Kenny Lu

2020-Feb-21

IT FDN 100 B Winter 2020

Assignment05

GitHub <https://github.com/lukenny/Assignment_05>

# Introduction

In this Fifth module, we cover dictionaries, list, how to read data from a file into a list and a dictionary. Also, we cover “Separation of Concerns” and 2-dimensional way to organize data. This module is particularly hands-on; pretty much all topics are applicable to the homework assignment.

Topic 1 – dictionary vs. list

List and dictionary are essentially different data structures. A list stores a sequence of objects in a certain order such that you can index into the list, or iterate over the list. Per our last module, we know list is mutable after they have been created. Python dictionary is an implementation of a hash table and is a key-value store. It is not ordered it is fast for lookups by key; hence, it’s more performance oriented. If that’s the case, why don’t we just use dictionary all the time, one of the drawbacks of dictionary is it uses a lot of memory. Here are some characteristics and we can do a better comparison with some examples. Objects in a list maintain their ordering unless explicitly re-ordered. They can be of different types, and types can be mixed. They can be accessed and referenced via numeric indices. Each object in a dictionary has a key and a value. Ordering is not necessary elements are accessed and referenced via key values. Key values can be of any hash table type. Values can be of any type and types can be mixed. Here’s a quick review of list like we did with module 4.

1. In [13]: teams = ['Sounders', 'Timbers', 'Galaxy', 'Whitecaps']
3. In [14]: **print**(teams[2])
4. Galaxy
6. In [15]: teams.append('Dynamo')
8. In [16]: **print**(teams)
9. ['Sounders', 'Timbers', 'Galaxy', 'Whitecaps', 'Dynamo']
11. In [17]: **del** (teams[4])
13. In [18]: **print**(teams)
14. ['Sounders', 'Timbers', 'Galaxy', 'Whitecaps']

Listing 1 – sample code to demonstrate accessing, appending and deleting an element in a list

On the other hand, dictionaries are also known as an associative array and it consists of a collection of key-value pairs. Each key-value pair maps the key to its associated value. For instance, in the example below each city has the name of the team as value. Regardless of their position in the array, they can be accessed by the key name instead of indices and additional value can only be appended with its associated key.

1. In [19]: MLS\_team = {'Seattle':'Sounders', 'Portland':'Timbers', 'Los Angeles':'Galaxy', 'Vancouver':'Whitecaps'}
3. In [20]: MLS\_team['Seattle']
4. Out[20]: 'Sounders'
6. In [21]: MLS\_team['Vancouver']
7. Out[21]: 'Whitecaps'
9. In [22]: MLS\_team['New England'] = 'Revolution'
11. In [23]: MLS\_team
12. Out[23]:
13. {'Seattle': 'Sounders',
14. 'Portland': 'Timbers',
15. 'Los Angeles': 'Galaxy',
16. 'Vancouver': 'Whitecaps',
17. 'New England': 'Revolution'}
19. In [24]: **del** MLS\_team['New England']
21. In [25]: MLS\_team
22. Out[25]:
23. {'Seattle': 'Sounders',
24. 'Portland': 'Timbers',
25. 'Los Angeles': 'Galaxy',
26. 'Vancouver': 'Whitecaps'}

Listing 2 – sample code to demonstrate accessing, appending and deleting an element in a dictionary

Using [net-information](http://net-informations.com/python/iq/dict.htm) (external reference) webpage

Using [stackoverflow](https://stackoverflow.com/questions/3489071/in-python-when-to-use-a-dictionary-list-or-set) (external reference) webpage

Using [realpython](https://realpython.com/python-dicts/) (external reference) webpage

# Topic 2 – Separation of Concerns

Separation of Concerns is a computer science design principle where program should be written and separated into distinct sections such that each section addresses a distinct concern. In this context, concern is a set of information that associated with the code of the program. We can use assignment #2 as an example to go over data, processing and presentation.

1. # -- DATA -- #
2. FirstNum = None
3. SecondNum = None
5. # -- PRESENTATION -- #
6. # Get User input data
7. **print**("Basic Math script. Calculating the Sum, Difference, Product and Quotient of two numbers.")
8. FirstNum = input("Please enter the 1st number : ")
9. SecondNum = input("Please enter the 2nd number : ")
11. # -- PROCESSING -- #
12. # Process the data
13. Sum = int(FirstNum) + int(SecondNum);
14. Product = int(FirstNum) \* int(SecondNum);
15. Difference = int(FirstNum) - int(SecondNum);
16. Quotient = int(FirstNum) / int(SecondNum);
18. # -- PRESENTATION -- #
19. # Display the data
20. **print**("\nThe script calculated using the Numbers {} and {}\nThe Results are:\n".format(FirstNum, SecondNum))
21. **print**("Sum: \t\t" , Sum)
22. **print**("Difference: \t" , Difference)
23. **print**("Product: \t" , Product)
24. **print**("Quotient: \t" , Quotient)

Listing 4 – SoC demo with assignment #2

Using [Wikipedia](https://en.wikipedia.org/wiki/Separation_of_concerns) (external reference) webpage

# Summary

This module covers both technical topics plus some formatting and logistics best practices of programming, for instance ‘Separation of Concerns’ and ‘GitHub’. I feel like I get all the concepts with this module and the assignment itself seems much more straight-forward. I struggled a bit with the last assignment as I was a bit hung up with the logic but this one is more straight-forward as the specific asks are spelled out explicitly, ‘add function to load data’, ‘add function to delete an entry’ and ‘use dictionary instead of list’. For me, comprehending the pros and cons of ‘dictionary’ is definitely crucial and worthwhile because at work it’s widely adopted in our code base.

# Appendix

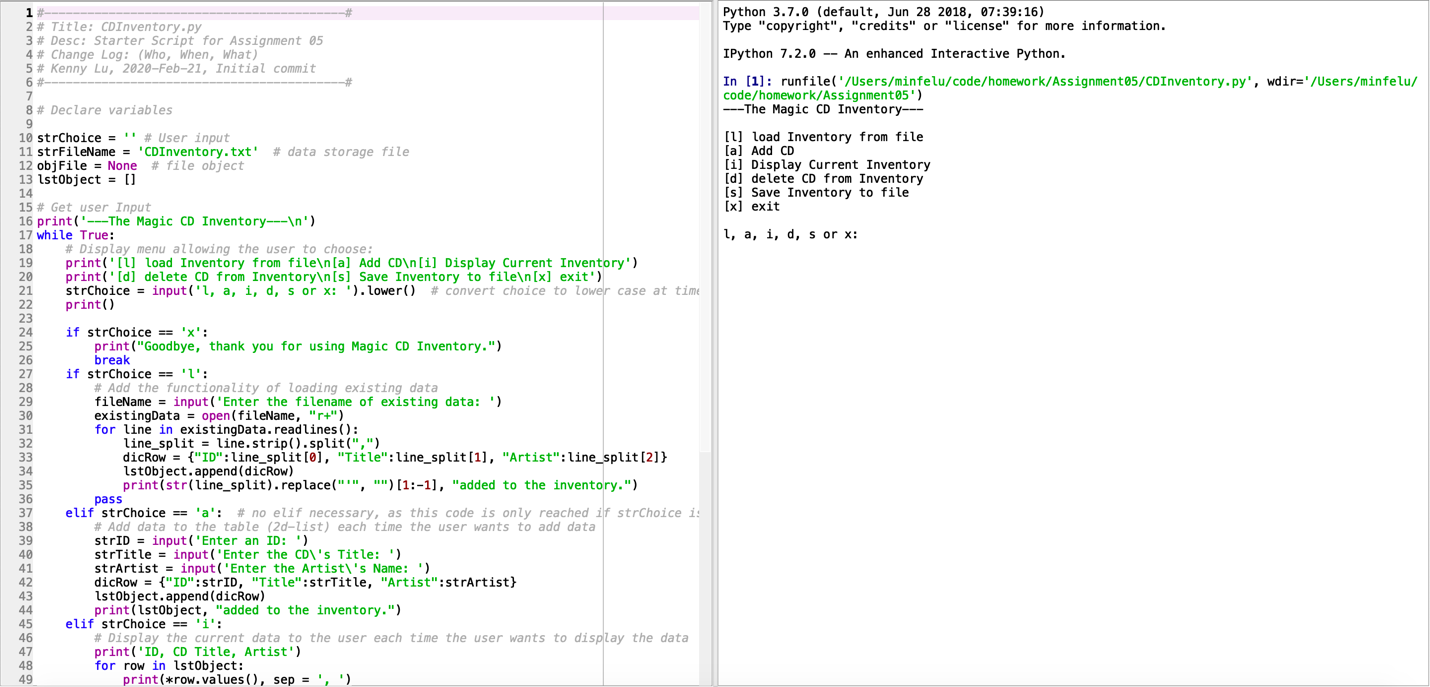


Figure 1 – Example output from script initial loaded in Spyder

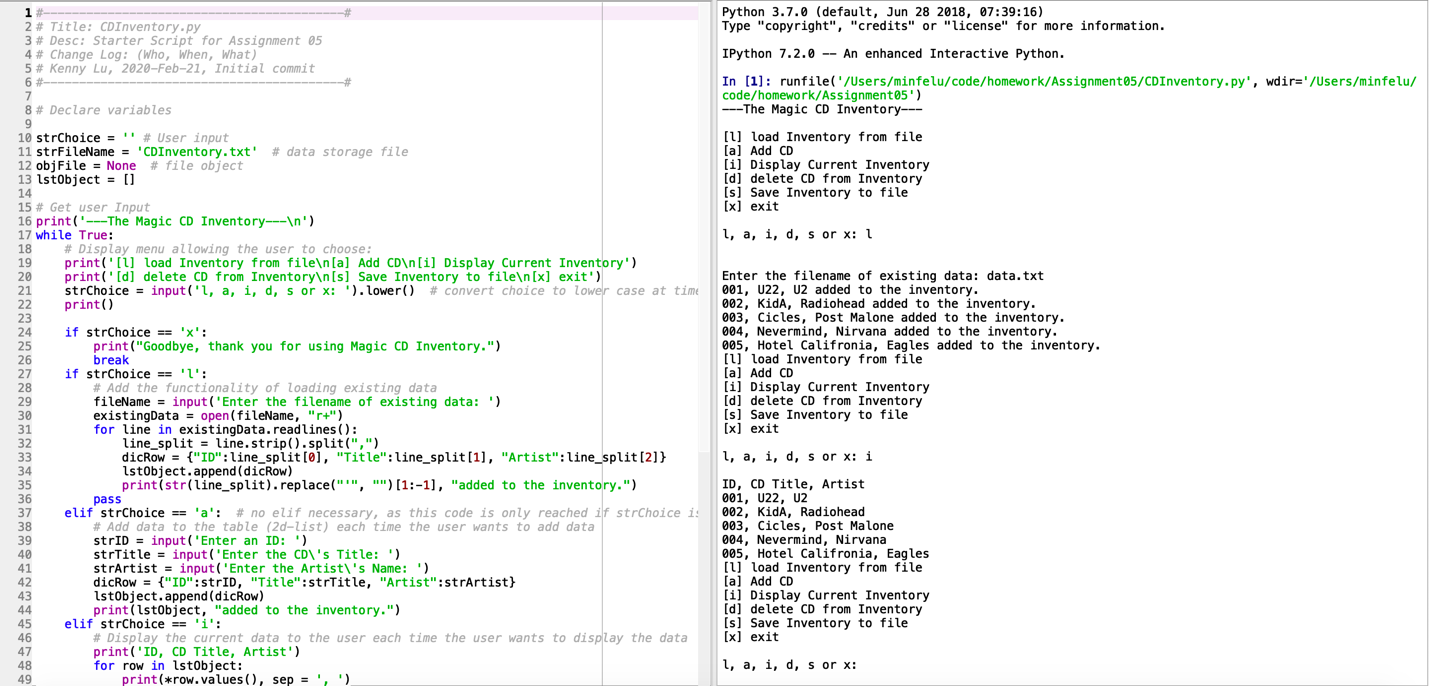


Figure 2 – Example output from script load data and display inventory

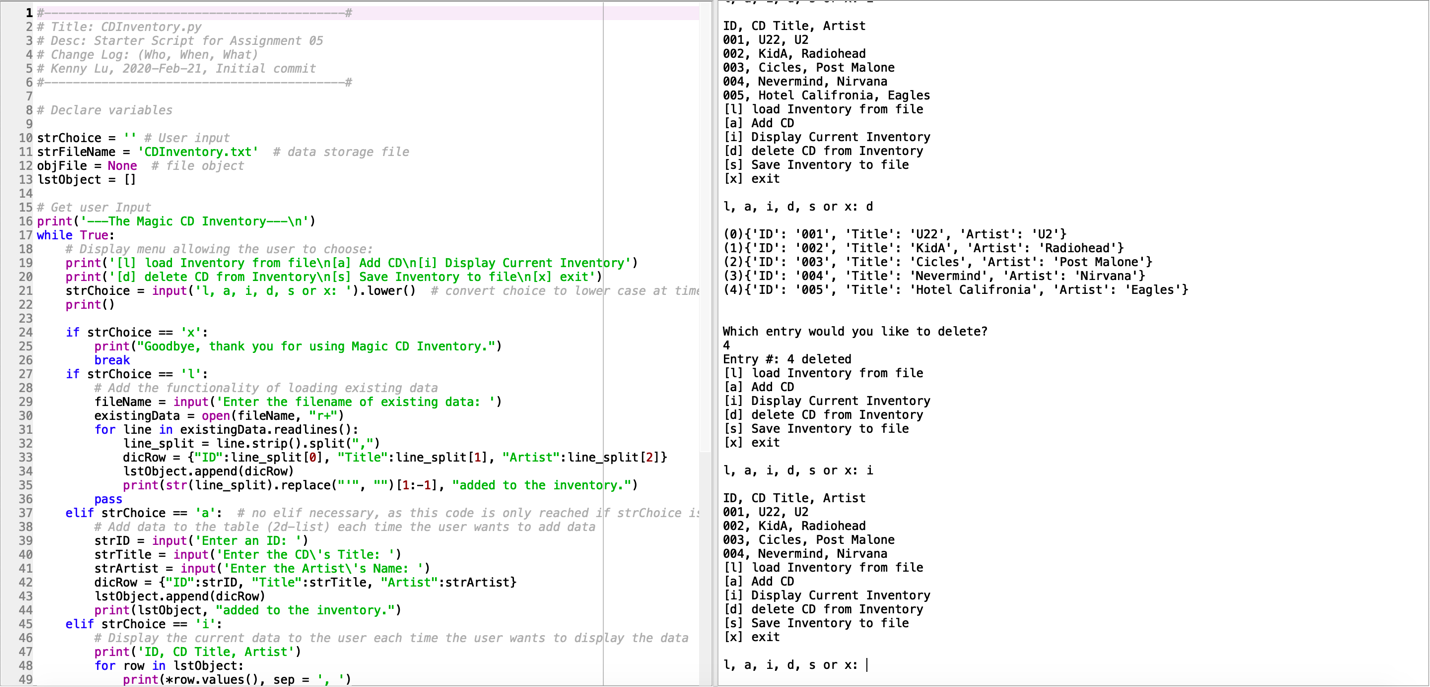


Figure 3 – Example output from script delete entry and display inventory

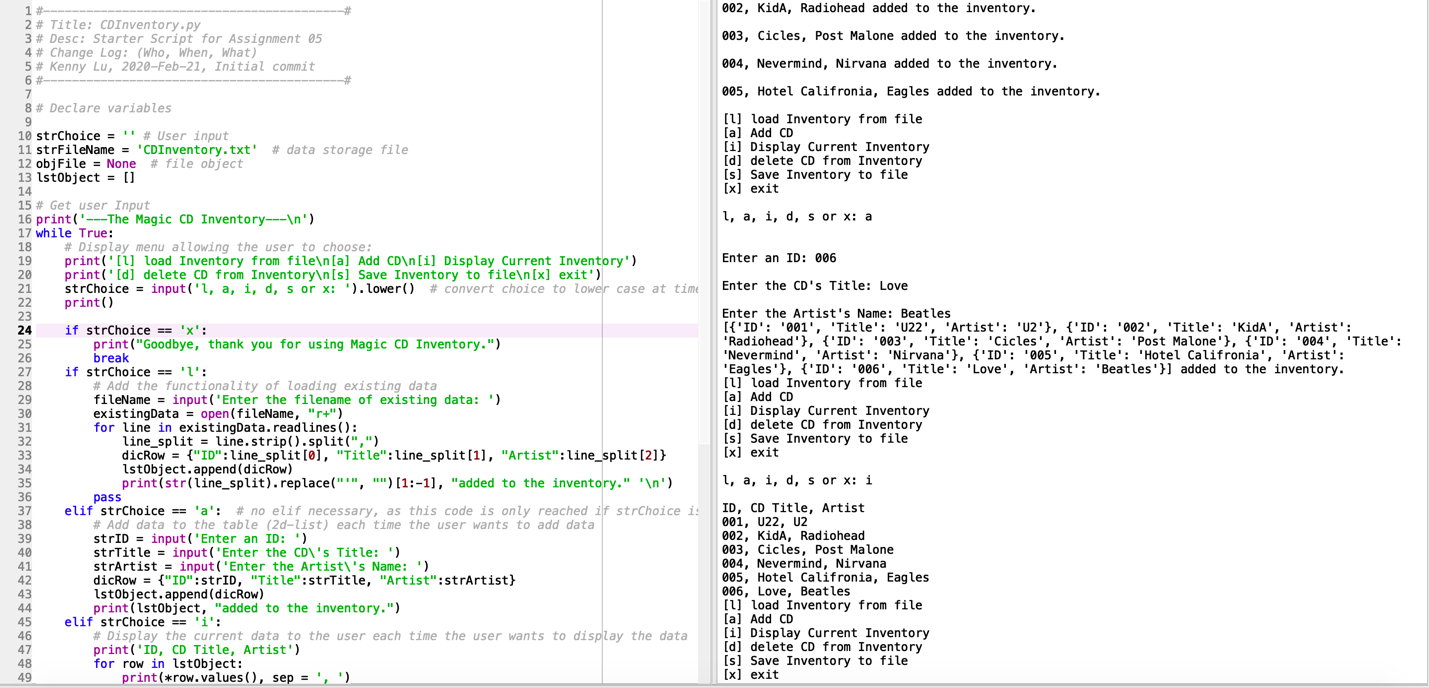


Figure 4 – Example output from script add entry and display inventory

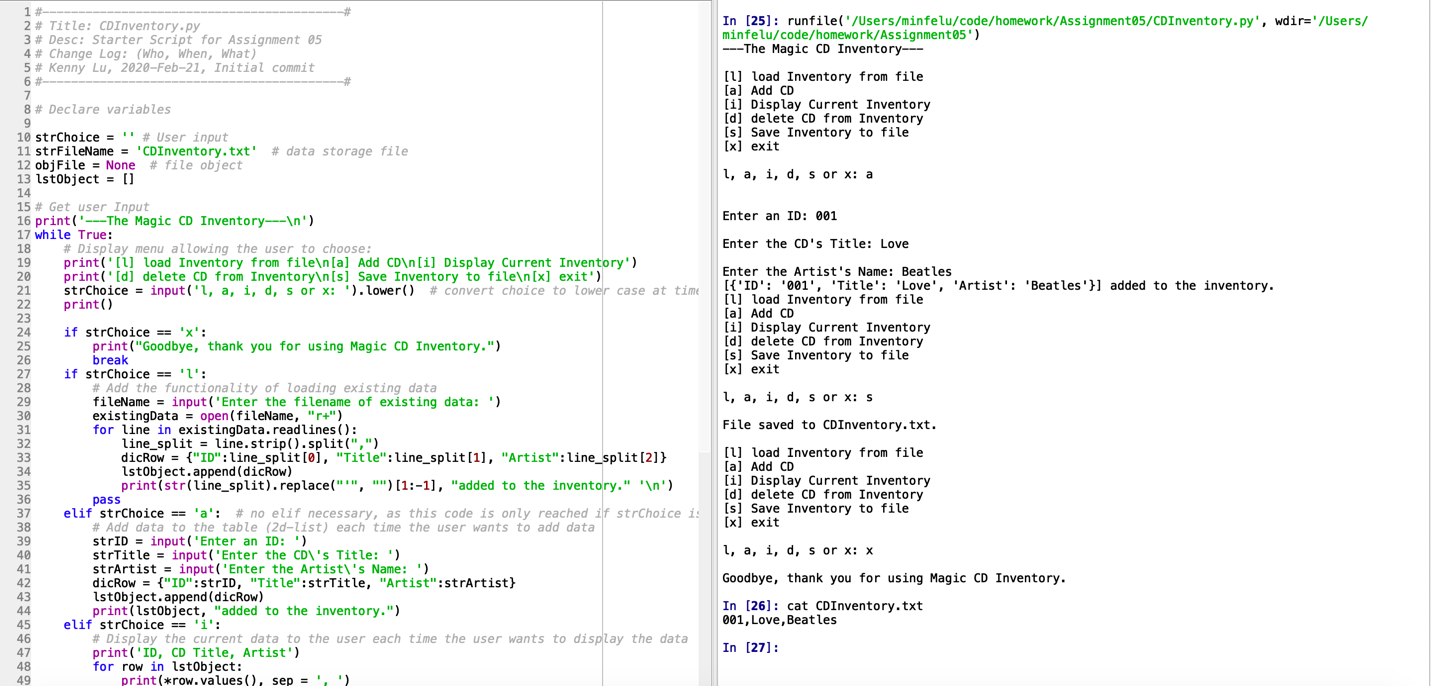


Figure 5 – Example output from script save data to CDInventory.txt and exit

1. #------------------------------------------#
2. # Title: CDInventory.py
3. # Desc: Starter Script for Assignment 05
4. # Change Log: (Who, When, What)
5. # Kenny Lu, 2020-Feb-21, Initial commit
6. #------------------------------------------#
8. # Declare variables
10. strChoice = '' # User input
11. strFileName = 'CDInventory.txt'  # data storage file
12. objFile = None  # file object
13. lstObject = []
15. # Get user Input
16. **print**('---The Magic CD Inventory---\n')
17. **while** True:
18. # Display menu allowing the user to choose:
19. **print**('[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
20. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit')
21. strChoice = input('l, a, i, d, s or x: ').lower()  # convert choice to lower case at time of input
22. **print**()
24. **if** strChoice == 'x':
25. **print**("Goodbye, thank you for using Magic CD Inventory.")
26. **break**
27. **if** strChoice == 'l':
28. # Add the functionality of loading existing data
29. fileName = input('Enter the filename of existing data: ')
30. existingData = open(fileName, "r+")
31. **for** line **in** existingData.readlines():
32. line\_split = line.strip().split(",")
33. dicRow = {"ID":line\_split[0], "Title":line\_split[1], "Artist":line\_split[2]}
34. lstObject.append(dicRow)
35. **print**(str(line\_split).replace("'", "")[1:-1], "added to the inventory." '\n')
36. **pass**
37. **elif** strChoice == 'a':  # no elif necessary, as this code is only reached if strChoice is not 'exit'
38. # Add data to the table (2d-list) each time the user wants to add data
39. strID = input('Enter an ID: ')
40. strTitle = input('Enter the CD\'s Title: ')
41. strArtist = input('Enter the Artist\'s Name: ')
42. dicRow = {"ID":strID, "Title":strTitle, "Artist":strArtist}
43. lstObject.append(dicRow)
44. **print**(lstObject, "added to the inventory.")
45. **elif** strChoice == 'i':
46. # Display the current data to the user each time the user wants to display the data
47. **print**('ID, CD Title, Artist')
48. **for** row **in** lstObject:
49. **print**(\*row.values(), sep = ', ')
50. **elif** strChoice == 'd':
51. # Add functionality of deleting an entry
52. **if** len(lstObject) != 0:
53. count = 0
54. **for** i **in** lstObject:
55. **print**("(",count,")", i, sep='')
56. count += 1
57. deleteEntry = int(input('\n' + "Which entry would you like to delete? " '\n'))
58. lstObject.pop(deleteEntry)
59. **print**("Entry #:", deleteEntry, "deleted")
60. **pass**
61. **else**:
62. **print**('Sorry nothing to delete empty inventory.')
63. **break**
64. **elif** strChoice == 's':
65. # Save the data to a text file CDInventory.txt if the user chooses so
66. objFile = open(strFileName, 'w')
67. **for** row **in** lstObject:
68. objFile.write(str(row['ID'] + "," + row['Title'] + "," + row['Artist'] + '\n'))
69. objFile.close()
70. **print**("File saved to CDInventory.txt.")
71. **break**
72. **else**:
73. **print**('Please choose either l, a, i, d, s or x!')

Figure 6 – CDInventory.py