Brent Kieszling

8/13/2020

IT FDN 110: Introduction to Programming (Python)

Module 5 Learning Document

# Introduction

In this paper I will cover the standout points of Module 5. Some of the topics to expect are:unpacking lists, dictionaries and how to access them, the ‘try’ function, and Separation of Concerns. Additionally I will discuss notes on the development of the program I am submitting for Assignment 5.

# Lessons

In previous lessons we learned how to unpack a list by using a ‘for’ loop. Another option to unpack a list that we learned this week is using the star operator ‘\*’. In figure 1 below, the star operator unpacks ‘lstHeader’ on the first iteration and ‘lstCD’ in the second iteration.

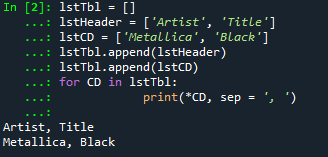


Figure 1 - Using the Star Operator

Next we discussed dictionaries. Dictionaries are mapping types and they provide an alternative option to lists for storing information. Dictionaries are made of two basic parts: a key and a corresponding value.

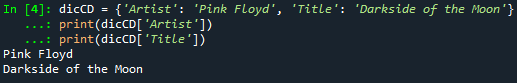


Figure 2 - Example Dictionary

In figure 2 above, the two keys for ‘dicCD’ are ‘Artist’ and ‘Title’ which hold the corresponding values of ‘Pink Floyd’ and ‘Darkside of the Moon’. The print functions return the value held by the key. Many things can be used as a key but they have to be immutable. A list, which is mutable, cannot be a key. Figure 3 shows two examples of modifying dictionaries.

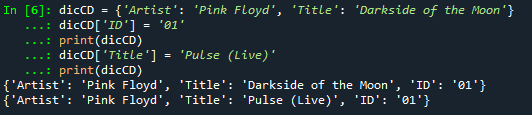


Figure 3- Modifiying Dictionaries

The first ‘print’ statement shows how to add a new key and value to a dictionary. The second shows how to modify a previous value. The difference is that ‘Title’ is already defined as a key word for ‘dicCD’ while ‘ID’ is not. This difference shows that each key in a dictionary must be unique.

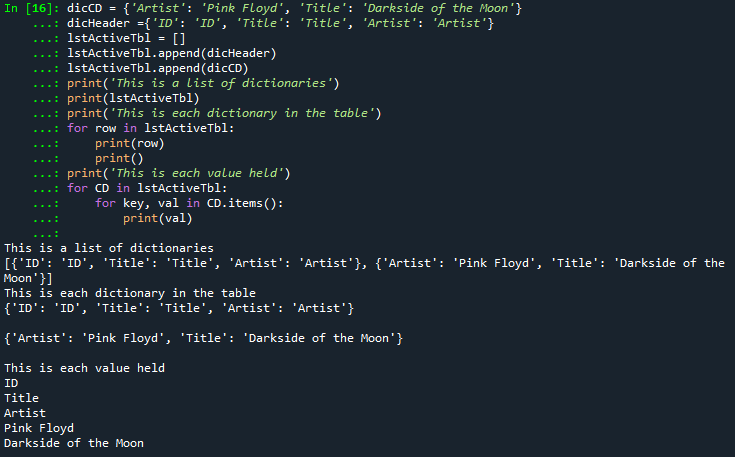


Figure 4 - Accessing Different Levels

Figure 4 shows an example of table holding dictionaries. Each print statement shows how to access information at different levels in the table (full table, the dictionaries, and finally the values). As a new programmer it is useful to recognize the different brackets as type identifiers. ‘[]’ denoting a list and ‘{}’ denoting dictionaries in the example above. Appendix A.2 shows examples on how to move information to or from the program to a text file. Writing to a file is lines 50-61 Reading from a file is lines 63-78. I found that writing the save condition first makes it easier to write the read condition. This is because any formatting applied to how the information is saved needs to be addressed when reading the file back into the program. Figure 5 below shows some of the available operators for dictionaries.

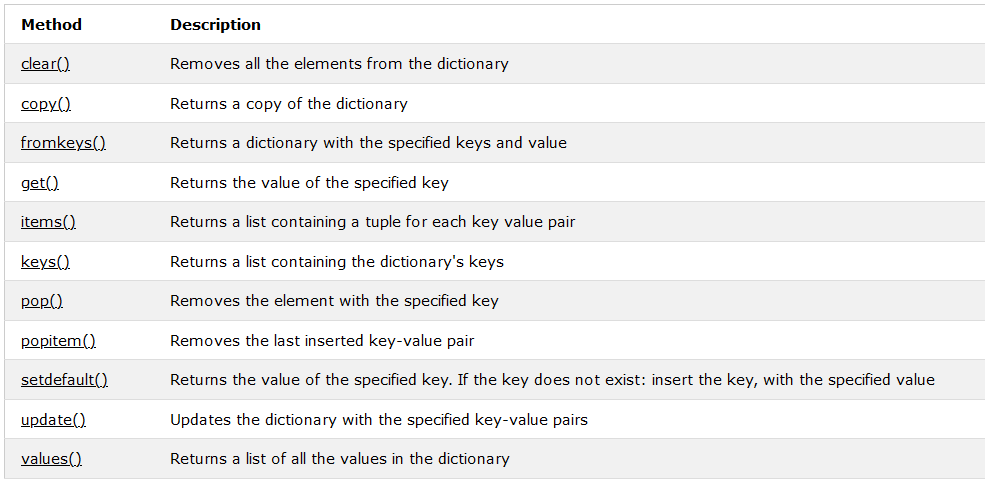


Figure 5 - Dictionary Methods1

Occasionally the user does something not planned for in the program and it causes the program to break. The ‘try’ function can be used handle unexpected user input and return an error message instead of crashing the program. If the line of code in the ‘try’ statement results in an error the program moves to the except statement and executes that instead.

Finally separation of concerns (SOC). SOC Is a way of organizing code structure with the goal of keeping distinct concerns or actions together. Typically coding is broken up into three sections: data, processing, and presentation. Appendix B.1 shows an example of coding utilizing the SOC organizational structure.

# Program: CD Inventory with Dictionaries

We were tasked with creating a program that would enable someone to save a digital record of their CDs, load the record from file, and be able to delete specific saved data. The coding for this can be seen in Appendix B.1. I started this program by creating an active table to hold all of the CD information. Next I established a standard format for the dictionaries. Each dictionary in this program uses the same key structure: ‘ID’, ‘Title’, ‘Artist’. After establishing initial variables, flags, and checking to see if the file already exists, the user enters the menu. The menu is controlled by a ‘while’ loop and has the following options:

* [x] Exit the program
* [l] Load data from saved file
* [a] Add data to the active table
* [i] Display the active table
* [d] Delete and entry
* [s] Save the active table to file

[x] Utilizes the break feature to leave the menu ‘while’ loop and end the program.

[l] To load data, I utilized ‘for’ loop to read each row of the text file. For each row I utilized ‘.strip()’ and ‘.split’ to condition the saved text and split the information in to the standard key structure. Finally I added these values to a dictionary and appended it to the active table.

[a] To add a new CD, I assigned user input the variables for ID, Title, and Artist. Next I added the values to a dictionary based on the appropriate key and added it to the active table. This section also contains two ‘while’ loops to guide the user to load data before adding to the active table. I would have preferred to track the ID of each CD for the user so that duplicates were not generated however, this was in the initial code provided so I decided to leave it in.

[i] To display the active table, I used a ‘for’ loop to move through each dictionary in the active table. For each dictionary I printed the ‘values’ held by each ‘key’. I used a formatting option suggested by Mr. Klos to specify the allowed space for each value being printed. ID is assigned 6 characters while Artist and Title are assigned 22. Spaces are added if not all character slots are utilized.

[d] To delete an option, the user has to enter the corresponding ID. Next, I used a ‘for’ loop to search all the dictionaries in the active table if the ID matched exactly the entry was removed from the active table. Based on my coding it is possible for the user to have multiple CDs that have been assigned the same ID value in which case they would be removed as well.

[s] To save data, I utilized a ‘for’ loop to pull all the values held in a dictionary, separate them by a ‘,’ and then add a new line prior to moving to the next dictionary. The way I chose to save the information overwrites what is currently saved in the text file. I did not want the user to accidently erase their saved information. To address this issue, I created two ‘while’ loops and used a flag to track if the user has saved data this session or not. If the user has not, I give them the option to return to the main menu instead of saving.

# Summary

I was successful in building the CD Inventory program. I enjoyed learning how to create, modify, and update dictionaries. I liked being able to use the ‘key’ words in the dictionary to get the specific value I wanted. Last week I spent a lot of extra time understanding how to read, write, and condition the input read from a text file which paid off for this week’s assignment. The most challenging issue this week was getting ‘break’ to function the way I desired in my code. Initially I had thought ‘break’ ended the current logic. I was incorrectly placing it in an ‘if’ statement with the goal of breaking out at that point and returning to the main menu. In reality ‘break’ only ends the current loop that it is in. Initially my only loop was a ‘while’ loop used for the main menu which is why my program would terminate once it hit ‘break’. Once I understood this, I changed my coding to include an inner while loop so that ‘break’ correctly returned me to the outer loop, the main menu.

# Appendix A.1

1. #------------------------------------------#
2. # Title: Lab05\_A
3. # Desc: Loading data from file to lists
4. # Change Log: (Who, When, What)
5. # DBiesinger, 2030-Jan-01, Created File
6. #Brent Kieszling, 2020-Aug-7, Updated all sections
7. #------------------------------------------#
9. # Declare variabls
11. strChoice = '' # User input
12. lstTbl = []  # list of lists to hold data
13. lstRow = []  # list of data row
14. lstHeader = ['Artist', 'Title']
16. strFileName = 'CDInventory.txt'  # data storage file
17. objFile = None  # file object
19. # Get user Input
20. **print**('Write or Read file data.')
21. **while** True:
22. **print**('\n[a] add data to list\n[w] to write data to file\n[r] to read data from file')
23. **print**('[d] display data\n[exit] to quit')
24. strChoice = input('a, w, r, d, or exit: ').lower()  # convert choice to lower case at time of input
25. **print**('\n\n')
27. **if** strChoice == 'exit':
28. **break**
30. **if** strChoice == 'a':  #no elif necessary, as this code is only reached if strChoice is not 'exit'
31. # Add data to list in memory
32. **if** lstTbl == []:
33. lstTbl.append(lstHeader)
34. **else**:
35. **pass**
36. # TODO ask user to input data and store it in the in-memory list
37. newArtist = input('Enter the CDs Artist: ')
38. newTitle = input('Enter the CDs Title: ')
39. lstRow = [newArtist, newTitle]
40. lstTbl.append(lstRow)
42. **elif** strChoice == 'w':
43. #List to File
44. #TODO add code here to write from in-memory list to file
45. strRow = ''
46. objFile = open(strFileName, 'w')
47. **for** row **in** lstTbl:
48. **for** item **in** row:
49. strRow += str(item) + ','
50. #This deletes the last comma and adds a new line instead
51. strRow = strRow[:-1] + '\n'
52. objFile.write(strRow)
53. objFile.close()

56. **elif** strChoice == 'r':
57. #File to print
58. # TODO read the file line by line into in-memory list.
59. objFile = open(strFileName, 'r')
60. **for** row **in** objFile:
61. lstSavedCDs = row.strip().split(',')
62. **if** lstSavedCDs **in** lstTbl:
63. **pass**
64. **else**:
65. lstTbl.append(lstSavedCDs)
66. objFile.close()


70. **elif** strChoice == 'd':
71. #Display data
72. #TODO display the data to the user.
73. **for** CD **in** lstTbl:
74. **print**(\*CD, sep = ', ')
75. **else**:
76. **print**('Please choose either a, w, r or exit!')

# Appendix A.2

1. #------------------------------------------#
2. # Title: Lab05\_B
3. # Desc: Loading data from file to dictionaries
4. # Change Log: (Who, When, What)
5. # DBiesinger, 2030-Jan-01, Created File
6. #Brent Kieszling, 2020-Aug-7, Updated all sections
7. #------------------------------------------#
9. # Declare variabls
11. strChoice = '' # User input
12. lstTbl = []  # list of lists to hold data
13. #discRow =   # list of data row
14. dicHeader = {'Artist': 'Artist', 'Title': 'Title'}
16. strFileName = 'CDInventoryB.txt'  # data storage file
17. objFile = None  # file object
19. #Check and see if the file exists
20. **import** os
21. **if** os.path.exists(strFileName):
22. **pass**
23. **else**:
24. objFile = open(strFileName, 'a')
25. objFile.close()
26. # Get user Input
27. **print**('Write or Read file data.')
28. **while** True:
29. **print**('\n[a] add data to list\n[w] to write data to file\n[r] to read data from file')
30. **print**('[d] display data\n[exit] to quit')
31. strChoice = input('a, w, r, d, or exit: ').lower()  # convert choice to lower case at time of input
32. **print**('\n\n')
34. **if** strChoice == 'exit':
35. **break**
37. **if** strChoice == 'a':  #no elif necessary, as this code is only reached if strChoice is not 'exit'
38. # Add data to list in memory
39. # Add the header to the active table if not already present
40. **if** lstTbl == []:
41. lstTbl.append(dicHeader)
42. **else**:
43. **pass**
44. # ask user to input data and store it in the in-memory list
45. newArtist = str(input('Enter the CDs Artist: '))
46. newTitle = str(input('Enter the CDs Title: '))
47. dicNewCD = {'Artist': newArtist, 'Title': newTitle}
48. lstTbl.append(dicNewCD)
50. **elif** strChoice == 'w':
51. #List to File
52. # add code here to write from in-memory list to file
53. strRow = ''
54. objFile = open(strFileName, 'w')
55. #This just saves the value held by the key words, this way we avoid saving
56. #something like Artist, Artist and Title,Tile
57. **for** row **in** lstTbl:
58. strArtist = row['Artist']
59. strTitle = row['Title']
60. objFile.write(strArtist + ',' + strTitle + '\n')
61. objFile.close()
63. **elif** strChoice == 'r':
64. #File to print
65. # read the file line by line into in-memory list.
66. objFile = open(strFileName, 'r')
67. **for** row **in** objFile:
68. lstSavedCDs = row.strip().split(',')
69. #Assigns the correct value to the key word based on where it is expected
70. #in the list position, Artist is always first and Title second
71. dicSavedCDs = {'Artist': lstSavedCDs[0], 'Title': lstSavedCDs[1]}
72. #This prevents multiples of the same information from being added to the
73. #active table. Does not stop typos
74. **if** dicSavedCDs **in** lstTbl:
75. **pass**
76. **else**:
77. lstTbl.append(dicSavedCDs)
78. objFile.close()
80. **elif** strChoice == 'd':
81. #Display data
82. #display the data to the user.
83. **for** row **in** lstTbl:
84. #since all of the dictionaries have the same key words 'Artist' and 'Title'
85. # We can just print the value held at the key word as we go through each
86. #dictionary instead of nested 'for' loops.
87. **print**(row['Artist'] + ', ' + row['Title'])
89. **else**:
90. **print**('Please choose either a, w, r or exit!')

# Appendix B.1

1. #------------------------------------------#
2. # Title: CDInventory.py
3. # Desc: Starter Script for Assignment 05
4. # Change Log: (Who, When, What)
5. # DBiesinger, 2030-Jan-01, Created File
6. #Brent Kieszling, 2020-August-10, Modified all sections
7. #Brent Kieszling, 2020-August-11, Fixed option [a] and added SOC formatting
8. #------------------------------------------#
10. #DATA--
11. strChoice = '' # User input
12. lstActiveTbl = []  # list of dictionaries to hold data
13. dicHeader ={'ID': 'ID', 'Title': 'Title', 'Artist': 'Artist'}
14. dicRow = {'ID': '', 'Title': '', 'Artist': ''}
15. strFileName = 'CDInventory.txt'  # data storage file
16. objFile = None  # file object
17. haveLoaded = False # Load tracker
18. strMainMenu = 'Please select **from** the following options: \n \
19. [l] load Inventory **from** file\n \
20. [a] Add CD\n \
21. [i] Display Current Inventory \n \
22. [d] delete CD **from** Inventory\n \
23. [s] Save Inventory to file\n \
24. [x] exit \n'
26. strNotLoadedA = 'Adding a new entry before loading saved data may \
27. result **in** duplicate CD IDs. Would you like to **return** to the main menu? (y/n) \n'
29. strNotLoadedS = 'Saving before loading will overwrite saved data.\n\
30. Would you like to **continue**? (y/n) \n'
31. #PROCESSING--
32. #Check and see if the file is present, if not create the file
33. **import** os
34. **if** os.path.exists(strFileName):
35. **pass**
36. **else**:
37. objFile = open(strFileName, 'a')
38. objFile.close()
40. # Get user Input
41. #PRESENTATION INPUT/OUTPUT--
42. **print**('Welcome to: The Magic CD Inventory!\n')
43. **while** True:
44. # 1. Display menu allowing the user to choose:
45. # convert choice to lower case at time of input
46. strChoice = str(input(strMainMenu)).lower()
47. **print**()

50. #----------------Menu option [x]-----------------------------------------
51. # Exit the program if the user chooses so
52. #PROCESSING--
53. **if** strChoice == 'x':
54. **break**

57. #----------------Menu option [l]-----------------------------------------
58. # Load data from saved file
59. #PROCESSING--
60. **elif** strChoice == 'l':
61. haveLoaded = True
62. objFile = open(strFileName, 'r')
63. **for** row **in** objFile:
64. lstSavedCD = row.strip().split(',')
65. # All ID's are intergers except the header ID which is a string.
66. # This try statement navigates the difference in type while adding saved data.
67. **try**:
68. lstSavedCD[0] = int(lstSavedCD[0])
69. **except**:
70. **pass**
71. dicRow = {'ID' : lstSavedCD[0], 'Title': lstSavedCD[1], 'Artist': lstSavedCD[2]}
72. **if** dicRow **in** lstActiveTbl:
73. **pass**
74. **else**:
75. lstActiveTbl.append(dicRow)
76. objFile.close()
78. #PRESENTATION OUTPUT--
79. **print**('Saved data has been uploaded.')

82. #----------------Menu option [a]-----------------------------------------
83. # Add data to the table (2d-list) each time the user wants to add data
84. #PROCESSING--
85. **elif** strChoice == 'a':
86. # Add header to active table if not already present
87. **if** dicHeader **in** lstActiveTbl:
88. **pass**
89. **else**:
90. lstActiveTbl.append(dicHeader)
92. # Check if saved data has been loaded this session
93. **while** haveLoaded != True:
94. #PRESENTATION INPUT/OUTPUT--
95. x = str(input(strNotLoadedA)).lower()
96. #PROCESSING--
97. **if** x == 'y':
98. **break**
99. **else**:
100. #PRESENTATION INPUT/OUTPUT--
101. strID = input('Enter a numerical ID: ')
102. strTitle = input('Enter the CD\'s Title: ')
103. strArtist = input('Enter the Artist\'s Name: ')
104. #PROCESSING--
105. intID = int(strID)
106. dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}
107. lstActiveTbl.append(dicRow)
108. **break**
110. **while** haveLoaded == True:
111. #PRESENTATION INPUT/OUTPUT--
112. strID = input('Enter an ID: ')
113. strTitle = input('Enter the CD\'s Title: ')
114. strArtist = input('Enter the Artist\'s Name: ')
115. intID = int(strID)
116. #PROCESSING--
117. dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}
118. lstActiveTbl.append(dicRow)
119. **break**

122. #----------------Menu option [i]-----------------------------------------
123. # Display the current data to the user each time the user wants to display the data
124. #PROCESSING--
125. **elif** strChoice == 'i':
126. # All of the dictionaries have the same key words 'ID', 'Title' and 'Artist'
127. # Therefore we can print the value held at the key word as we go through each dictionary.
128. #PRESENTATION OUTPUT--
129. **for** row **in** lstActiveTbl:
130. # Formats the display into an easily readable table. ID is padded with 6 available
131. #characters while Artist and Title are padded with 22. Recommended by Mr Klos.
132. **print**('{:6}{:22}{:22}'.format(str(row['ID']), row['Title'], row['Artist']))

135. #----------------Menu option [d]-----------------------------------------
136. #Delete an entry
137. #PROCESSING--
138. **elif** strChoice == 'd':
139. #PRESENTATION INPUT/OUTPUT--
140. cdDelete = int(input('Please enter the ID of the CD you would like to remove. \n'))
141. #PROCESSING--
142. **for** row **in** lstActiveTbl:
143. #checkID is set to the value held by ID in each dictionary and then checks
144. #if it matches the users input.
145. checkID = row.get('ID')
146. **if** checkID == cdDelete:
147. lstActiveTbl.remove(row)
148. #PRESENTATION INPUT/OUTPUT--
149. **print**('The following CD has been removed: \n')
150. **print**(row)
151. #PROCESSING--
152. **else**:
153. **pass**

156. #----------------Menu option [s]-----------------------------------------
157. # Save the data to a text file CDInventory.txt if the user chooses so
158. #PROCESSING--
159. **elif** strChoice == 's':
160. **while** haveLoaded != True:
161. #PRESENTATION INPUT/OUTPUT--
162. z = str(input(strNotLoadedS)).lower()
163. #PROCESSING--
164. **if** z == 'y':
165. objFile = open(strFileName, 'w')
166. **for** row **in** lstActiveTbl:
167. strID = str(row['ID'])
168. strTitle = row['Title']
169. strArtist = row['Artist']
170. objFile.write(strID + ',' + strTitle + ',' + strArtist + '\n')
171. objFile.close()
172. **print**('Save succesful!')
173. **break**
174. **else**:
175. **print**('Failed to save.')
176. **break**
178. **while** haveLoaded == True:
179. #PRESENTATION INPUT/OUTPUT--
180. objFile = open(strFileName, 'w')
181. **for** row **in** lstActiveTbl:
182. strID = str(row['ID'])
183. strTitle = row['Title']
184. strArtist = row['Artist']
185. objFile.write(strID + ',' + strTitle + ',' + strArtist + '\n')
186. objFile.close()
187. **print**('Save succesful!')
188. **break**

191. # Handle unexpected menu input
192. #PRESENTATION OUTPUT--
193. **else**:
194. **print**('Please choose from the following letters: l, a, i, d, s or x')

# Appendix B.2

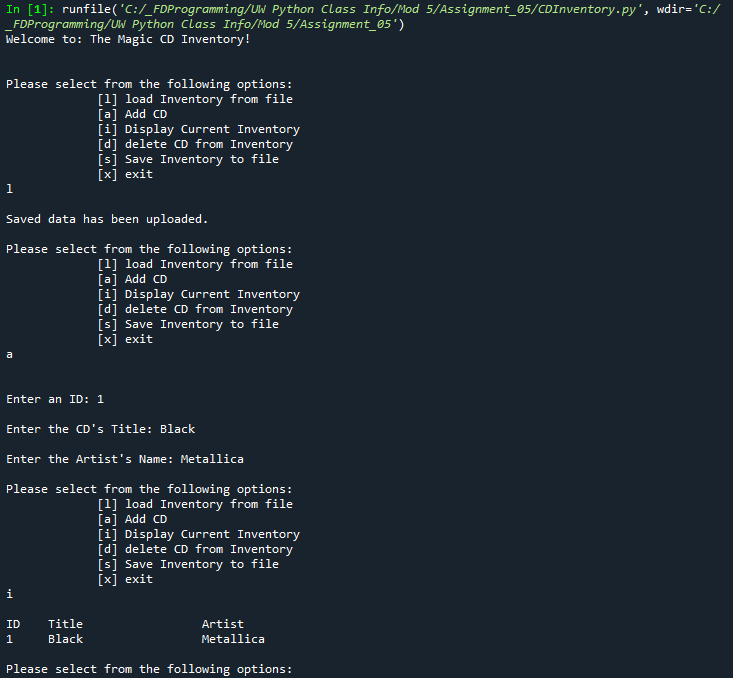


Figure 6 - Program run in Spyder part 1/2



Figure 7 - Program run in Spyder part 2/2

# Appendix B.3

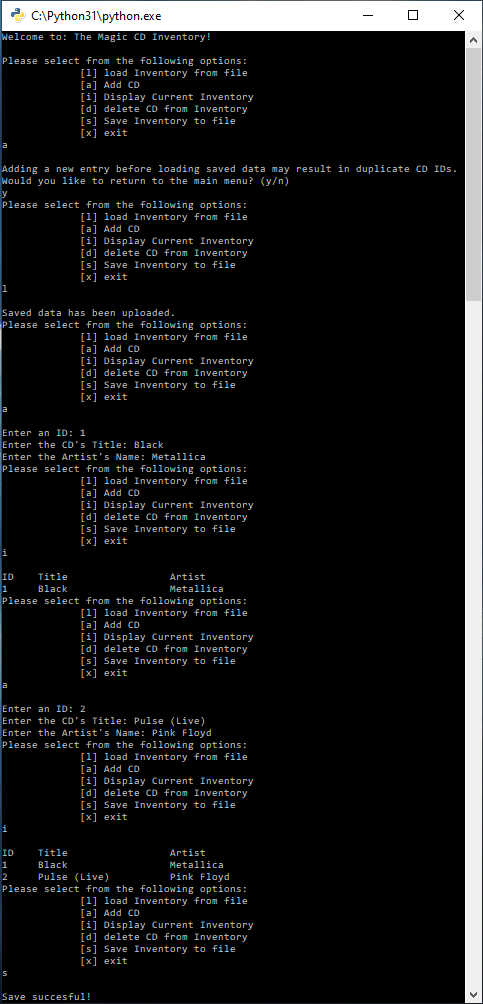


Figure – Program run in Terminal 1/2

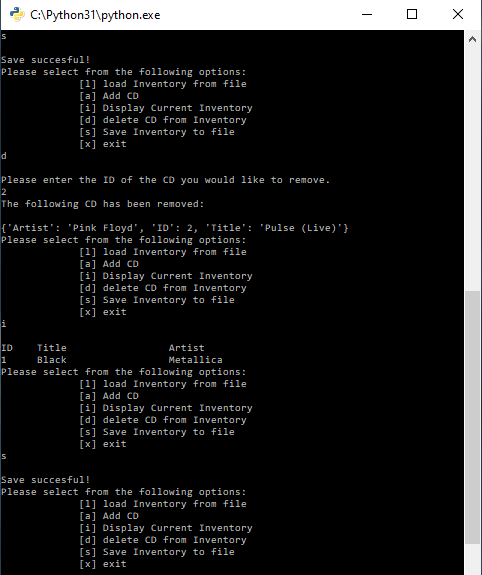


Figure - Program run in Terminal 2/2

# References

1. https://www.w3schools.com/python/python\_dictionaries.asp Accessed: 8/12/2020