**Mitchell Post**

**MAR-07-2021**

**IT FDN 110 A**

**MOD 8**

# **Introduction**

This week we explored object-oriented programming using classes and constructors. Last week we passed in data from one function to another using variables and argument assignments. This week we used the .self utility combined with getters and setters to assigned and pass through variables globally and locally as needed. From my brief exposure to this concept it seems like python likes to communicate in this fashion. A lot of the objectives we were doing in the past weeks seem to be achieved smoother in this way.

It seems like now using object-oriented programming that the last couple weeks was long hand using lists and dics in lists. For instance, the way python handled objects is much more robust then lists, they seem to self-encapsulate and even output cleaner when called on to read the objects.

# **Assignment**

* Using weeks 8’s CDInventory.py pseudo code perform the following tasks
  + Read and understand the pseudocode.
  + Add code to make the application work.
  + Make sure to include error handling.
  + Test Functionality of program as whole

**Read and understand the pseudocode**

After reading the pseudocode it became apparent most of the code used in last weeks program will be utilized again in this week’s assignment. The while loop will be mostly unchanged, and I decided that I would use pickle to transfer the data to file as it seems to be more robust then the write to txt method, we learned the week prior. With this I saw that the main bulk of the assignment would be in changing the methods in which the program takes in user data, creates its list (now with objects) and passes data around now using .self and \_\_init\_\_ method

I created a mini program below to understand this concept once. This program was basically the user input function that would be needed in the main code for the week and past weeks as well as an append function. Then in the while loop I called for user input twice to test the how multiple objects would be handled and read then printed the data back. After I got this working, I applied it to the pseudo code with code taken from the previous week.

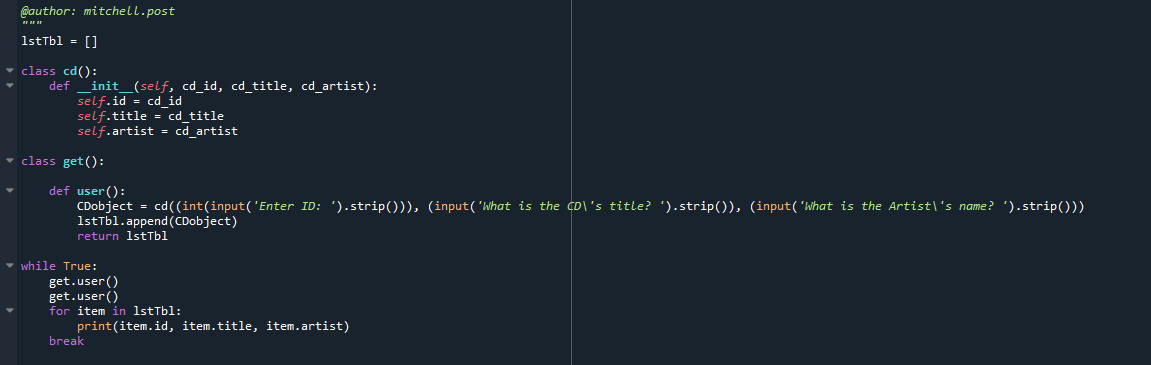


Figure 1 – Proof of concept Program

**Add code to make the application work / Make sure to include error handling.**

Like I mentioned in the last section I added some code from last week in combination of the new code that I created to handle this week’s curriculum. With that I submitted for review and it was mentioned that I need to add the Getter and Setters so that is the last thing I did for adding code to the pseudo code for week 8. You can see below each value for the \_\_init\_\_ has a getter and setter applied. I used error handling from last weeks code to in the main loop.

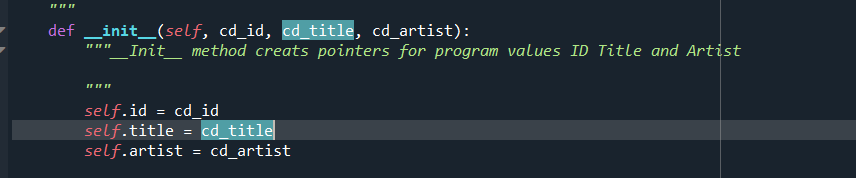


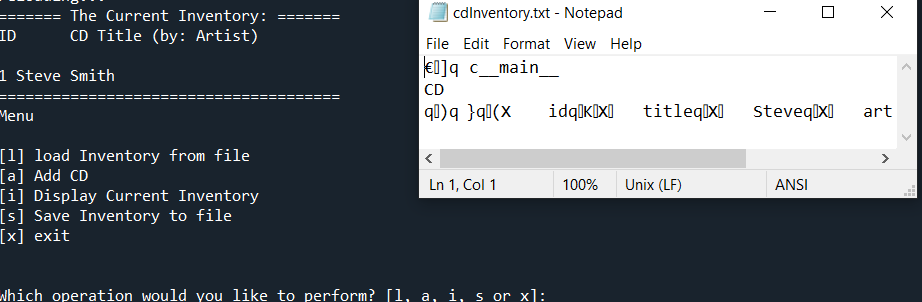
Figure 2 – INIT and variables defined



Figure 3 – Getters and Setters

**Test Functionality of program as whole**

After applying this week’s concepts, I was able to test the program as a whole. I did notice that in this Pseudo code and assignment there was no mention of “delete” utility, so I removed that from the program. This might not be necessary, but it has been taught to give the customer exactly what was outline so I thought it would be appropriate to remove it.



*Figure 4 – Testing Functionality pickle output*

**References**

Dawson, M. (2009). *Python® Programming for the Absolute Beginner, Third Edition*. Course Technology PTR.

# **Appendix**

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

# Title: Assignmen08.py

# Desc: Assignnment 08 - Working with classes

# Change Log: (Who, When, What)

# MPOST, 1200-MAR-04, Troubleshot object O programming

# MPOST, 1400-MAR-04, Created Aux program for proof of concept

# MPOST, 1430-MAR-04, Tested Succesfull

# MPOST, 1500-MAR-04, Added code to week 8's psuedocode

# MPOST, 0900-MAR-07, Added getter/Setters updated docstring

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

# -- DATA -- #

strFileName = 'cdInventory.txt'

lstTbl = []

import pickle

class CD:

"""Stores data about a CD:

properties:

self.id = cd\_id

self.title = cd\_title

self.artist = cd\_artist

Getter and setter for each property setup below

"""

def \_\_init\_\_(self, cd\_id, cd\_title, cd\_artist):

"""\_\_Init\_\_ method creats pointers for program values ID Title and Artist

"""

self.id = cd\_id

self.title = cd\_title

self.artist = cd\_artist

@property

def ID(self):

return self.id

@ID.setter

def ID(self,value):

if type(value) == int:

self.id = value

@property

def TITLE(self):

return self.title

@TITLE.setter

def TITLE(self,value):

if type(value) == str:

self.title = value

@property

def ARTIST(self):

return self.artist

@ARTIST.setter

def ARTIST(self,value):

if type(value) == str:

self.artist = value

pass

# -- 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)

"""

@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 object

Args:

file\_name (string): name of file used to read the data from

Returns:

None.

"""

try:

table.clear() # this clears existing data and allows to load data from file

with open(file\_name, "rb+") as objFile:

table = pickle.load(objFile)

return table

except:

print("\n\n\n\*\*\*CDInventory.dat not found\*\*\*\n\n\n")

@staticmethod

def write\_file(strFileName, lstTbl):

""" Additional file proccesing function to save file to .txt

Function saves objects to file using pickle method"""

with open(strFileName, "wb+") as objFile: #Pickle to save objects to file

pickle.dump(lstTbl, objFile)

pass

# -- 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('[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', '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

@staticmethod

def show\_inventory(table):

"""Displays current inventory table

Args:

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

Returns:

None.

"""

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

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

for item in lstTbl:

print(item.id, item.title, item.artist)

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

@staticmethod

def input\_user():

"""User input taken then assigned to strID, strTitle, and stArtist

Args:

lstTbl.append(CDobject)

Returns:

lstTbl

"""

while True:

try:

CDobject = CD((int(input('Enter ID: ').strip())), (input('What is the CD\'s title? ').strip()), (input('What is the Artist\'s name? ').strip()))

lstTbl.append(CDobject)

return lstTbl

break

except ValueError:

print("Invalid entry. ID must be integer")

pass

while True:

IO.print\_menu()

strChoice = IO.menu\_choice()

if strChoice == 'x':

break

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...')

lstTbl = FileIO.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 (No Change)

elif strChoice == 'a':

IO.input\_user()

IO.show\_inventory(lstTbl)

continue # start loop back at top.

elif strChoice == 'i':

IO.show\_inventory(lstTbl) ## Calls for show function dispalying user inventory

continue # start loop back at top.

elif strChoice == 's':

IO.show\_inventory(lstTbl) ## Calls for show function dispalying user inventory

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

if strYesNo.lower() in ['yes', 'y']:

FileIO.write\_file(strFileName, lstTbl) ## Calls for to save inventory to target .txt file

else:

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

continue # start loop back at top.

### catch-all should not be possible, as user choice gets vetted in IO, but to be save (no change):

else:

print('General Error')