Dan Oranski

2/24/20

IT FDN 100 B

Assignment 05

Observations of Assignment 05

# Introduction

This document covers the work through of Assignment 05. This assignment was very much a continuation of the learning in Assignment 04. In this assignment, students applied the concepts of dictionaries to the CD inventory program created in the previous module. Programming best practices were discussed and implemented. The concepts were unified by modifying the script from Assignment 04 to utilize dictionaries in place of lists.

# Dictionaries

Dictionaries are a useful data type in Python, and are referred to as “associative memories” or “associative arrays” in other programming languages. Unlike sequences, which are indexed by a range of numbers, dictionaries are indexed by keys. Dictionary keys can be any immutable type and strings and numbers can always be keys. Tuples can be used as keys, but only if they contain only strings, numbers or other tuples.

Dictionaries are best thought of as set of key:value pairs. The keys within must be unique. Inputting and extracting data to and from dictionaries is done by calling the key. It is convenient to populate dictionaries with for loops. Dictionaries can also be manipulated manually. Here are some options for doing so:

list(d) - Returns a list of all the keys used in the dictionary *d*.

len(d) - Returns the number of items in the dictionary *d*.

d[key] - Returns the item of d with key *key*. Raises a KeyError if *key* is not in the map.

Set d[key] to value. - *key* in *d*: Return True if *d* has a key *key*, else False. *key* not in *d*: Equivalent to not key in d.

clear() - Remove all items from the dictionary.

items() - Return a new view of the dictionary’s items ((key, value) pairs).

keys() - Return a new view of the dictionary’s keys.

values() - Return a new view of the dictionary’s values.[[1]](#footnote-1)

LAB 05-B directed students to convert the row objects in LAB 05-A from lists to dictionaries. This required modifying how the data was called and referenced with key:value pairs vs indices. The images on the next page highlight the changes between the two scripts

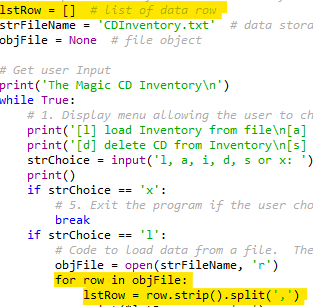


Figure 1 - LAB 05-A List

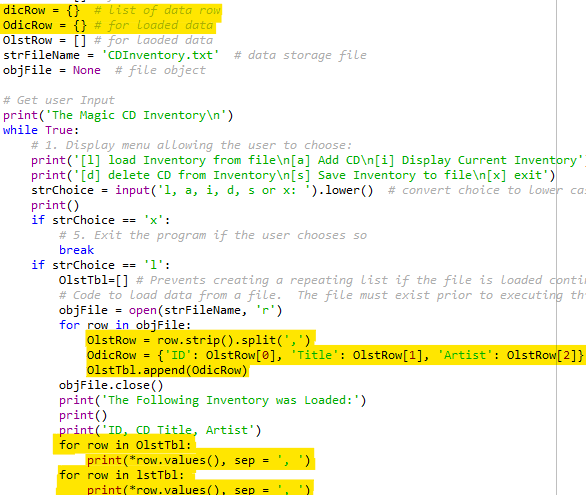


Figure 2 - LAB 05-B Dictionary

# Improving Scripts

As the amount of code grows within a script, it becomes critical that the contents is well documented and organized. This is important because it is unlikely that the same person will be establishing and continually improving the code. In order for several programmers to work on the same set of code, it is necessary that each of the programmers can jump in and get to work immediately. This requires good programming practices.

## Separation of Concerns

Separation of Concerns (SoC), also called programming pattern, is a way to break apart a program to better organize the logical flow. This principle separates a computer program into distinct sections, addressing a different concern with each section. A concern can be thought of as information that affects the code of a computer program.

For the most part, computer programs can be broken apart into three sections: Data, Processing and Input-Output. Students were instructed to insert these sections into their code going forward in the course.[[2]](#footnote-2)

## Functions

Functions allow programmers to group statements and execute the statements based on calling an associated given name. When using Python, functions must be defined before they are called. Declaring a function simply brings it into memory, but calling it within the code is what actually executes the function.

## Script Templates

Script templates allow programmers to quickly populate a repeatable layout for writing code. Script templates can include headers, sections and repeatable objects. Students were shown how to set a template when utilizing the Spyder IDE.

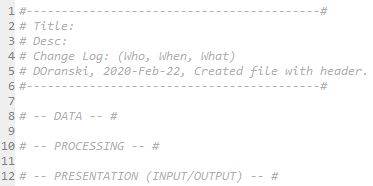


Figure 3 - Script template

## Structured Error Handling

When encountering an error in a Python script, if structured error handling is not present in the script, the program will crash and a generic Python error message will output. This is less than desirable from a usage perspective since the user will have to restart the program and could lose or corrupt their already existing data.

Including error handling via the Try-Except construct allows programmers to provide a means to reset the progress of the script, display a message regarding what went wrong, and prevent crashing the program.[[3]](#footnote-3)

# Application of Learning Objectives

Tying everything together, the student was asked to modify the Assignment 04 Python script while adding in functionality to read from a file and delete entries in the library. The script was created, troubleshot, and run in the Spyder IDE.

For this assignment, I attempted to make the program as functional as possible. As part of this, I wrote the script to renumber the entries each time the file is loaded. This was necessary to keep a reasonable numbering system in place after the deletion functionality was introduced. The logic works as follows: delete entries, save the file, and load the file back into memory. The saved data will have the original numbering, the loaded data will have the new numbering. I also added functionality to confirm both the delete and save commands in the event that the user makes an irreversible mistake. The program will create a new file if one doesn’t exist and will modify an already existing file.

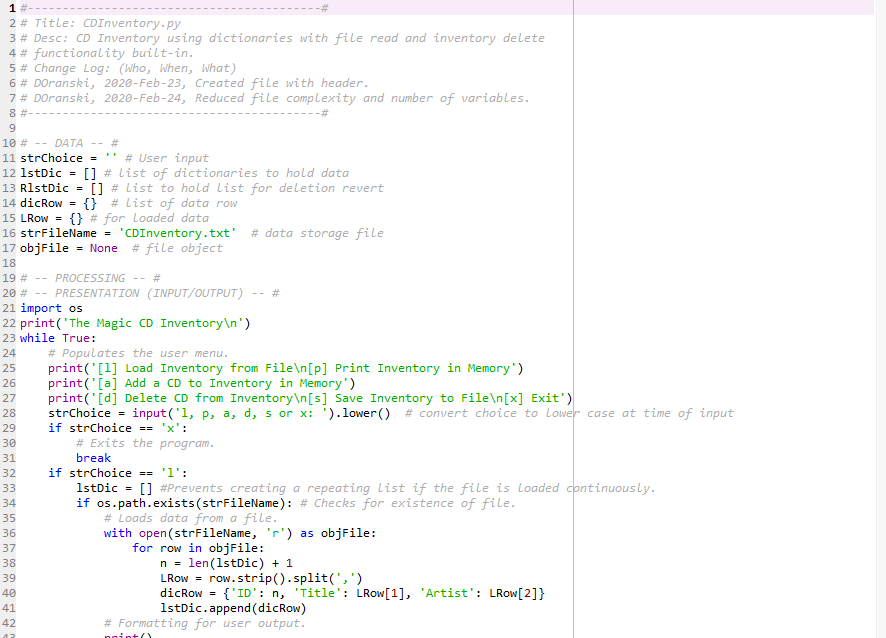


Figure 4 - Assignment05 script

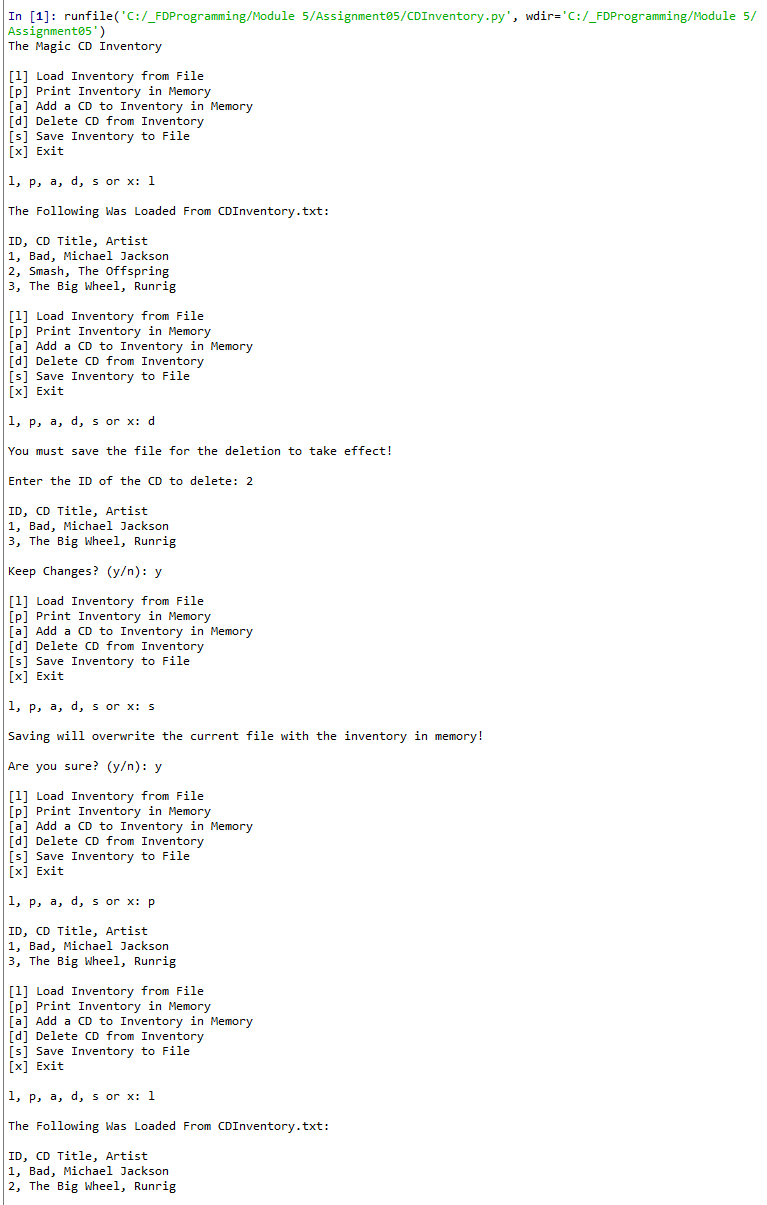


Figure 5 - CDInventory.py execution

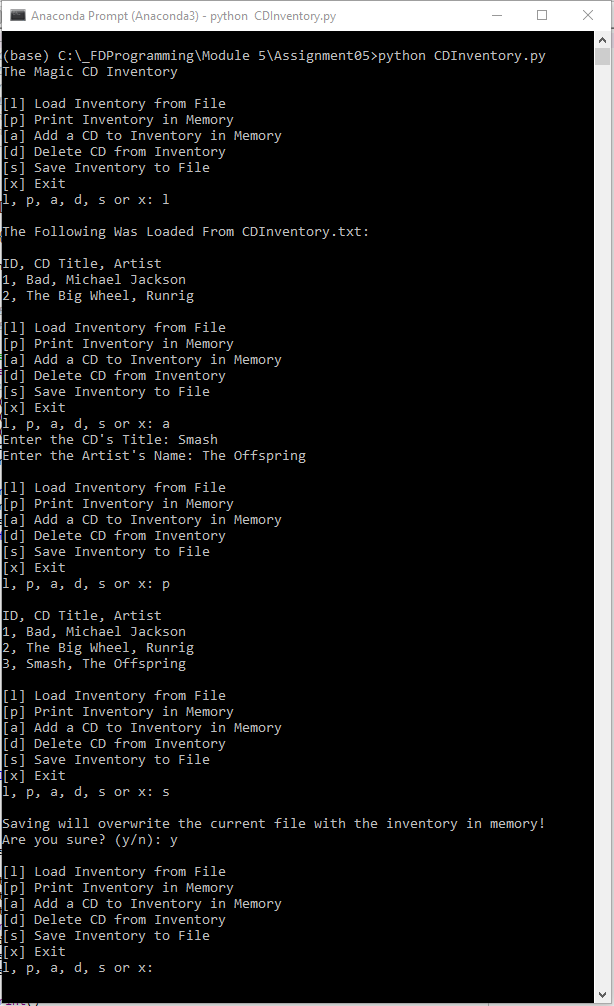


Figure 6 - Terminal execution

The files for this Assignment were uploaded to GitHub [here](https://github.com/angryeng/Assignment_05).

# Summary

Assignment 05 was an excellent opportunity to revisit the learning of objectives of Module 4, and apply some additional concepts. Dictionaries are a useful tool that in my opinion are a better data storage object for this task than a list. The labs were focused on understanding the differences between lists and dictionaries. Coupling the learning from Module 4, loops were appropriately applied to dictionaries. This lesson truly solidified many of the concepts learned earlier in the course.

1. <https://docs.python.org/3/tutorial/datastructures.html>, retrieved 2020-Feb-23 [↑](#footnote-ref-1)
2. <https://medium.com/machine-words/separation-of-concerns-1d735b703a60>, retrieved on 2020-Feb-23 [↑](#footnote-ref-2)
3. <https://docs.python.org/3/tutorial/errors.html>, retrieved on 2020-Feb-22 [↑](#footnote-ref-3)