Shaked Bason
September-02-2020
Foundations of Programming (Python)
Assignment 08

Assignment 08

Instruction

In this assignment, questions such as “What is the difference between a class and the objects made from a class?,” and “How are fields and attributes and property functions related?” were answered. Throughout the assignment, I learned the difference between a property and a method.

Github Link for my Assignment - https://github.com/shakba/Assignment08

Module 8 and Labs – Step by step

Classes

Classes are the blueprint for an object. A class packages the data and functionality of an object. A class is a piece of software that defines all the members and methods of the object. For example, Saturn is one of many planets. Although each planet is unique, they all have common characteristics. It is possible to define a class of “planet”. The “Planet” class, will contain the properties common to all planets.

Class Structure

Fields

Fields are the data stores of a class. We can create fields in the same way of creating variables.

Constructors

Constructors are a method that is invoked when creating an object. You can use constructors to ensure proper data types in the fields. The method __init__ runs automatically whenever an object of the class is created in the computer memory. Keyword “Self” points to the object on which the method operates. Therefore we add the parameter “self” to the methods. Constructors run once during creation of the object and limited to this one purpose and hence are implicitly called when creating an object with a function call like syntax.

Attributes

Attributes are internal fields or variables that hold data. Attributes are defined to reference the parameter values passed through the __init__ method.

Properties

Properties are special methods which you can control validity of values assigned to attributes in a class and to make the attributes private and enforce the interaction with them thru methods that have control mechanisms built in. every private attribute usually has a pair of properties called “getter” and “setter”.
“getter”- get the attribute.

“setter”- set the attribute to a value.
When defining “getters” and “setters” in Python, we need to make sure that at first we define the getter property followed by the setter property.

Methods

We will use the methods as we use the functions in a script. They allow me to organize my statements into blocks that can be invoked by calling the method’s name.
The __str__ method returns some or all of the objects data as string. The method returns the name of the class and an address identifier. In this module we also got familiar with Static methods. We will use these methods when we want methods to be called on class level and not on instance level.
Static methods are usually defined like any other methods in a class. To indicate a static method, the decorator @staticmethod is used
A private method is a Class method that can only be called from inside the Class where it is defined.

Researchers

- https://docs.python.org/3/tutorial/classes.html
- https://www.afternerd.com/blog/python-private-methods/
- https://www.educative.io/edpresso/what-is-a-python-class-attribute

Creating a program

I noted what the code does and added it to this document using the planet-b website.

Finally, I created my code for this assignment (Appendix01), our known and familiar CDInventory. I modify my code to work with some new instructions.

Appendix #1

```
CD_Inventory.py', wdir='/Users/shakedbason/_FDNprogramming/Mod_08/LABS AND CODES/
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]: a


Enter ID: 1

What is the CD's title? wish you were here

What is the Artist's name? pink floyed
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Wish You Were Here (by:Pink Floyed)
======================================
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]:
```

Figure 1 - Spyder output

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


Enter ID: 1

What is the CD's title? perfect

What is the Artist's name? ed shearn
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Perfect (by:Ed Shearn)
=====================================
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]: d

======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Perfect (by:Ed Shearn)
=====================================

Which ID would you like to delete ? one
\ID CAN NOT BE STRING!

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]:
```

Figure 2 - Spyder output 2

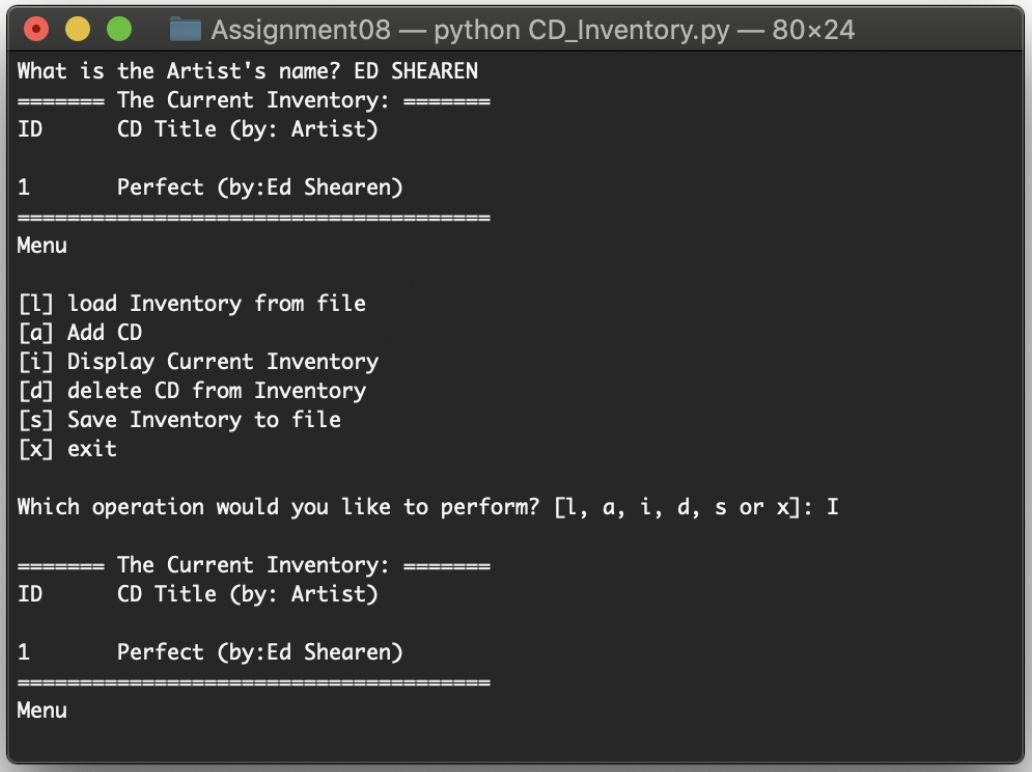

Figure 3 - Terminal output

Summary

Module 8 taught me about working with classes, working with classes structure includes fields, constructors, attributes, properties and methods. I continued improving my usage of classes and functions from the previous modules. I used Spyder as my IDE on this assignment.

Appendix

#1 CDInventory.py

```
#------------------------------------------#
# Title: CD_Inventory.py
# Desc: Assignnment 08 working with classes
# Change Log: (Who, When, What)
#shakedbason, 2020-Aug-30, created file
#shakedbason, 2020-Aug-30, added modified
#------------------------------------------#

import pickle

# -- DATA -- #
strFileName = 'cdInventory.dat'
lstOfCDObjects = []
lstIdIndex = []

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:

    """
    #Make all attributes private.
    def __init__(self, cd_id, cd_title, cd_artist):

        self.__cdId = int(cd_id)
        self.__cdTitle = cd_title.title()
        self.__cdArtist = cd_artist.title()

    @property
    def cd_Id(self):
        return self.__cdId

    @cd_Id.setter
    def cd_Id(self, newId):
        self.__cdId = int(newId)

    @property
    def cd_Title(self):
        return self.__cdTitle

    @cd_Title.setter
```

```
    def cd_Title(self, newTitle):
        self.__cdTitle = newTitle

    @property
    def cd_Artist(self):
        return self.__cdArtist

    @cd_Artist.setter
    def cd_Artist(self, newArtist):
        self.__cdArtist = newArtist

    def __str__(self):
        return f'{self.cd_Id}\t{self.cd_Title} (by:{self.cd_Artist})'

# -- 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 load_inventory(file_name):
        """Function manage data ingestion from file to a list of dictionaries
        Reads data from binary file and use pickle
        Args:
            file_name (string type)
            table (list of objects)
        Returns:
            None.
        """
        with open(file_name, 'rb') as objFile:
            val = pickle.load(objFile)
            print('\nCdInventory loaded\n')
            return val

    @staticmethod
    def save_inventory(file_name, table):
        """Function writes and add data to file
        appends data
        Args:
            file_name (string type)
            table (list of objects)
        Returns:
            None.
```

```
            """
            with open(file_name, 'wb') as objFile:
                pickle.dump(table, objFile)
            print('\nCDInventory saved\n')

    # -- PRESENTATION (Input/Output) -- #
    class IO:
        """Handling Input / Output"""
        @staticmethod
        def print_menu():
            """Displays a menu by user choice
            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']:
                choice = input('Which operation would you like to perform? [l, a, i, d, 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 CD objects): list data structure (list of CD objects) that holds the data during runtime.
            Returns:
                None.
            """
            print('======= The Current Inventory: =======')
            print('ID\tCD Title (by: Artist)\n')
            for row in table:
                # Prints the CD object based on __str__ method
                print(row)
```

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

        @staticmethod
        def addItem():
            """Function to get user input for ID, title, and artist

            Args:
                None.

            Returns:
                StrID (string)
                Strtitle (string)
                StArtist (string)
            """
            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

        @staticmethod
        def delItem(intIDDel, table, spotId):
            """Function to DELETE existing data from table.

            Args:
                idRemove (int): ID to remove CD data
                table (list of dic): 2D data structure
                spotId(List)

            Returns:
                None
            """
            intRowNr = -1
            blnCDRemoved = False
            for row in spotId:
                intRowNr += 1
                if intIDDel == row:
                    del table[intRowNr]
                    blnCDRemoved = True
                    break
            if blnCDRemoved:
                print('The CD was removed')
            else:
                print('Could not find this CD!')

    # -- Main Body of Script -- #
    # 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...')
            lstOfCDObjects.clear()
            cdFile = FileIO.load_inventory(strFileName)
            lstOfCDObjects.extend(cdFile)
            IO.show_inventory(lstOfCDObjects)
        else:
            input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue
to the menu.')
            IO.show_inventory(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
        userID, userTitle, userArtist = IO.addItem()

        # Check for blank fields
        if(not userID or not userTitle or not userArtist):
            print("Cannot leave ID, CD Title or Artist Name field blank!\n")
            continue
        newCD = CD(userID, userTitle, userArtist)
        lstOfCDObjects.append(newCD)
        IO.show_inventory(lstOfCDObjects)
    # 3.4 process display current inventory

    elif strChoice == 'i':
        IO.show_inventory(lstOfCDObjects)

    # 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(lstOfCDObjects)
        # 3.5.1.2 ask user which ID to remove
        try:
            intIdSel = int(input('Which ID would you like to delete ? ').strip())
        except ValueError:
```

```
            print('\ID CAN NOT BE STRING!\n')
            continue
        lstIdIndex.clear()
        for row in lstOfCDObjects:
            lstIdIndex.append(row.cd_Id)
        # 3.5.2 search thru table and delete CD
        IO.delItem(intIdSel, lstOfCDObjects, lstIdIndex)
        IO.show_inventory(lstOfCDObjects)
    # 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(lstOfCDObjects)
        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
            FileIO.save_inventory(strFileName,lstOfCDObjects)
        else:
            input('The inventory was NOT saved to file. Press [ENTER] to return to the
menu.')
    # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be
save:
    else:
        print('General Error')

```