James

3/20/2022

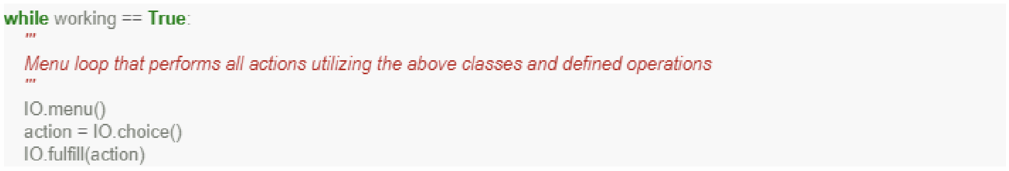
IT FDN 110 B: Introduction to Programming Python

Assignment 08

- Overview

This week we delved into classes, and while not as intuitive as some of our past formatting, they bring a lovely compression to the final operation, as offloading the functions allows for an insane amount of saved space.

[The mere four lines of code for the actual loop]

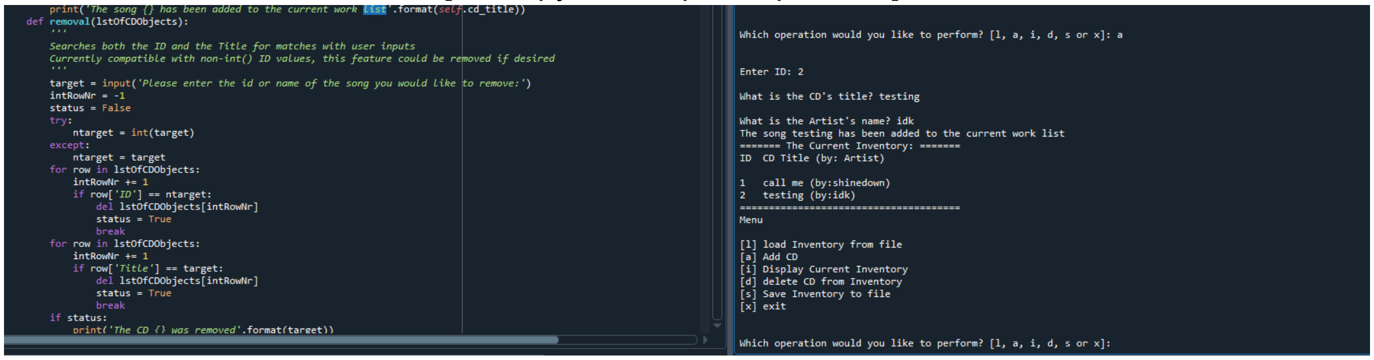


```
while working == True:
    '''
    Menu loop that performs all actions utilizing the above classes and defined operations
    '''
    IO.menu()
    action = IO.choice()
    IO.fulfill(action)
```

- Class is in Session

The biggest struggle this week for me was figuring out how to utilize class. While part of it was me overthinking it as a radically separate entity from functions and as a result not utilizing a lot of applicable lessons at first, even after clearing up those misunderstandings the formatting took me a bit of time. Single underscores where I needed double, putting self in when creating the object instead of letting the class function fill in itself from the formatting, and getting used to the application of class functions within class functions. Ultimately, I'm quite happy with how it turned out and after it was all said and done, I am still appalled at how little script the loop itself is contained in.

[The Spyder Script in Operation]



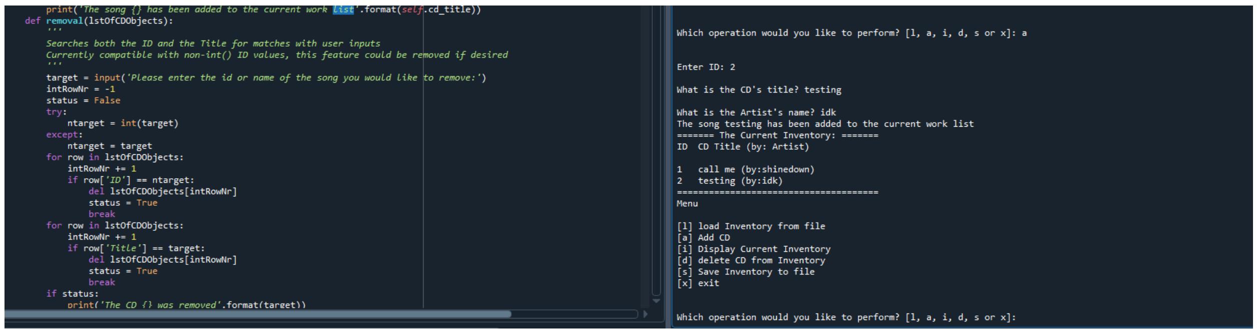
```
        print('The song {} has been added to the current work list'.format(self.cd_title))
    def removal(lstOfCDObjects):
        '''
        Searches both the ID and the Title for matches with user inputs
        Currently compatible with non-int() ID values, this feature could be removed if desired
        '''
        target = input('Please enter the id or name of the song you would like to remove:')
        intRowNr = -1
        status = False
        try:
            ntarget = int(target)
        except:
            ntarget = target
        for row in lstOfCDObjects:
            intRowNr += 1
            if row['ID'] == ntarget:
                del lstOfCDObjects[intRowNr]
                status = True
                break
        for row in lstOfCDObjects:
            intRowNr += 1
            if row['Title'] == target:
                del lstOfCDObjects[intRowNr]
                status = True
                break
        if status:
            print('The CD {} was removed'.format(target))
```

```
Which operation would you like to perform? [l, a, i, d, s or x]: a


Enter ID: 2

What is the CD's title? testing

What is the Artist's name? idk
The song testing has been added to the current work list
======= The Current Inventory: =======
ID  CD Title (by: Artist)

1   call me (by:shinedown)
2   testing (by:idk)
======================================
Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit


Which operation would you like to perform? [l, a, i, d, s or x]:
```

- Trying to Docstrings

I’m still quite new to the formatting on notes, and while digital resources were helpful, they vary based on personal preference. I went for brief operation descriptions with occasional questions that could be relevant to future changes in the code (ex: would we prefer if the function accepted irregular ID values or just reset to the top?) but I don’t know if for example I should spend space trying to credit who all helped with the final script or if I should leave that to the header.

[Example of it Running in Spyder and Docstrings]



```
        print('The song {} has been added to the current work list'.format(self.cd_title))
    def removal(lstOfCDObjects):
        '''
        Searches both the ID and the Title for matches with user inputs
        Currently compatible with non-int() ID values, this feature could be removed if desired
        '''
        target = input('Please enter the id or name of the song you would like to remove:')
        intRowNr = -1
        status = False
        try:
            ntarget = int(target)
        except:
            ntarget = target
        for row in lstOfCDObjects:
            intRowNr += 1
            if row['ID'] == ntarget:
                del lstOfCDObjects[intRowNr]
                status = True
                break
        for row in lstOfCDObjects:
            intRowNr += 1
            if row['Title'] == target:
                del lstOfCDObjects[intRowNr]
                status = True
                break
        if status:
            print('The CD {} was removed'.format(target))
```

```
Which operation would you like to perform? [l, a, i, d, s or x]: a


Enter ID: 2

What is the CD's title? testing

What is the Artist's name? idk
The song testing has been added to the current work list
======= The Current Inventory: =======
ID  CD Title (by: Artist)

1   call me (by:shinedown)
2   testing (by:idk)
======================================
Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit


Which operation would you like to perform? [l, a, i, d, s or x]:
```

- Summary

While the formatting took some getting used to, the final result and last three lines of script still have me impressed by how much I was underestimating the usefulness of classes when I started. I don’t know how they compare on computer run times, but the readability alone, especially in larger projects we may work on some day, is great.

- Appendix

```
#------------------------------------------------------#
# Title: Assignmen08.py
# Desc: Assignnment 08 - Working with classes
# James Miller, 2022-March-20, Ported over prior functions, reformatted, added coding
# DBiesinger, 2030-Jan-01, created file
# DBiesinger, 2030-Jan-01, added pseudocode to complete assignment 08
#------------------------------------------------------#

# -- DATA -- #
strFileName = 'cdInventory.txt'
lstOfCDObjects = []
working = True

class CD:
    def __init__(self, cd_id, cd_title, cd_artist):
        '''
        unlike the other operations I want to create new CD items to put into the list
        '''
        self.cd_id = cd_id
        self.cd_title = cd_title
        self.cd_artist = cd_artist
    def append(self, lstOfCDObjects):
        '''
        adds the input cd details to the active list of CDs, currently formatted for a list of dictionaries
        '''
        cd_dictionary = {'ID': self.cd_id, 'Title': self.cd_title, 'Artist': self.cd_artist}
        lstOfCDObjects.append(cd_dictionary)
        print('The song {} has been added to the current work list'.format(self.cd_title))
    def removal(lstOfCDObjects):
        '''
        Searches both the ID and the Title for matches with user inputs
        Currently compatible with non-int() ID values, this feature could be removed if desired
        '''
        target = input('Please enter the id or name of the song you would like to remove:')
        intRowNr = -1
        status = False
        try:
            ntarget = int(target)
        except:
            ntarget = target
        for row in lstOfCDObjects:
            intRowNr += 1
            if row['ID'] == ntarget:
                del lstOfCDObjects[intRowNr]
                status = True
                break
        for row in lstOfCDObjects:
            intRowNr += 1
```

```python
            if row['Title'] == target:
                del lstOfCDObjects[intRowNr]
                status = True
                break
        if status:
            print('The CD {} was removed'.format(target))
        else:
            print('Could not find the input {}'.format(target))
    pass

# -- PROCESSING -- #
class FileIO:
    def savelog(lstOfCDObjects, strFileName):
        '''
        Saves inventory to the file after user confirmation and warns of potential duplicate issues
        Went back to csv as the instructions weren't clear on which format was desired and csv
        is easier to manually check for formating errors'
        '''
        confirm = input('If you are ready to save the work log, \ntype \'yes\' to proceed, otherwise we will return to the menu: ').lower().strip()
        if confirm == 'yes':
            try:
                objFile = open(strFileName, 'a')
                for row in lstOfCDObjects:
                    lstValues = list(row.values())
                    lstValues[0] = str(lstValues[0])
                    objFile.write(','.join(lstValues) + '\n')
                    objFile.close()
                input('Inventory saved, attempting to save again may cause duplicate entries\nPress [ENTER] to return to the menu')
            except:
                print('An unknown error occurred when attempting to save to the file')
        else:
            input('The inventory was NOT saved to the file. Press [ENTER] to return to the menu.')

    def loadinventory(lstOfCDObjects, strFileName):
        '''
        Loads inventory from the file after confirmation from the user, and warns of potential duplicate issues
        '''
        print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
        confirm = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled: ').lower().strip()
        if confirm == 'yes':
            try:
                lstOfCDObjects.clear()  # this clears existing data and allows to load data from file
                open(strFileName, 'a') # added to allow the program to run without an initial text file
                objFile = open(strFileName, 'r')
                for line in objFile:
                    data = line.strip().split(',')
                    dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
                    lstOfCDObjects.append(dicRow)
                    objFile.close()
                input('Inventory loaded, attempting to save may cause duplicate entries\nPress [ENTER] to return to the menu')
                IO.display(lstOfCDObjects)
            except:
                print('An unkown error occurred when attempting to load the file')
                input('canceling... Inventory data NOT reloaded. Press [ENTER] to return to the menu.')
                IO.display(lstOfCDObjects)
        else:
            input('The inventory was NOT loaded from the file. Press [ENTER] to return to the menu.')
    pass

class IO:
    def menu():
        '''
        Displays users options as a repeatable menu prompt
        '''
        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')
    def choice():
        '''
        Requests and returns the users selection for the menu, could be combined with IO.menu
        but there are use cases to keeping them seperate
        '''
        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')
        print()
        return choice
    def display(lstOfCDObjects):
        '''
        Ported over the lovely inventory display  of DBiesinger and tweaked the formatting for class compatibility
        '''
        print('======= The Current Inventory: =======')
        print('ID\tCD Title (by: Artist)\n')
        for row in lstOfCDObjects:
            print('{}\t{} (by:{})'.format(*row.values()))
        print('======================================')
    def requestcd():
        '''
        Gets cd information to use in CD.append
        '''
        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 fulfill(request):
        '''
        Using the user input it directs the script through the proper operations
        '''
        if request == 'x':
            global working
            working = False
            return
        if request == 'l':
            FileIO.loadinventory(lstOfCDObjects, strFileName)

        elif request == 'a':
            cdinfo = IO.requestcd()
            newcd = CD(cdinfo[0],cdinfo[1],cdinfo[2])
            CD.append(newcd,lstOfCDObjects)
            IO.display(lstOfCDObjects)

        elif request == 'i':
            IO.display(lstOfCDObjects)

        elif request == 'd':
            CD.removal(lstOfCDObjects)
            IO.display(lstOfCDObjects)

        elif request == 's':
            IO.display(lstOfCDObjects)
            FileIO.savelog(lstOfCDObjects, strFileName)

        else:
            print('An error has occurred')
    pass

while working == True:
    '''
    Menu loop that performs all actions utilizing the above classes and defined operations
    '''
```

```
IO.menu()
action = IO.choice()
IO.fulfill(action)
```