
 94
 95
 96
        @staticmethod
 97
        def load_inventory(file_name):
 98
            """Function to manage data ingestion from file to a list of dictionaries
 99
100
            Reads the data from file identified by file_name into a 2D table
101
            (list of dicts) table one line in the file represents one dictionary row in tab
102
103
                file_name (string): name of file used to read the data from
104
105
                table (list of dict): 2D data structure (list of dicts) that holds the data
106
107
            Returns:
108
                None.
            .....
109
110
                with open(file_name, 'rb') as objFile:
111
                    table = pickle.load(objFile)
112
113
                     return table
114
            except FileNotFoundError:
115
                pass
116
```

The FileIO class is populated with the load\_inventory function which was largely a derivative of the pickle method. After researching online, it looks as if pickling was the best way to manage the reading of files into the script. This is corroborated by the fact that we went over pickling in preceding courses.

### FileIO.save\_inventory - The File Saving Function

```
117
       @staticmethod
118
       def save_inventory(file_name, table):
            """Writes the inventory of IDs, CD Names, and Artists to a text file
119
120
121
122
                file_name (string): The name of the file that it will write to
123
                table (list of dict): 2D data structure (list of dicts) that holds the
124
125
               None but saves a file in the directory of the python script
126
127
128
129
           with open(file_name, 'wb') as objFile:
130
                table = pickle.dump(table, objFile)
```

Figure 6: Writing Files from the Script

The save\_inventory method of the FileIO class is largely a derivative from a subsequent scripts pickling method. This seemed to be the most viable solution for handle class and objects.

## FileIO.show\_inventory - The Inventory Showing Function

Figure 7: Showing Inventory

```
167
       @staticmethod
168
       def show_inventory(table):
            """Displays current inventory table
169
170
171
172
           Args:
                table (list of dict): 2D data structure (list of dicts) that holds the
173
174
175
            Returns:
176
                None.
177
178
            print('====== The Current Inventory: ======')
179
180
            print('ID\tCD Title (by: Artist)\n')
            for CD in table:
                print('{}\t{} (by:{})'.format(CD.ID, CD.title, CD.artist))
```

Douglas, our class TA, wrote all of us to address a question that many of us must have posed to him - "Should we start from scratch or should we adjust previously written code?" As you can probably tell, I took the latter strategy. This code was largely readjusted to call on the CDs to display.

### IO.addCD - The CD adding method

The CD adding method, ask Douglas graciously explained, is a simple method of the table object.

Figure 8: Add CD

```
185
       @staticmethod
186
       def add_cd(row, table):
            """Adds a dictionary row to the inventory
187
188
189
            Args:
190
                row (dictionary): dictionary that holds the name of the ID, cd, and a
191
                table (list of dict): 2D data structure (list of dicts) that holds th
192
193
            Returns:
194
                None.
195
196
197
            table.append(row)
198
            return table
```

## IO.ask - The CD information entering method

Figure 9: Add CD

```
201
        @staticmethod
202
         def ask():
             """Ask user for new ID, CD Title and Artist
203
204
205
             Args:
206
                  None
207
             Returns:
208
                  CD (Object): A dictionary entry with ID (int): integer that holds the ID tag, title (string): string that holds the name of the CD
209
210
211
                       and an artist (string): string that holds the name of the Artist.
212
             title = ''
213
             artist = ''
214
215
             while True:
216
                  CD_id = input('Enter ID: ').strip()
217
                       CD_id = int(CD_id)
218
219
                       break
220
                  except ValueError:
                       print('That is not an integer')
221
             title = input('What is the CD\'s title? ').strip()
222
             artist = input('What is the Artist\'s name? ').strip()
223
224
             CD_row = CD(CD_id, title, artist)
225
             return CD_row
226
```

The CD adding method takes advantage of the CD class by entering information through the CD object.

#### Summary

As Dirk mentions in his notes and lectures, I do not yet follow appreciate the power of Object Oriented Programming. Dirk mentions that it takes more than just one time using it appreciate its power and beauty. I figure that some of the ways that I can come to appreciate it is to try and building really difficult code without it. Also as Dirk mentions, its usage is probably so standardized and par for the course that it might be difficult to "learn" it again.

Some of the future investigations can include adding the features of pickling to this code in order to increase efficiency of data storage and management. Another feature that can be looked into adding is the feature of preventing duplicate IDs to be stored.

#### References

https://www.youtube.com/watch?v=lkXfgP-fAkY&feature=youtu.be https://www.youtube.com/watch?v=KZdFvyCOLUQ&feature=youtu.be https://www.youtube.com/watch?v=qvRyls8NX-E&feature=youtu.be https://www.youtube.com/watch?v=QuwU34OT4XA&feature=youtu.be https://www.youtube.com/watch?v=swAXTwW6xoA&feature=youtu.be https://www.youtube.com/watch?v=5MM6IaESdQ0&feature=youtu.be https://realpython.com/python3-object-oriented-programming/ https://www.youtube.com/watch?v=IHaTbJPdB-s&feature=youtu.be

#### Appendix: The Code

Made using <u>Planet B's Syntax Highlighter</u>

```
1. #-----#
2. # Title: Assignmen08.py
3. # Desc: Assignnment 08 - Working with classes
4. # Change Log: (Who, When, What)
5. # AYeo, 2020-Mar-10 9:06 PM, Added working CD class
6. # AYeo, 2020-
   Mar-10 1:52 PM, Added Main Body from previous script
7. # AYeo, 2020-Mar-10 2:99 PM, Added Structured Error Handling
8. # AYeo, 2020-
   Mar-15 9:29 PM, Fixed CD class to have property, private methods,
    setters
9. # AYeo, 2020-
   Mar-15 9:29 PM, Edited FileIO to use CD class, HAVE NOT TESTED
10. # AYeo, 2020-
   Mar-15 9:29 PM, Edited FileIO to use CD class, Successfully Teste
   d
11. # AYeo, 2020-
   Mar-16 5:00 PM, Correct FileIO.load inventory to use pickle, Succ
   essfully Tested
12. # AYeo, 2020-
   Mar-16 5:00 PM, Correct FileIO.save inventory to use pickle, NOT
   TESTED
13. # AYeo, 2020-
   Mar-16 5:00 PM, Correct FileIO.save inventory to use pickle, Test
   ed
14.
15.
16. #-----#
17.
18. # -- DATA -- #
19. # DBiesinger, 2030-Jan-01, created file
20. from os import path
21. import pickle
22.
23. strFileName = 'cdInventory.txt'
24. lstOfCDObjects = []
25.
26.
27. class CD:
28. """Stores data about a CD:
29.
30. properties:
```

```
31.
            cd id: (int) with CD ID
            cd title: (string) with the title of the CD
32.
33.
            cd artist: (string) with the artist of the CD
34.
        methods:
35.
        11 11 11
36.
37.
        # -- Contructor -- #
38.
        def __init__(self, cd_id, cd_title, cd_artist):
39.
            # -- Attributes -- #
40.
            self. ID = cd id
            self.__title = cd title
41.
42.
            self. artist = cd artist
43.
44.
        # -- Properties -- #
45.
        @property
46.
        def ID(self):
            return self.__ID
47.
48.
49.
        @property
50.
        def title(self):
51.
            return self. title.title()
52.
53.
        @property
        def artist(self):
54.
55.
            return self. artist.title()
56.
57.
        @ID.setter
        def ID(self, value):
58.
59.
            if str(value).isnumeric():
                self.__ID = value
60.
61.
            else:
62.
                raise Exception('The ID must be numeric')
63.
64.
        @title.setter
        def title(self, value):
65.
            if str(value).isnumeric():
66.
67.
                raise Exception('The title must be a string')
68.
            else:
69.
                self. title = value
70.
71.
        @artist.setter
        def artist(self, value):
72.
73.
            if str(value).isnumeric():
74.
                raise Exception('The artist must be a string')
75.
            else:
                self. artist = value
76.
```

```
77.
       # -- Methods -- #
78.
        def str (self):
79.
           return self.ID
80.
            return self.title
81.
            return self.artist
82.
83.
84.
85. # -- PROCESSING -- #
86. class FileIO:
        """Processes data to and from file:
87.
88.
89.
       properties:
90.
91.
       methods:
            save inventory(file name, lst Inventory): -> None
92.
            load inventory(file name): -> (a list of CD objects)
93.
94.
        0.00
95.
96.
       @staticmethod
97.
        def load_inventory(file_name):
            """Function to manage data ingestion from file to a list
98.
   of dictionaries
99.
            Reads the data from file identified by file name into a 2
100.
   D table
            (list of dicts) table one line in the file represents one
101.
    dictionary row in table.
102.
103.
            Args:
104.
                file name (string): name of file used to read the dat
   a from
                table (list of dict): 2D data structure (list of dict
105.
   s) that holds the data during runtime
106.
107.
            Returns:
108.
                None.
            .....
109.
110.
111.
                with open(file name, 'rb') as objFile:
112.
                    table = pickle.load(objFile)
                    return table
113.
114.
            except FileNotFoundError:
115.
                pass
116.
117.
        @staticmethod
```

```
def save inventory(file name, table):
            """Writes the inventory of IDs, CD Names, and Artists to
119.
    a text file
120.
121.
            Args:
122.
                file_name (string): The name of the file that it will
    write to
123.
                table (list of dict): 2D data structure (list of dict
   s) that holds the data during runtime
124.
125.
            Returns:
126.
                None but saves a file in the directory of the python
   script
127.
128.
            with open(file name, 'wb') as objFile:
129.
                table = pickle.dump(table, objFile)
130.
131.
132.
133.# -- PRESENTATION (Input/Output) -- #
134.class IO:
        """Handling Input / Output"""
135.
136.
        @staticmethod
137.
        def print menu():
138.
            """Displays a menu of choices to the user
139.
140.
            Args:
141.
                None.
142.
143.
            Returns:
144.
                None.
145.
146.
            print('Menu\n\n[1] load Inventory from file\n[a] Add CD\n
147.
    [i] Display Current Inventory')
            print('[d] delete CD from Inventory\n[s] Save Inventory t
148.
   o file\n[x] exit\n')
149.
150.
        @staticmethod
151.
        def menu choice():
            """Gets user input for menu selection
152.
153.
154.
            Args:
155.
                None.
156.
157.
            Returns:
```

```
158. choice (string): a lower case sting of the users input
   t out of the choices l, a, i, d, s or x
159.
160.
161.
           choice = ' '
           while choice not in ['l', 'a', 'i', 'd', 's', 'x']:
162.
               choice = input('Which operation would you like to per
163.
   form? [1, a, i, d, s or x]: ').lower().strip()
           print() # Add extra space for layout
164.
165.
           return choice
166.
167.
       @staticmethod
168.
       def show inventory(table):
            """Displays current inventory table
169.
170.
171.
172.
           Args:
173.
               table (list of dict): 2D data structure (list of dict
   s) that holds the data during runtime.
174.
175.
           Returns:
176.
               None.
177.
178.
179.
           print('====== The Current Inventory: ======')
           print('ID\tCD Title (by: Artist)\n')
180.
           for CD in table:
181.
               print('{}\t{} (by:
182.
   {})'.format(CD.ID, CD.title, CD.artist))
183.
           print('======')
184.
185.
       @staticmethod
       def add_cd(row, table):
186.
            """Adds a dictionary row to the inventory
187.
188.
189.
           Args:
190.
               row (dictionary): dictionary that holds the name of t
   he ID, cd, and artist
191.
               table (list of dict): 2D data structure (list of dict
   s) that holds the data during runtime
192.
193.
           Returns:
194.
               None.
195.
196.
197.
           table.append(row)
```

```
198.
            return table
199.
200.
        @staticmethod
201.
202.
        def ask():
            """Ask user for new ID, CD Title and Artist
203.
204.
205.
            Args:
206.
                None
207.
208.
            Returns:
209.
                CD (Object): A dictionary entry with ID (int): integ
    er that holds
210.
                the ID tag, title (string): string that holds the name
    of the CD
211.
                    and an artist (string): string that holds the nam
    e of the Artist.
212.
213.
            title = ''
            artist = ''
214.
215.
            while True:
216.
                CD id = input('Enter ID: ').strip()
217.
                try:
218.
                    CD id = int(CD id)
219.
                    break
220.
                except ValueError:
                    print('That is not an integer')
221.
222.
            title = input('What is the CD\'s title? ').strip()
223.
            artist = input('What is the Artist\'s name? ').strip()
224.
            CD row = CD(CD id, title, artist)
225.
            return CD row
226.
227.class DataProcessor:
228.
        @staticmethod
        def delete cd(intIDDel,table):
229.
230.
            """Deletes a CD row from the table
231.
232.
            Args:
233.
                intIDDel (int): ID which indicate which entry user wo
   uld like to delete
234.
                table (list of dict): 2D data structure (list of dict
    s) that holds the data during runtime
235.
236.
            Returns:
                  table (list of dict): 2D data structure (list of di
237.
    cts) that holds the data during runtime
```

```
238.
239.
            intRowNr = -1
240.
            blnCDRemoved = False
            for CD in table:
241.
242.
                intRowNr += 1
                if CD.ID == intIDDel:
243.
244.
                    del table[intRowNr]
245.
                    blnCDRemoved = True
246.
                    break
247.
                if blnCDRemoved:
248.
                    print('The CD was removed')
249.
                else:
250.
                    print('Could not find this CD!')
251.
            return table
252.
253.# -- Main Body of Script -- #
254.# Load data from file into a list of CD objects on script start
255.FileIO.load inventory(strFileName)
256.
257.# 2. start main loop
258.while True:
        # 2.1 Display Menu to user and get choice
259.
260.
        IO.print menu()
        strChoice = IO.menu choice()
261.
262.
        # 3. Process menu selection
263.
        # 3.1 process exit first
264.
265.
        if strChoice == 'x':
266.
            break
267.
        # 3.2 process load inventory
268.
        if strChoice == 'l':
269.
            print('WARNING: If you continue, all unsaved data will be
    lost and the Inventory re-loaded from file.')
            strYesNo = input('type \'yes\' to continue and reload fro
270.
   m file. otherwise reload will be canceled')
271.
            if strYesNo.lower() == 'yes':
                print('reloading...')
272.
                lstOfCDObjects = FileIO.load inventory(strFileName)
273.
                IO.show inventory(lst0fCD0bjects)
274.
275.
            else:
276.
                input('canceling... Inventory data NOT reloaded. Pres
   s [ENTER] to continue to the menu.')
                IO.show_inventory(lst0fCD0bjects)
277.
278.
            continue # start loop back at top.
279.
        # 3.3 process add a CD
        elif strChoice == 'a':
280.
```

```
281.
            # 3.3.1 Ask user for new ID, CD Title and Artist
            row = I0.ask()
282.
            # 3.3.2 Add item to the table
283.
284.
            IO.add cd(row,lstOfCDObjects)
285.
            IO.show inventory(lst0fCD0bjects)
            continue # start loop back at top.
286.
        # 3.4 process display current inventory
287.
        elif strChoice == 'i':
288.
            IO.show inventory(lst0fCD0bjects)
289.
            continue # start loop back at top.
290.
        # 3.5 process delete a CD
291.
        elif strChoice == 'd':
292.
293.
            # 3.5.1 get Userinput for which CD to delete
294.
            # 3.5.1.1 display Inventory to user
            IO.show inventory(lst0fCD0bjects)
295.
            # 3.5.1.2 ask user which ID to remove
296.
297.
            while True:
298.
                try:
299.
                    intIDDel = int(input('Which ID would you like to
   delete? ').strip())
300.
                    break
                except ValueError:
301.
                    print('That is not an integer')
302.
            # 3.5.2 search thru table and delete CD
303.
304.
            DataProcessor.delete cd(intIDDel,lst0fCD0bjects)
305.
            IO.show inventory(lst0fCD0bjects)
            continue # start loop back at top.
306.
307.
        # 3.6 process save inventory to file
308.
        elif strChoice == 's':
309.
            # 3.6.1 Display current inventory and ask user for confir
   mation to save
310.
            IO.show inventory(lstOfCDObjects)
            strYesNo = input('Save this inventory to file? [y/
311.
   n] ').strip().lower()
            # 3.6.2 Process choice
312.
            if strYesNo == 'v':
313.
314.
                # 3.6.2.1 save data
315.
                FileIO.save inventory(strFileName, lstOfCDObjects)
316.
            else:
317.
                input('The inventory was NOT saved to file. Press [EN
   TER] to return to the menu.')
            continue # start loop back at top.
318.
319.
        # 3.7 catch-
   all should not be possible, as user choice gets vetted in IO, but
    to be save:
320. else:
```

321. print('General Error')
322.

#### Appendix: The Spyder Output

```
    runfile('/Users/andyyeo/Google Drive/Intro to Python/Mod 08/

   Assignment08/Assignment_08.py', wdir='/Users/andyyeo/
   Google Drive/Intro to Python/Mod_08/Assignment08')
2.
   Menu
3.
4. [1] load Inventory from file
5. [a] Add CD
6. [i] Display Current Inventory
7. [d] delete CD from Inventory
8. [s] Save Inventory to file
9. [x] exit
10.
11.
12. Which operation would you like to perform? [1, a, i, d, s or x]:
13.
14.
15. Enter ID: 1
16.
17. What is the CD's title? asdfa
18.
19. What is the Artist's name? vasdc
20. ====== The Current Inventory: ======
           CD Title (by: Artist)
21. ID
22.
23. 1
           Asdfa (by:Vasdc)
25. Menu
26.
27. [1] load Inventory from file
28. [a] Add CD
29. [i] Display Current Inventory
30. [d] delete CD from Inventory
31. [s] Save Inventory to file
32. [x] exit
33.
34.
35. Which operation would you like to perform? [1, a, i, d, s or x]:
   а
36.
37.
38. Enter ID: 2
39.
```

```
40. What is the CD's title? asc
41.
42. What is the Artist's name? casdg
43. ====== The Current Inventory: ======
          CD Title (by: Artist)
44. ID
45.
46. 1
          Asdfa (by:Vasdc)
47. 2
          Asc (by:Casdg)
49. Menu
50.
51. [1] load Inventory from file
52. [a] Add CD
53. [i] Display Current Inventory
54. [d] delete CD from Inventory
55. [s] Save Inventory to file
56. [x] exit
57.
58.
59. Which operation would you like to perform? [1, a, i, d, s or x]:
   S
60.
61. ====== The Current Inventory: ======
62. ID
          CD Title (by: Artist)
63.
64. 1
          Asdfa (by:Vasdc)
65. 2
          Asc (by:Casdg)
67.
68. Save this inventory to file? [y/n] y
69. Menu
70.
71. [1] load Inventory from file
72. [a] Add CD
73. [i] Display Current Inventory
74. [d] delete CD from Inventory
75. [s] Save Inventory to file
76. [x] exit
77.
78.
79. Which operation would you like to perform? [l, a, i, d, s or x]:
   d
80.
81. ====== The Current Inventory: ======
82. ID
          CD Title (by: Artist)
83.
```

```
84. 1
          Asdfa (by:Vasdc)
          Asc (by:Casdg)
85. 2
86. =============
87.
88. Which ID would you like to delete? 1
89. ====== The Current Inventory: ======
          CD Title (by: Artist)
90. ID
91.
92. 2
      Asc (by:Casdg)
93. ==============
94. Menu
95.
96. [1] load Inventory from file
97. [a] Add CD
98. [i] Display Current Inventory
99. [d] delete CD from Inventory
100.[s] Save Inventory to file
101.[x] exit
102.
103.
104.Which operation would you like to perform? [1, a, i, d, s or x]:
105.
106.===== The Current Inventory: ======
107.ID
          CD Title (by: Artist)
108.
          Asc (by:Casdg)
111.
112.Which ID would you like to delete? 2
113.===== The Current Inventory: ======
        CD Title (by: Artist)
114.ID
117.Menu
118.
119.[1] load Inventory from file
120.[a] Add CD
121.[i] Display Current Inventory
122.[d] delete CD from Inventory
123.[s] Save Inventory to file
124.[x] exit
125.
126.
127. Which operation would you like to perform? [1, a, i, d, s or x]:
```

```
128.
129.WARNING: If you continue, all unsaved data will be lost and the I
   nventory re-loaded from file.
130.
131.type 'yes' to continue and reload from file. otherwise reload wil
   l be canceledyes
132.reloading...
133.===== The Current Inventory: ======
        CD Title (by: Artist)
134.ID
135.
136.1
           Asdfa (by:Vasdc)
137.2
           Asc (by:Casdg)
138.===========
139.Menu
140.
141.[1] load Inventory from file
142.[a] Add CD
143.[i] Display Current Inventory
144.[d] delete CD from Inventory
145.[s] Save Inventory to file
146.[x] exit
147.
148.
149. Which operation would you like to perform? [1, a, i, d, s or x]:
150.
151.===== The Current Inventory: ======
          CD Title (by: Artist)
152.ID
153.
154.1
           Asdfa (by:Vasdc)
155.2
           Asc (by:Casdg)
157.Menu
158.
159.[1] load Inventory from file
160.[a] Add CD
161.[i] Display Current Inventory
162.[d] delete CD from Inventory
163.[s] Save Inventory to file
164.[x] exit
165.
167. Which operation would you like to perform? [1, a, i, d, s or x]:
   Х
```

#### Appendix: The Terminal Output

```
1. Last login: Wed Mar 11 14:14:33 on ttys000
2. ^[[A^[[A
3. The default interactive shell is now zsh.
4. To update your account to use zsh, please run `chsh -s /bin/
For more details, please visit https://support.apple.com/kb/
   HT208050.
6. (base) Andys-MacBook-Pro:∼ andyyeo$ cd desktop/assignment08
7. (base) Andys-MacBook-Pro:assignment08 andyyeo$ zip -r dir.zip . -
   x ".*" -x " MAXOSX"
   adding: Assignment08 Knowledge Document .pdf (deflated 29%)
9.
     adding: Assignment 08 Starter.py (deflated 71%)
10. (base) Andys-MacBook-Pro:assignment08 andyyeo$
      [Restored Mar 16, 2020 at 7:00:13 PM]
12. Last login: Wed Mar 11 15:00:02 on ttys000
13.
14. The default interactive shell is now zsh.
15. To update your account to use zsh, please run `chsh -s /bin/
   zsh`.
16. For more details, please visit https://support.apple.com/kb/
   HT208050.
17. (base) Andys-MacBook-Pro:assignment08 andyyeo$ clear
18.
19.
20.
21.
22.
24. (base) Andys-MacBook-Pro:assignment08 andyyeo$ cd desktop/
   assignment08
25. -bash: cd: desktop/assignment08: No such file or directory
26. (base) Andys-MacBook-
   Pro:assignment08 andyyeo$ python Assignment 08.py
27. Menu
28.
29. [1] load Inventory from file
30. [a] Add CD
31. [i] Display Current Inventory
32. [d] delete CD from Inventory
33. [s] Save Inventory to file
34. [x] exit
35.
36. Which operation would you like to perform? [1, a, i, d, s or x]:
```

```
37.
38. Enter ID: 1
39. What is the CD's title? asdfd
40. What is the Artist's name? casd
41. ====== The Current Inventory: ======
42. ID CD Title (by: Artist)
43.
44. 1 Asdfd (by:Casd)
46. Menu
47.
48. [1] load Inventory from file
49. [a] Add CD
50. [i] Display Current Inventory
51. [d] delete CD from Inventory
52. [s] Save Inventory to file
53. [x] exit
55. Which operation would you like to perform? [1, a, i, d, s or x]:
   а
56.
57. Enter ID: 2
58. What is the CD's title? asdfasdg
59. What is the Artist's name? asgdg
60. ====== The Current Inventory: ======
61. ID CD Title (by: Artist)
62.
       Asdfd (by:Casd)
63. 1
64. 2 Asdfasdg (by:Asgdg)
65. ==============
66. Menu
67.
68. [1] load Inventory from file
69. [a] Add CD
70. [i] Display Current Inventory
71. [d] delete CD from Inventory
72. [s] Save Inventory to file
73. [x] exit
75. Which operation would you like to perform? [1, a, i, d, s or x]:
   s
77. ====== The Current Inventory: ======
78. ID CD Title (by: Artist)
79.
80. 1 Asdfd (by:Casd)
```

```
81. 2
      Asdfasdg (by:Asgdg)
83. Save this inventory to file? [y/n] y
84. Menu
85.
86. [1] load Inventory from file
87. [a] Add CD
88. [i] Display Current Inventory
89. [d] delete CD from Inventory
90. [s] Save Inventory to file
91. [x] exit
92.
93. Which operation would you like to perform? [1, a, i, d, s or x]:
   d
94.
95. ====== The Current Inventory: ======
96. ID CD Title (by: Artist)
97.
98. 1
      Asdfd (by:Casd)
99. 2
      Asdfasdg (by:Asgdg)
101.Which ID would you like to delete? 1
102.===== The Current Inventory: ======
103.ID CD Title (by: Artist)
104.
105.2
      Asdfasdg (by:Asgdg)
107.Menu
108.
109.[1] load Inventory from file
110.[a] Add CD
111.[i] Display Current Inventory
112.[d] delete CD from Inventory
113.[s] Save Inventory to file
114.[x] exit
116.Which operation would you like to perform? [1, a, i, d, s or x]:
   d
117.
118.===== The Current Inventory: ======
119.ID CD Title (by: Artist)
120.
121.2
      Asdfasdg (by:Asgdg)
123.Which ID would you like to delete? 2
124.===== The Current Inventory: ======
```

```
125.ID CD Title (by: Artist)
126.
128.Menu
129.
130.[1] load Inventory from file
131.[a] Add CD
132.[i] Display Current Inventory
133.[d] delete CD from Inventory
134.[s] Save Inventory to file
135.[x] exit
136.
137. Which operation would you like to perform? [1, a, i, d, s or x]:
   1
138.
139.WARNING: If you continue, all unsaved data will be lost and the I
   nventory re-loaded from file.
140.type 'yes' to continue and reload from file. otherwise reload wil
   1 be canceledyes
141.reloading...
142.===== The Current Inventory: ======
143.ID CD Title (by: Artist)
144.
145.1
      Asdfd (by:Casd)
146.2 Asdfasdg (by:Asgdg)
148.Menu
149.
150.[1] load Inventory from file
151.[a] Add CD
152.[i] Display Current Inventory
153.[d] delete CD from Inventory
154.[s] Save Inventory to file
155.[x] exit
156.
157. Which operation would you like to perform? [1, a, i, d, s or x]:
158.
159.===== The Current Inventory: ======
160.ID CD Title (by: Artist)
161.
162.1 Asdfd (by:Casd)
163.2
      Asdfasdg (by:Asgdg)
165.Menu
166.
```

```
167.[1] load Inventory from file
168.[a] Add CD
169.[i] Display Current Inventory
170.[d] delete CD from Inventory
171.[s] Save Inventory to file
172.[x] exit
173.
174.Which operation would you like to perform? [l, a, i, d, s or x]:
    x
175.
176.(base) Andys-MacBook-Pro:assignment08 andyyeo$
177.
```