James

3/13/2022

IT FDN 110 B: Introduction to Programming Python

Assignment 07

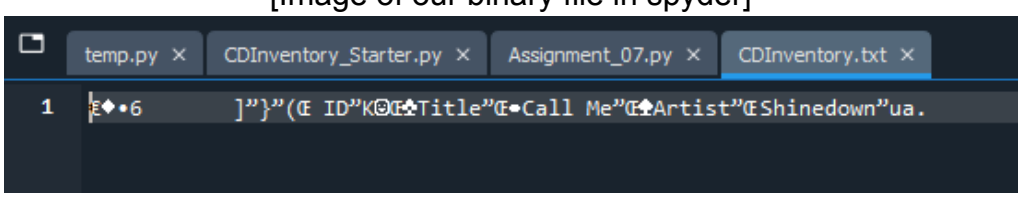
- Overview

Once again we use the basis of cd inventory to explore the relatively simple but extremely useful try: except: commands as well as the alternative method of binary data.

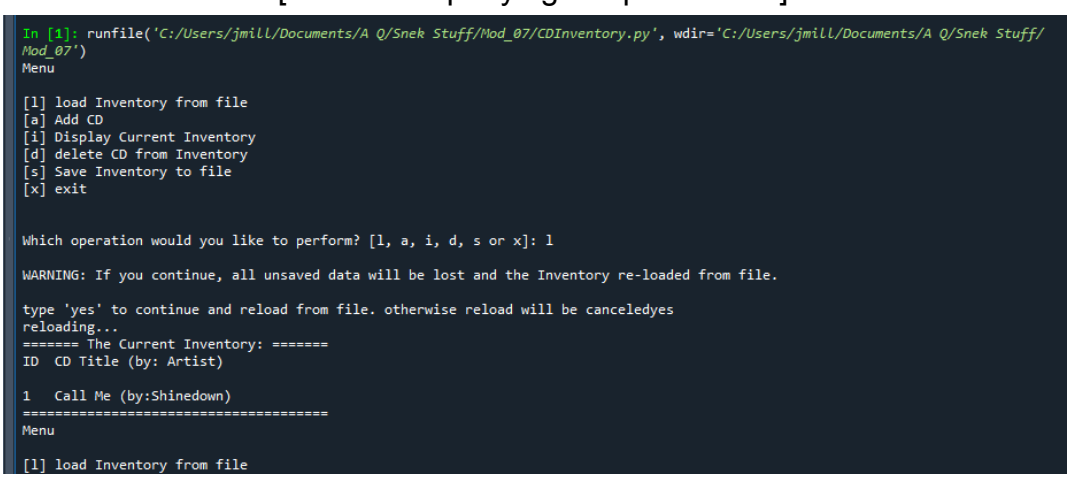
- Python, Try not closing over any errors

I am surprised I haven't heard about it before, as try is a very simple solution to a very common issue in python, that of potential user errors/situations we don't want to crash the script. Try: allows us to issue a command to python with the possibility of an error such as *trying to int() a letter* and fallback on an except: or more if this goes wrong instead of crashing. I found the site w3schools.com particularly helpful as it both outlined different tools related to try as well as allowing you to test scripts to answer some nuances on how exactly try works (ex: making sure it both tests the command for errors and runs the command if it encounters none)

[Image of our binary file in spyder]



[The Accompanying Script in Action]

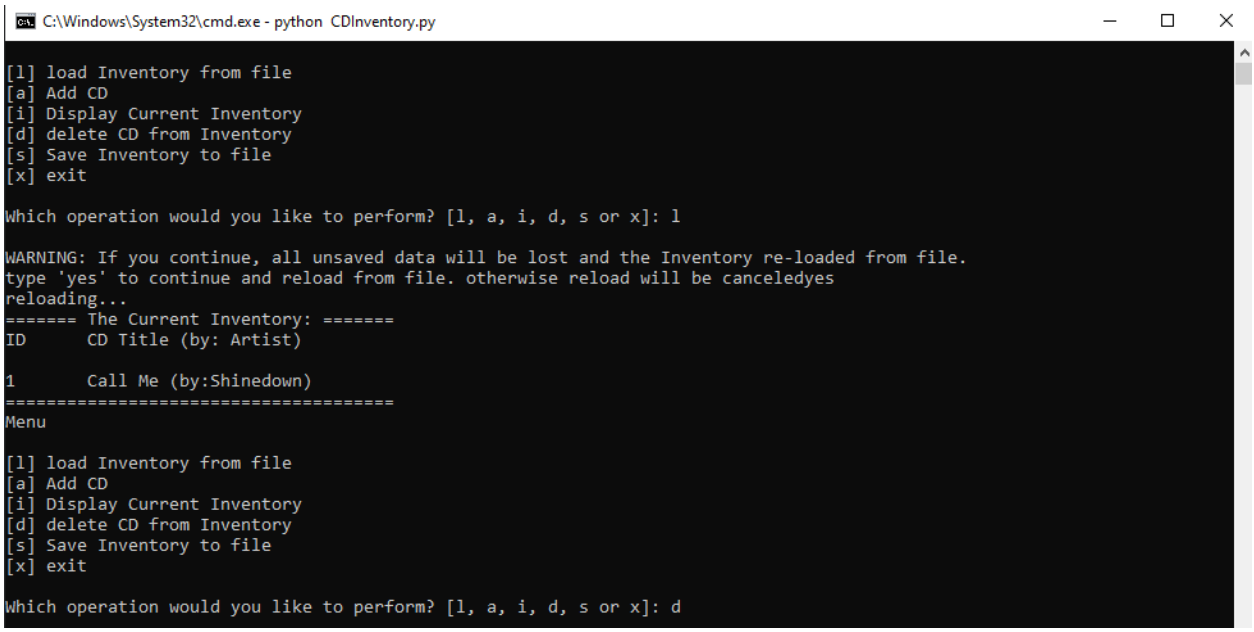


- In a Pickle with Pickling

While I was fortunately able to find a site that laid out the formatting for uploading to binary files and extracting from them, I still needed to run tests on how the data comes out of them before making the script around this. Using some simple print commands, I saw first hand that it returned the data in its original list of dictionaries form, which is quite helpful as it didn't need converting to fit into the old table replacement method and could in fact skip the for item append process.

However, I found that telling it to modify the function variable (lstTbl) that worked with the previous method now no longer modified the global variable. Through print statements I was able to deduce the issue and in class we covered global variables, but I still don't know what went wrong. Does append modify an input function variable while equals forms a new variable in the function by default?

[Reading from binary in CMD]



- Summary

I am still gushing about how useful try/except is and how much I missed it before I even know how. It seems relatively simple to use for how versatile it is as a tool and I am very glad to add it to my repertoire.

Special thanks to (https://www.w3schools.com/python/python_try_except.asp) and (https://www.synopsys.com/blogs/software-security/python-pickling/) whose direct examples and further details on Try/Except and pickling helped me understand how to utilize the new material.

- Appendix

```python
#------------------------------------------#
# Title: Assignment06_Starter.py
# Desc: Working with classes and functions.
# Change Log: (James Miller, 3/12/22, Cleaned up notes, added try/except fail safes)
# (converted file method to binary)
# DBiesinger, 2030-Jan-01, Created File
#------------------------------------------#

# -- DATA -- #
strChoice = '' # User input
lstTbl = []  # list of lists to hold data
dicRow = {}  # list of data row
strFileName = 'CDInventory.txt'  # data storage file
objFile = None  # file object
warning = ''
import pickle


# -- PROCESSING -- #
class DataProcessor:
    def add_single(vlist):
        # takes the input song and adds it to the current work log
        dicRow = {'ID': vlist[0], 'Title': vlist[1], 'Artist': vlist[2]}
        lstTbl.append(dicRow)
        print('The song '+ vlist[1] +' has been added to the current log')
        return

    def eliminate(info):
        intRowNr = -1
        blnCDRemoved = False
        for row in lstTbl:
            intRowNr += 1
            if row['ID'] == info:
                del lstTbl[intRowNr]
                blnCDRemoved = True
                break
        if blnCDRemoved:
            print('The CD was removed')
        else:
            print('Could not find this CD!')

class FileProcessor:
    """Processing the data to and from text file"""

    @staticmethod
    def read_file(file_name, table):
        """Function to manage data ingestion from file to a list of dictionaries

```

```
        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.

        Args:
            file_name (string): name of file used to read the data from
            table (list of dict): 2D data structure (list of dicts) that holds the data during runtime

        Returns:
            None.
        """
        table.clear()  # this clears existing data and allows to load data from file
        open(file_name, 'a') # added to allow the program to run without an initial text file
        objFile = open(file_name, 'rb')
        try:
            frompickle = pickle.load(objFile)
        except:
            frompickle = []
        global lstTbl
        lstTbl = frompickle
        # I don't know why, but now that I'm in pickle I needed to use global to get it to modify lstTbl
        # instead of just using the table variable (lstTbl)
        objFile.close()
        # Once again keeping the old script in case
        #for line in objFile:
        #    data = line.strip().split(',')
        #    dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
        #    table.append(dicRow)

    @staticmethod
    def write_file(file_name, table): #converted to binary
        # writes current inventory table to the file
        objFile = open(file_name, 'wb')
        pickle.dump(table, objFile)
        objFile.close()
        warning = 'online'
        return warning
        # In case something goes wrong, keeping the old script in notes
        #for row in table:
        #   lstValues = list(row.values())
        #   lstValues[0] = str(lstValues[0])
        #   objFile.write(','.join(lstValues) + '\n')


# -- PRESENTATION (Input/Output) -- #

class IO:
    """Handling Input / Output"""

```

```
    @staticmethod
    def print_menu():
        """Displays a menu of choices to the user

        Args:
            None.

        Returns:
            None.
        """

        print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
        print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')

    @staticmethod
    def menu_choice():
        """Gets user input for menu selection

        Args:
            None.

        Returns:
            choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x

        """
        choice = ' '
        while choice not in ['l', 'a', 'i', 'd', 's', 'x']:
            choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
            if choice not in ['l', 'a', 'i', 'd', 's', 'x']:
                print('Unusual input detected') # added a message to let user know why they were asked
for a new input
        print()  # Add extra space for layout
        return choice

    @staticmethod
    def show_inventory(table):
        """Displays current inventory table


        Args:
            table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.

        Returns:
            None.

        """
        print('======= The Current Inventory: =======')
        print('ID\tCD Title (by: Artist)\n')
```

```
        for row in table:
            print('{}\t{} (by:{})'.format(*row.values()))
        print('=====================================')

    def def_cd():
        userID = input('Enter ID: ').strip() #allows us to take the input once and test to avoid crashes
        try:
            strID = int(userID)
        except:
            print('Irregular ID input detected')
            strID = (userID)
        strTitle = input('What is the CD\'s title? ').strip()
        stArtist = input('What is the Artist\'s name? ').strip()
        latestcd = [strID,strTitle,stArtist]
        return latestcd
    def mark():
        target = (input('Enter the id number you wish to delete')) # edited to also avoid crashes for
unusual ids
        try:
            target = int(target)
        except:
            print ('Irregular ID detected and marked for removal')
        return target
    def confirm_save():
        flag = 'no'
        if warning == 'online':
            print('You have already saved to file this session,\ncontinuing may cause duplicate entrees')
        confirm = input('Save the current work log to inventory? Y/N:').strip().lower()
        if confirm == 'y':
            flag = 'yes'
        elif confirm == 'yes':
            flag = 'yes'
        return flag


# 1. When program starts, read in the currently saved Inventory
FileProcessor.read_file(strFileName, lstTbl)

# 2. start main loop
while True:
    # 2.1 Display Menu to user and get choice
    IO.print_menu()
    strChoice = IO.menu_choice()

    # 3. Process menu selection
    # 3.1 process exit first
    if strChoice == 'x':
        break
```

```
        # 3.2 process load inventory
        if strChoice == 'l':
            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 from file. otherwise reload will be
canceled')
            if strYesNo.lower() == 'yes':
                print('reloading...')
                FileProcessor.read_file(strFileName, lstTbl)
                IO.show_inventory(lstTbl)
            else:
                input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
                IO.show_inventory(lstTbl)
            continue  # start loop back at top.
        # 3.3 process add a CD
        elif strChoice == 'a':
            # 3.3.1 Ask user for new ID, CD Title and Artist, adds it, then shows the new table
            thecd = IO.def_cd()
            DataProcessor.add_single(thecd)
            IO.show_inventory(lstTbl)
            continue  # start loop back at top.
        # 3.4 process display current inventory
        elif strChoice == 'i':
            IO.show_inventory(lstTbl)
            continue  # start loop back at top.
        # 3.5 process delete a CD
        elif strChoice == 'd':
            # 3.5.1.1 display Inventory to user
            IO.show_inventory(lstTbl)
            # 3.5.1.2 ask user which ID to remove
            target = IO.mark()
            DataProcessor.eliminate(target)
            # 3.5.2 search thru table and delete CD
            IO.show_inventory(lstTbl)
            continue  # start loop back at top.
        # 3.6 process save inventory to file
        elif strChoice == 's':
            # 3.6.1 Display current inventory and ask user for confirmation to save
            IO.show_inventory(lstTbl)
            strYesNo = IO.confirm_save()
            # 3.6.2 Process choice
            if strYesNo == 'yes': #due to formatting, try/except is unecessary here
                warning = FileProcessor.write_file(strFileName, lstTbl)
                print('Current inventory log added to files.')
            else:
                input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
            continue  # start loop back at top.
        # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be safe:
```

```
        else:
            print('General Error')
```