Shane McLeod

27 November 2022

IT FDN 110 B

Assignment 7

Foundations of Python – Assignment 7

# Introduction

This document will outline concepts learned in Assignment 7

# GitHub Link

<https://github.com/shaneam12/Assignment_07>

# Assignment 7

Exception Handling

I found some good insight into exception handling. Those pages are here:

<https://www.geeksforgeeks.org/python-exception-handling/>

Pickling in Python

This website was a good source that helped me understand Pickling:

<https://snyk.io/blog/guide-to-python-pickle/>

## Assignment Comments

For this assignment, the goal was to modify the previous code from Assignment 6 to adapt the new processes we learned with Exception handling and Pickling. Exception handling is a very useful tool because it can help quickly define any errors if there is an issue that causes the program to crash. This allows us to remedy these errors easily without having to dig deep into the code to figure out what is causing the problem. For the pickle module, it is nice because it will serialize the data for you and input it into a file that is easy to read back later.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 1 - Screenshot of Code Working in Spyder

Text

Description automatically generated

Figure 2 - Screenshot of Code Working in Terminal

# Summary

In this assignment we learned the role of exception handling and pickling in our code.

# Appendix

## Listing CDInventory.py

#------------------------------------------#

# Title: Assignment06\_Starter.py

# Desc: Working with classes and functions.

# Change Log: (Who, When, What)

# DBiesinger, 2030-Jan-01, Created File

# Shane McLeod, 2022-Nov-26, Modified code for error handling and binary data

#------------------------------------------#

import pickle

# -- DATA -- #

strChoice = '' # User input

lstTbl = []  # list of lists to hold data

dicRow = {}  # list of data row

strFileName = 'CDInventory.dat'  # data storage file

objFile = None  # file object

# -- PROCESSING -- #

class DataProcessor:

    @staticmethod

    def add\_data(strID, strTitle, stArtist):

        """Add data to the table"""

        try:

            intID = int(strID)

            dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}

            lstTbl.append(dicRow)

        except Exception as e:

            print('There was an error.')

            print('Build in error info:')

            print(type(e), e, e,\_\_doc\_\_, sep='\n')

        return lstTbl

    @staticmethod

    def delete\_CD(lstTbl):

        """Delete CD from table"""

        intRowNr = -1

        blnCDRemoved = False

        for row in lstTbl:

            intRowNr += 1

            if row['ID'] == intIDDel:

                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

        try:

            objFile = open(file\_name, 'rb')

        except FileNotFoundError as e:

            print('There was an error.')

            print('Build in error info:')

            print(type(e), e, e,\_\_doc\_\_, sep='\n')

        try:

            data = pickle.load(objFile)

            # print(data)

            # dicRow = {'ID': int(float(data[0])), 'Title': data[1], 'Artist': data[2]}

            for x in range(0, len(data)):

                table.append(data[x])

        except Exception as e:

            print('There was an error.')

            print('Build in error info:')

            print(type(e), e, e,\_\_doc\_\_, sep='\n')

        objFile.close()

    @staticmethod

    def write\_file(file\_name, table):

        """Write data to text file"""

        try:

            objFile = open(file\_name, 'wb')

        except FileNotFoundError as e:

            print('There was an error.')

            print('Build in error info:')

            print(type(e), e, e,\_\_doc\_\_, sep='\n')

        for row in table:

            lstValues = list(row.values())

            try:

                lstValues[0] = str(lstValues[0])

            except Exception as e:

                print('There was an error.')

                print('Build in error info:')

                print(type(e), e, e,\_\_doc\_\_, sep='\n')

            pickle.dump(lstTbl, objFile)

        objFile.close()

# -- 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']:

            try:

                choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()

            except ValueError as e:

                print('Not and applicable character')

                print('Build in error info:')

                print(type(e), e, e,\_\_doc\_\_, sep='\n')

        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('======================================')

    @staticmethod

    def get\_inputs():

        """Grabs inputs from user"""

        try:

            strID = input('Enter ID: ').strip()

            strTitle = input('What is the CD\'s title? ').strip()

            stArtist = input('What is the Artist\'s name? ').strip()

        except ValueError as e:

            print('Not and applicable character')

            print('Build in error info:')

            print(type(e), e, e,\_\_doc\_\_, sep='\n')

        return strID, strTitle, stArtist

# 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

        strID, strTitle, stArtist = IO.get\_inputs()

        # 3.3.2 Add item to the table

        lstTbl = DataProcessor.add\_data(strID, strTitle, stArtist)

        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 get Userinput for which CD to delete

        # 3.5.1.1 display Inventory to user

        IO.show\_inventory(lstTbl)

        # 3.5.1.2 ask user which ID to remove

        try:

            intIDDel = int(input('Which ID would you like to delete? ').strip())

        except Exception as e:

            print('There was an error.')

            print('Build in error info:')

            print(type(e), e, e,\_\_doc\_\_, sep='\n')

        # 3.5.2 search thru table and delete CD

        DataProcessor.delete\_CD(lstTbl)

        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 = input('Save this inventory to file? [y/n] ').strip().lower()

        # 3.6.2 Process choice

        if strYesNo == 'y':

            # 3.6.2.1 save data

            FileProcessor.write\_file(strFileName, lstTbl)

        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 save:

    else:

        print('General Error')