Shane McLeod

3 December 2022

IT FDN 110 B

Assignment 8

Foundations of Python – Assignment 8

# Introduction

This document will outline concepts learned in Assignment 8

# GitHub Link

<https://github.com/shaneam12/Assignment_08>

# Assignment 8

For this assignment, the goal was to apply object-oriented programming to create a more optimized CD Inventory program. This was quite confusing for me, so I tried my best to create a program that fully took advantage of OOP. Using an outline and code from Assignment 6 for inspiration, I created the code seen in Assignment 8 where OOP is used. The most important class was the CD class because there is where all the properties are contained. The other classes (FileIO and IO) mainly just consisted of methods needed to properly run the code. I added the classes and methods into my code and added some exception handling to help identify any issues.

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 about object-oriented programming, what it is used for, and how to add/modify its attributes.

# Appendix

## Listing CDInventory.py

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

# Title: Assignmen08.py

# Desc: Assignnment 08 - Working with classes

# Change Log: (Who, When, What)

# DBiesinger, 2030-Jan-01, created file

# DBiesinger, 2030-Jan-01, added pseudocode to complete assignment 08

# Shane M, 2022-Dec-04, filled in needed code

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

# -- DATA -- #

strFileName = 'cdInventory.txt'

lstOfCDObjects = []

class CD:

    """Stores data about a CD:

    properties:

        cd\_id: (int) with CD ID

        cd\_title: (string) with the title of the CD

        cd\_artist: (string) with the artist of the CD

    methods:

    """

    # -- Fields -- #

    cd\_id = 0

    cd\_title = ''

    cd\_artist = ''

    # -- Constructor -- #

    def \_\_init\_\_(self, id, title, artist):

        # -- Attributes -- #

        self.\_\_cd\_id = id

        self.\_\_cd\_title = title

        self.\_\_cd\_artist = artist

    # -- Properties -- #

    @property

    def getData(self):

        """Returns list of data"""

        try:

            theID = int(self.\_\_cd\_id)

        except:

            raise Exception('The CD ID must be an integer')

        dicRow = {'ID': theID, 'Title': str(self.\_\_cd\_title), 'Artist': str(self.\_\_cd\_artist)}

        return dicRow

# -- PROCESSING -- #

class FileIO:

    """Processes data to and from file:

    properties:

    methods:

        save\_inventory(file\_name, lst\_Inventory): -> None

        load\_inventory(file\_name): -> (a list of CD objects)

    """

    # -- Fields -- #

    # -- Constructor -- #

        # -- Attributes -- #

    # -- Properties -- #

    # -- Methods -- #

    @staticmethod

    def save\_inventory(file, lst):

        """Saves data to text file"""

        try:

            with open(file, 'w') as objFile:

                for row in lst:

                    lstValues = list(row.values())

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

                    objFile.write(','.join(lstValues) + '\n')

                objFile.close()

        except Exception as e:

            raise Exception('There was an error saving the data to the file')

    def load\_inventory(file, lst):

        """Reads data from text file"""

        lst.clear()

        try:

            with open(file, 'r') as objFile:

                for line in objFile:

                    data = line.strip().split(',')

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

                    lst.append(dicRow)

            objFile.close()

        except:

            raise Exception('There was an error reading the data from the file')

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

class IO:

    """Handles user Input and Output"""

    @staticmethod

    def menu():

        """Function to diplay menu"""

        print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')

        print('[s] Save Inventory to file\n[x] exit\n')

    def userChoice():

        """Recieves input for User's choice"""

        choice = ' '

        while choice not in ['l', 'a', 'i', 's', 'x']:

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

        print()  # Add extra space for layout

        return choice

    def displayData(table):

        """Displays data in current list"""

        try:

            print('======= The Current Inventory: =======')

            print('ID\tCD Title (by: Artist)\n')

            for row in table:

                print('{}\t{} (by:{})'.format(\*row.values()))

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

        except:

            raise Exception('Unable to display data')

    def get\_inputs():

        """Grabs a new CD input"""

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

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

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

        return strID, strTitle, stArtist

# -- Main Body of Script -- #

# 1. Load data from file into a list of CD objects on script start

FileIO.load\_inventory(strFileName, lstOfCDObjects)

# Display menu to user

    # show user current inventory

    # let user add data to the inventory

    # let user save inventory to file

    # let user load inventory from file

    # let user exit program

# 2. start main loop

while True:

    # 2.1 Display Menu to user and get choice

    IO.menu()

    strChoice = IO.userChoice()

    # 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...')

            FileIO.load\_inventory(strFileName, lstOfCDObjects)

            IO.displayData(lstOfCDObjects)

        else:

            input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')

            IO.displayData(lstOfCDObjects)

        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

        theCD = CD(strID, strTitle, stArtist)

        lstOfCDObjects.append(theCD.getData)

        IO.displayData(lstOfCDObjects)

        continue  # start loop back at top.

    # 3.4 process display current inventory

    elif strChoice == 'i':

        IO.displayData(lstOfCDObjects)

        continue  # start loop back at top.

    # 3.5 process save inventory to file

    elif strChoice == 's':

        # 3.5.1 Display current inventory and ask user for confirmation to save

        IO.displayData(lstOfCDObjects)

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

        # 3.5.2 Process choice

        if strYesNo == 'y':

            # 3.5.2.1 save data

            FileIO.save\_inventory(strFileName, lstOfCDObjects)

        else:

            input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')

        continue  # start loop back at top.

    # 3.6 catch-all should not be possible, as user choice gets vetted in IO, but to be save:

    else:

        raise Exception('General Error')