Bita Massoudi

12/03/2022

Foundations of Programming: Python

Assignment 08

bitamass/Assignment 08 (github.com)

Object Oriented Programming

Introduction

In this assignment I learned about the concept of OOP(object oriented programming) and used it to complete the CDInventory script. In OOP, there exists a concept called 'class' that lets the programmer structure the codes of software in a fashioned way. Because of the use of classes and objects, the programming became easy to understand and code.

Explanation of the script

Three attributes were created for the CD object: cd_id, cd_title, and cd_artist. I initiated them by using __init__. I also defined 3 getters and 3 setters for the3 attributes (Fig 1). By making the attributes private, users can not access and modify the attributes.

```
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:
    txtList: return a string with 3 properties infomation.
    """

# TODONE Add Code to the CD class

def __init__(self, cd_id, cd_title, cd_artist):
    # attributes #

    self. __id= cd_id
    self. __ittle= cd_title
    self. __artist= cd_artist

# properties #

#property
def cd_id(self):
    return self.__id

#property
def cd_artist(self):
    return self.__artist

#cd_id.setter
def cd_id(self, value):
    if int(value).isstr():
        raise Exception('Please choose an integer')
    else:
    self.__cd_id = value

#cd_title.setter
def cd_title(self, value):
    self.__cd_title = value

#cd_artist.self, value):
    self.__cd_artist = value

# Methods #

def __str__(self):
    return self.txtList()

def txtList(self):
    return self.cd_artist
```

Fig 1. Class CD and attributes related

For class IO and class FileIO, I defined all functions in the classes as *staticmethod*. Staticmethod would allow for data processing and organization of statements.

The data (CD objects) is stored as lstOfCDObjects [CDobj1, CDobj2, CDobj3......]. This can be displayed as for-loop (Fig 2).

```
@staticmethod
def show_inventory(table):
    """Displays current inventory table
Args:
        table: list of object
        row: string for each line
    """
    print('======= The Current Inventory: =======')
    print('ID\tCD Title (by: Artist)\n')

# Use a for loop to print this "CD objects" list
    for row in table:
        print('row)
    print('=============')
```

Fig 2. For loop to show inventory

Staticmethod was used for read and write functions.

```
@staticmethod
def read_file(strFileName, lstOfCDObjects):
    """Function to read stored binary data
    file_name (string): name of file used to read the data from
    rowData: [0]=> CD, [1]=>TITLE, [2]=>Artist
    cd_info: object of CD info
    """
     lstOfCDObjects.clear()
     objFile = open(strFileName, 'r')
      for row in objFile:
           rowData = row.strip().split(',')
cd_info = CD(int(rowData[0]), rowData[1], rowData[2])
           lstOfCDObjects.append(cd_info)
     objFile.close()
           print('Text file does not exist!')
print('Build in error info:')
           print(type(e), e, e.__doc__, sep='\n')
def write_file(strFileName, lst0fCDObjects):
    """Save CD data
      Args:
          file_name (string): name of file used to save the data table: list of CD object(lstOfCDObjects)
           objList: A string from txtList
     while True:
                strYesNo = input('Save this inventory to file? [y/n] ').strip()
                 if not strYesNo:
           except Exception:
     print('Didn\'t enter anything! Type again!!')
# 3.6.2 Process choice 'y'
      # save data to txt file
if strYesNo == 'y':
           objFile = open(strFileName, 'w')
for item in lstOfCDObjects:
   objList = item.txtList()
                 objFile.write(objList + '\n')
           objFile.close()
            print('\nCD DATA SAVED\n')
           input('The inventory was NOT saved to file. Press [ENTER] to return
```

Fig 3. Script for read and write functions

I then tested the program:

I ran the script, added CD details and saved the data. Fig 4 illustrates the add and save functions.

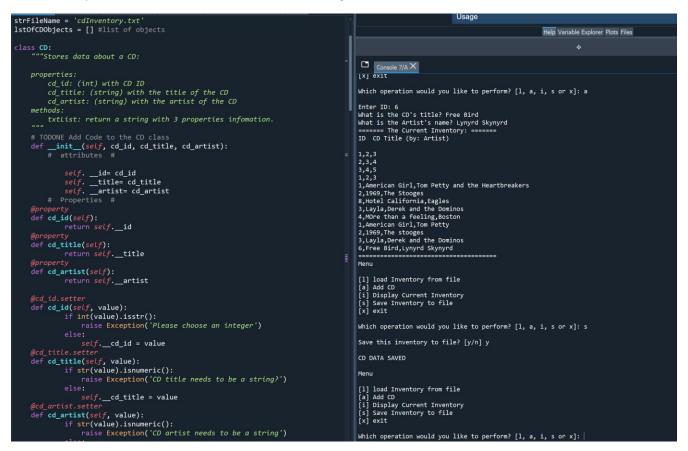


Fig 4. Add and save functions tested

I tested the load function and fig 5 illustrates this function.

```
def txtList(self):
                                                                                                                                                                                                           Save this inventory to file? [y/n] y
-- PROCESSING -- #
                                                                                                                                                                                                           CD DATA SAVED
                                                                                                                                                                                                           Menu
             "Processes data to and from file:
                                                                                                                                                                                                           [1] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit
      methods:
                 save_inventory(file_name, lst_Inventory): -> None
load_inventory(file_name): -> (a list of CD objects)
                                                                                                                                                                                                           Which operation would you like to perform? [l, a, i, s or x]: l
                                                                                                                                                                                                           MARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file. type 'yes' to continue and reload from file. otherwise reload will be canceled:
                                                                                                                                                                                                           reloading...
====== The Current Inventory: ======
ID CD Title (by: Artist)
      @staticmethod
def read_file(strfileName, lstOfCDObjects):
    """Function to read stored binary data
    file_name (string): name of file used to read the data from
    rowData: [0]=> CD, [1]=>JITILE, [2]=>Artist
    cd_info: object of CD info
                                                                                                                                                                                                           1,2,3
2,3,4
3,4,5
1,2,3
1,American Girl,Tom Petty and the Heartbreakers
2,1969,The Stooges
8,Hotel California, Eagles
3,Layla,Derek and the Dominos
4,More than a feeling,Boston
1,American Girl,Tom Petty
2,1969,The Stooges
3,Layla,Derek and the Dominos
6,Free Bird,Lynyrd Skynyrd
     try:
lstofCDObjects.clear()
objFile = open(strFileName, 'r')
for row in objFile:
    rowData = row.strip().split(',')
    cd_info = CD(int(rowData[0]), rowData[1], rowData[2])
    lstofCDObjects.append(cd_info)
    burile close()
               istOfCDObjects.appendee_____
objfile.close()
ept FileNotFoundError as e:
    print('Text file does not exist!')
    print('Bulld in error info:')
    print(type(e), e, e.__doc__, sep='\n')
                                                                                                                                                                                                           [1] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] exit
       @staticmethod
def write_file(strFileName, lstOfCDObjects)
                                                                                                                                                                                                            Which operation would you like to perform? [1, a, i, s or x]:
```

Fig 5. Load function testing

I tested the display script and function. Fig 6 illustrates display script.

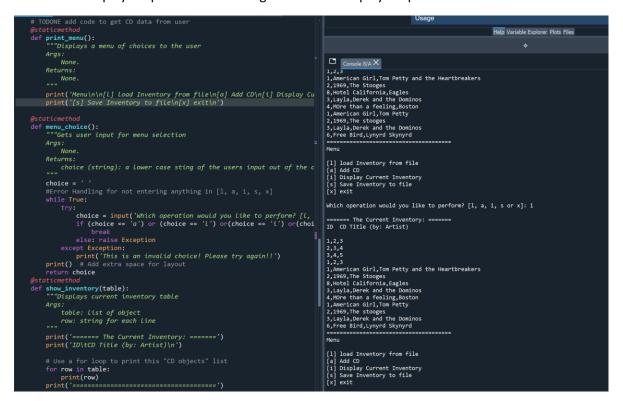


Fig 6. Display function testing

I finally moved to test all functionalities in the terminal window. Below are the screenshots (Fig 7, 8, 9) of the script working in the terminal window.

Fig 7. Add and save functions in terminal window

```
Which operation would you like to perform? [l, a, i, d, s or x]: l
WARNING: If you continue, all unsaved data will be lost and the Inventory r
e-loaded from file.
type 'yes' to continue and reload from file. otherwise reload will be cance
led:
yes
reloading...
====== The Current Inventory: ======
       CD Title (by: Artist)
1,2,3
2,3,4
1,American Girl,Tom Petty and the Heartbreakers
2,1969,The Stooges
8,Hotel California,Eagles
3,Layla,Derek and the Dominos
4,MOre than a feeling,Boston
1,American Girl,Tom Petty
2,1969,The stooges
3,Layla,Derek and the Dominos
```

Fig 8. Load function in terminal window

```
Which operation would you like to perform? [l, a, i, d, s or x]: i
====== The Current Inventory: ======
ID
       CD Title (by: Artist)
1,2,3
2,3,4
3,4,5
1,2,3
1,American Girl,Tom Petty and the Heartbreakers
2,1969,The Stooges
8,Hotel California,Eagles
3,Layla,Derek and the Dominos
4,MOre than a feeling,Boston
1,American Girl,Tom Petty
2,1969,The stooges
3,Layla,Derek and the Dominos
Menu
[1] load Inventory from file
[a] Add CD
[i] Display Current Inventory
```

Fig 9. Display function in terminal window

I also added several error handling. The screenshots below illustrate some of these error handling scripts and results (Fig 10 a, b; 11 a, b; 12 a, b).

Fig 10 a Error handling script against an invalid choice

```
Which operation would you like to perform? [l, a, i, s or x]: j
This is an invalid choice! Please try again!!
Which operation would you like to perform? [l, a, i, s or x]:
```

Fig 10 b. Error handling result against an invalid choice

```
@staticmethod
def write_file(strFileName, lstofCDObjects):
    """Save CD data
Args:
    file_name (string): name of file used to save the data
    table: list of CD object(lstofCDObjects)
    objlist: A string from txtList
    """
while True:
    try:
        strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
        if not strYesNo:
            raise Exception
            break
    except Exception:
        print('Didn\'t enter anything! Type again!!')
```

Fig 11 a Error handling script for save option

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

Save this inventory to file? [y/n]

Didn't enter anything! Type again!!
```

Fig 11 b Error handling result for save option

```
@staticmethod
def CDInput():
    """User Enter CD data
    Returns: intID, strTitle, stArtist
    """
    while True:
        strID = input('Enter ID: ').strip()
        #Error Handling if user enter non-numeric value
        try:
            intID = int(strID)
            break
        except ValueError:
            print('This is not an integer! Enter again!')
    strTitle = input('What is the CD\'s title? ').strip()
    stArtist = input('What is the Artist\'s name? ').strip()
    return intID, strTitle, stArtist
```

Fig 12 a. Error handling script against string input when integer is expected

```
Which operation would you like to perform? [1, a, i, s or x]: a

Enter ID: five
This is not an integer! Enter again!
```

Fig 12 b. Error handling results for string input when integer is expected

Conclusion:

This week we learned about OOP and applied the concepts to the CD Inventory. A class is a technique to group functions and data members and put them in a container so that they can be accessed later by using a dot (.) operator. Objects are the basic runtime entities of object-oriented programming. It defines the instance of a class. Objects get their variables and functions from classes, and the class we will be creating are the templates made to create the object.