John Stevens

March 15, 2020

IT FDN 100 B

Assignment 07

CD Inventory(w/pickle and structured

error handling) Script

# Introduction

This document will show the modification of a pre coded script in Python to use pickle to load an existing binary file or create a new one, ask a user to input an inventory of CDs, display the inventory, delete any inputted CDs and then save the inventory list to a binary file using pickle. The script also uses structured error handling to capture error to provide a more stable environment.

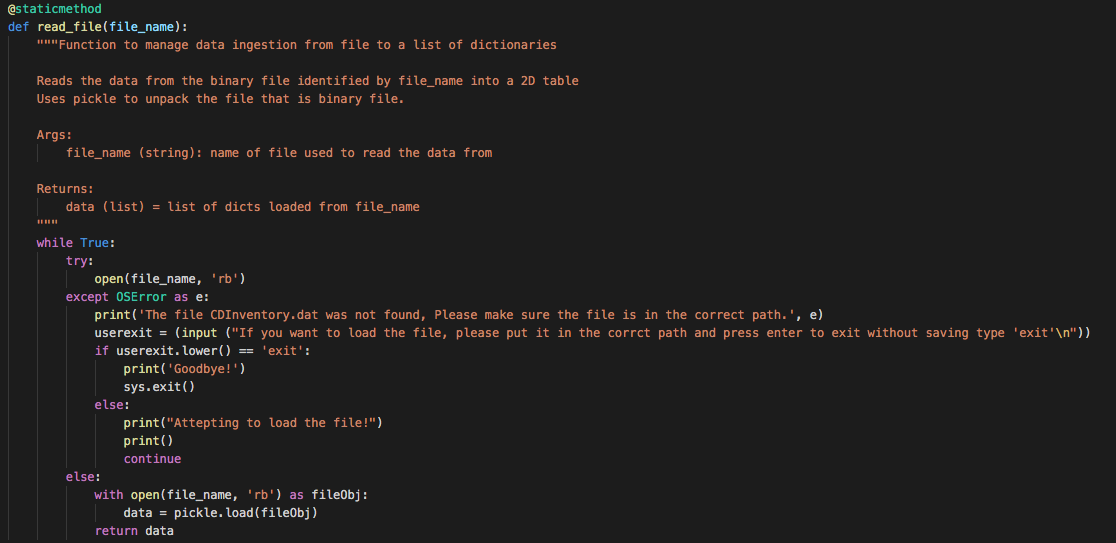
# Script Creation

Began by loading the pickle https://docs.python.org/3/library/pickle.html[[1]](#footnote-1)1 and sys modules.

Then declaring one variable not included in the original script, set the variable ‘saveFlag” to 0 for tracking unsaved changes.

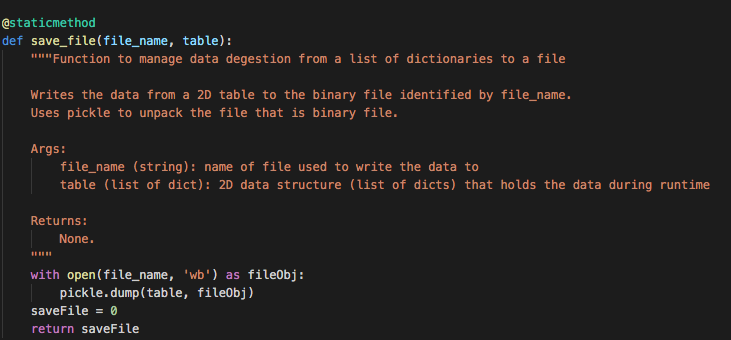
I will cover the changes form the previous script, I began by adding pickle to the read\_file function to load the pre built binary file (PYTHON PROGRAMMING FOR THE ABSOLUTE BEGINNER 3RD ADDITION, MICHAEL DAWSON, CENGAGE LEARNING 2010, P. 200-204) .

The read\_file function,



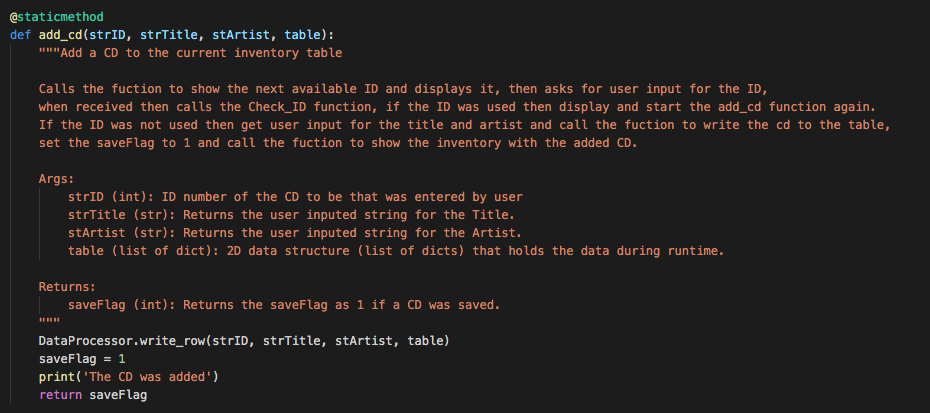
Here we can see the function will attempt to open the file declared in ‘file\_name’. If it fails it will catch the OSError https://docs.python.org/3/library/exceptions.html[[2]](#footnote-2)2, display to the user the error, pause and give the user a chance to put the file in the correct place for the script to load it. The script will attempt to load the file until the file is found or the user exits the script. Once loaded the script will open the file with the ‘rb’ flag for read binary and use the pickle module to load the file into ‘data’ which it then returns.

The next function save\_file,



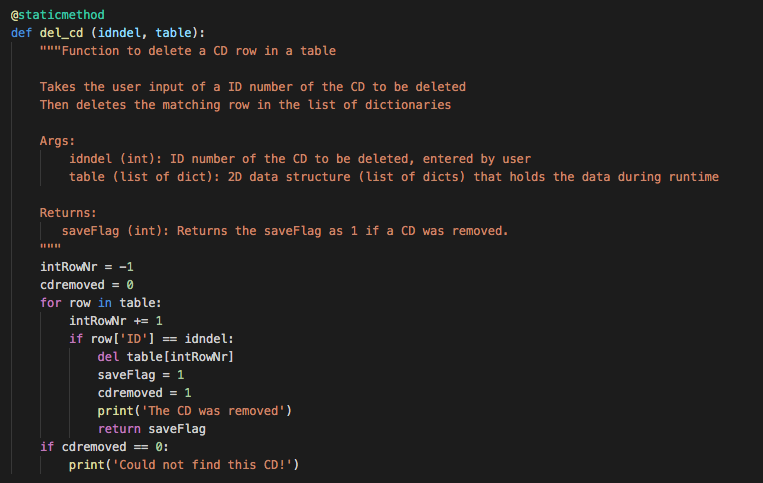
This function uses pickle as well to save the list of dicts to a Binalry file. Here the script uses the ‘wb’ flag for write binary and uses the pickle.dump module to save to the file. Then sets the ‘saveFlag’ which it then returns.

The next function is add\_cd,



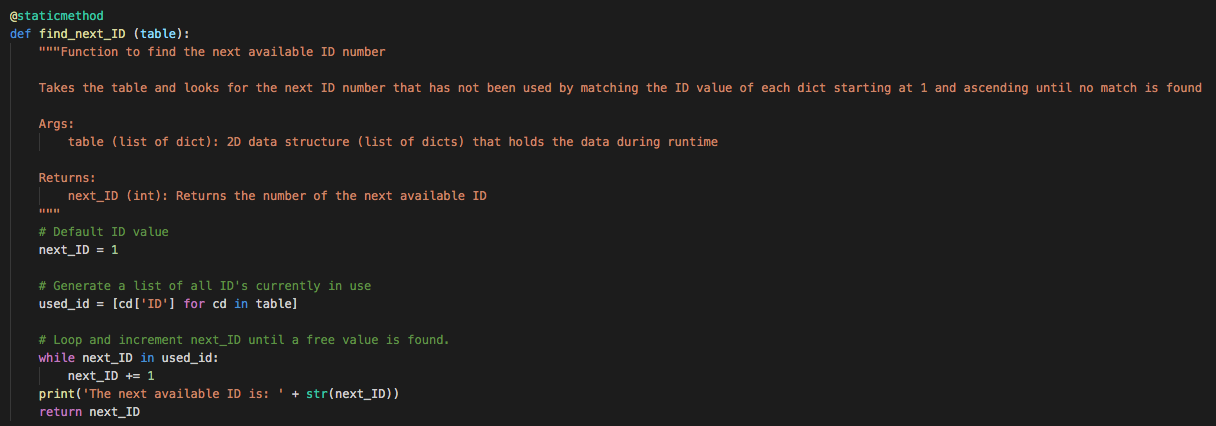
This function was moved to the DataProcessor Class. It now take the user input and passes it to DataProcessor.write\_row then changes the ‘saveFlag’ to 1 and returns it.

The next function is del\_cd,



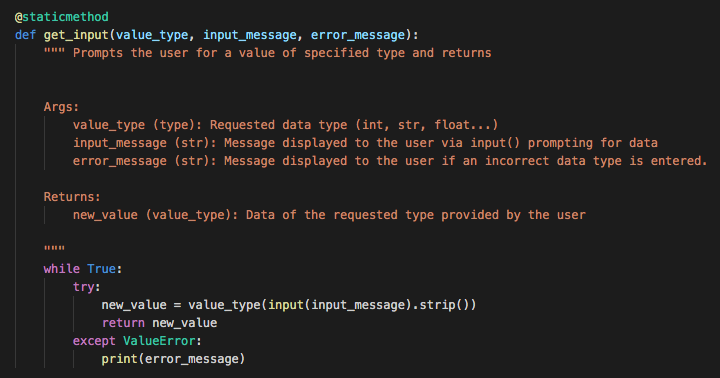
This function deletes an CD from user input. The variable “idndel” with “table” are used to try and match the cd to be deled my “ID”. If successful the saveFlag is set to 1 and returned. If not then the the users is told and the user is returned to the main menu.

The next function is find\_next\_id



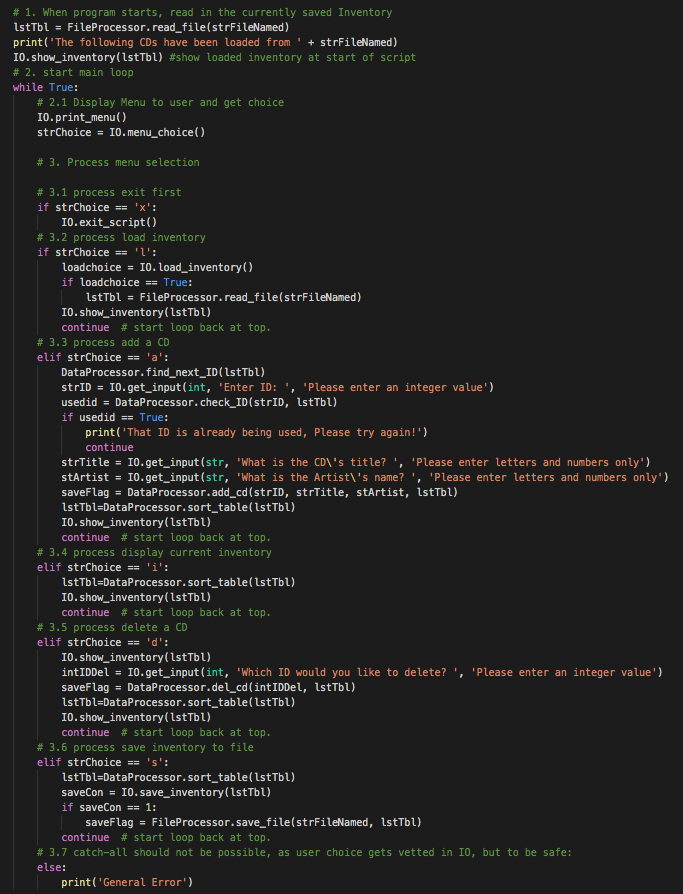
Here the function is taking the all the “ID’ in the list of dicts that are being used and looks for the first free ID counting up from one, when found that ID is displayed and returned.

The next function is get\_input,



This function is called when a user is inputting a value. When the data is inputted, the function will validate the input as correct for its type, if not the error message is displayed. This will loop until the user enters correct a correct value for its type and then returns that value.

The main body of the script,



starts by calling the IO.script\_start function with the variables “strFileNamed” and “lstTbl” to load and display the inventory from the file. Then the IO.print\_menu is displayed and user input is requested with the IO.Menu\_choice function. The users input is stored into “strChoice”. Then depending on the users input the script will do the following, If the user enters ‘x’ the function IO.exit\_script will be called. If ‘l’ is entered then the function IO.load\_inventory is called with the variables “strFileName” and “lstTbl” passed on. If ‘a’ is entered then the function IO.add\_cd is called with the variable “lstTbl” passed on. If ‘I’ is entered then the function IO.show\_inventory is called with the variable “lstTbl” passed on. If ‘d’ is entered then the function IO.del\_cd is called with the variable “lstTbl” passed on. If ‘s’ is entered then the function IO.save\_inventory is called with the variables “strFileName” and “lstTbl” passed on.

Several of the choices here have more code to help avoid cross Class calls. Add a cd for example calls several functions an passes the returned results along to complete the process.

# Performing the Script

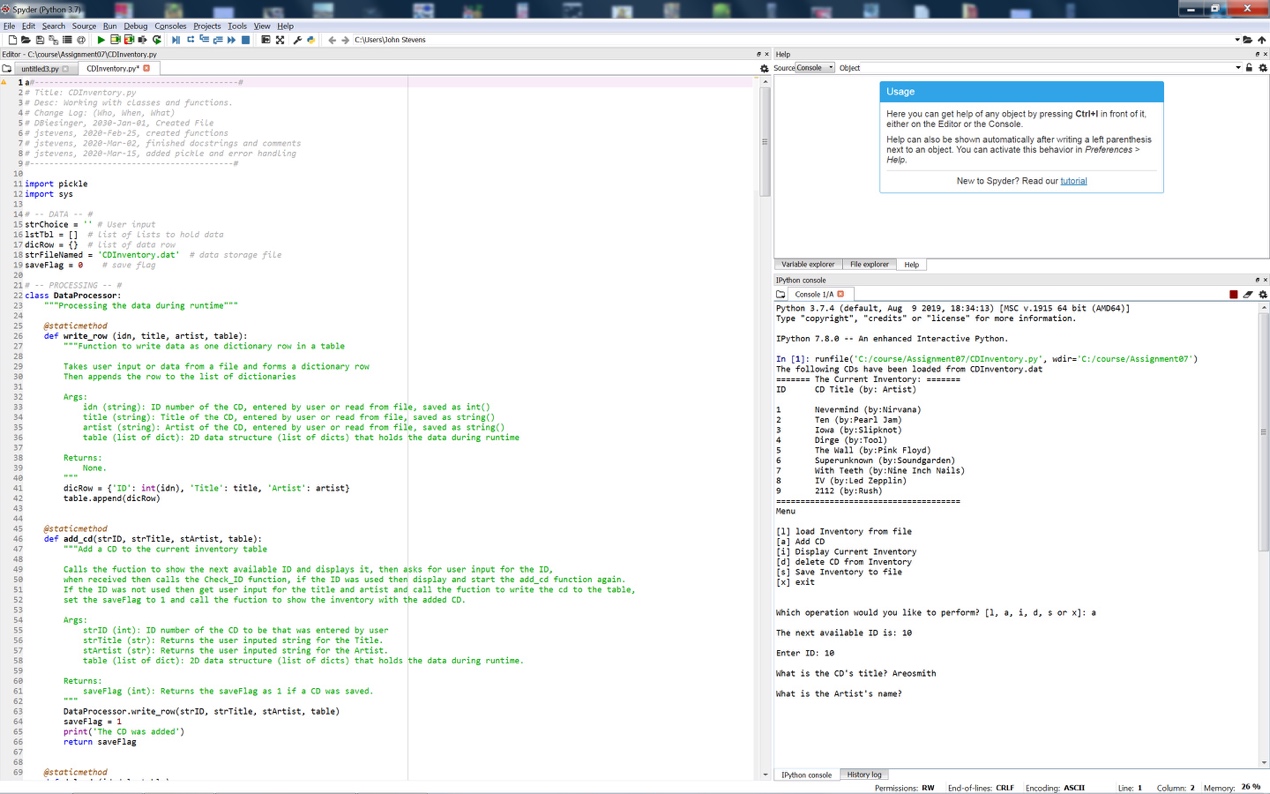


Figure - Script performing in Spyder.

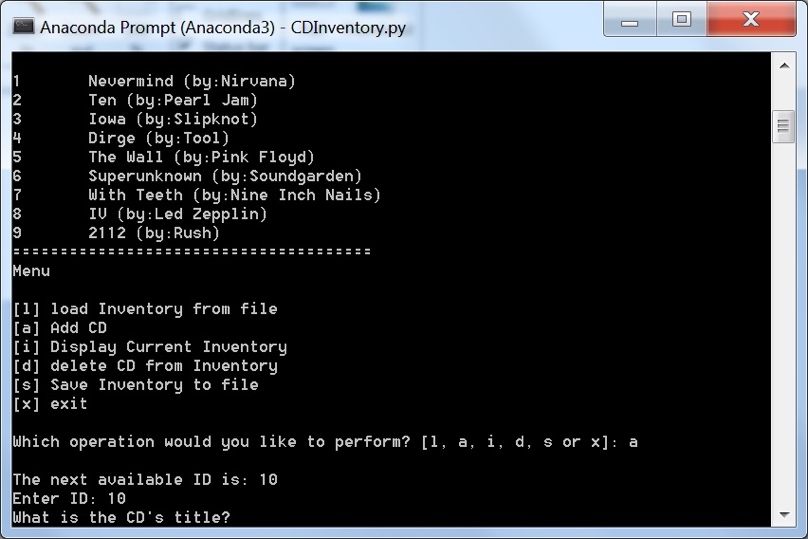


Figure 3 - Script performing in terminal

As seen in Figure 2 and Figure 3 I ran the script in Spyder and Terminal respectively. I loaded 9 test CDs from the file CDInventory.dat, then added 10th CD, displayed the list and saved the list to a file, then deleted the 10th CD.

# Summary

In this assignment I was able to modify an existing script to use pickle for loading and saving data to a binary file and add in structured error handling to provide expanded functionality to load, add, delete, show and save CDs to a list. This script and document and posted at https://github.com/pjfan73/Assignment07

1. 1 Retrieved 2020-03-12 [↑](#footnote-ref-1)
2. 2 Retrieved 2020-03-12 [↑](#footnote-ref-2)