Jennifer Magat

August 29, 2021

Foundations of Programming (Python)

Assignment_08

Applying Objected Oriented Programming a script

Introduction

For the 8th Module, we learned about the basics of Object Oriented Programming (OOP). The purpose of this document is to explore some of those learnings and apply those to develop code for the CDInventory.py script.

Objected Oriented Programming

Object-oriented programming (OOP) is a method of structuring a program by bundling related properties and behaviors into individual objects.[1] Classes are useful in defining objects. For the assignment we have identified several classes (CD, FileIO, IO) that contain the definition of objects that get called in code in the main body of the script. The main components of a class includes fields, constructor, properties and methods. In the CDInventory.py script the constructor section of the code defined the CD ID, CD Title and the CD Artist which where the values assigned to various components of the classes within the script. Within the constructor, we used self to represent the instance of the class. With this keyword, you can access the attributes and methods of the class in python.[2] Within the script most of the functions were contained in the method section. In this area we define the processes or functions that are available when an object is called.

Modifying the CDInventory.py script.

For this assignment, we were provided a starter script that provided pseudocode and a framework for the code that we need to develop when updating the CDInventory.py script.

First I added code to the CD class under the Constructor section. In class we learned that it is good practice to have the property and the setter to placed right next to each other to ensure that both aspects of the code are created as a pair. Figure 1 below displays the objects defined in the CD class.

[1] https://realpython.com/python3-object-oriented-programming , retrieved 2021-Aug-29
[2] https://www.edureka.co/blog/self-in-python , retrieved 2021-Aug-29

```python
    # TODOne Add Code to the CD class


    # -- Constructor -- #
    def __init__(self, cd_id, cd_title, cd_artist):
        self.__cd_id = cd_id
        self.__cd_title = cd_title
        self.__cd_artist = cd_artist

    @property
    def cd_id(self):
        return self.__cd_id

    @cd_id.setter
    def cd_id(self, new_cd_id):
        self.__cd_id = new_cd_id

    @property
    def cd_title(self):
        return self.__cd_title


    @cd_title.setter
    def cd_title(self, new_cd_title):
        self.__cd_title = new_cd_title

    @property
    def cd_artist(self):
        return self.__cd_artist


    @cd_artist.setter
    def cd_artist(self, new_cd_artist):
        self.__cd_artist = new_cd_artist

```

Figure 1 – objects defined in the CD class

Next I added the code to process the data from file. It has two parts, one is the load_inventory to file and the second one is to save_inventory to file. In the last week's assignment, the data being loaded in the program was a binary file but in this case, we are referencing a text file. The pickle method is not implemented in this week's assignment. Figure 2 below displays the code for loading inventory from the text file.

```python
    # TODOne Add code to process data from a file
    @staticmethod
    def load_inventory(file_name, lst_Inventory):
        """Function to load inventory from a text file

        Reads the data from file identified by file_name into a 2D table

        Args:
            file_name (string): name of file used to read the data from
            lst_Inventory: 2D data structure holding data

        Returns:
            None.

        Raises:
            FileNotFoundError
        """
        try:
            with open(file_name, 'r') as objFile:
                lst_Inventory.clear()
                for line in objFile:
                    data = line.strip().split(',')
                    addCD = CD(int(data[0]), data[1], data[2])
                    lst_Inventory.append(addCD)
        except FileNotFoundError:
            print('File not found: ', file_name)

```

Figure 2 – code to load_inventory from txt file

I struggled in the section for save_inventory to file. When I opened the CDInventory.txt file it appears that the information saved refers to the property object but not the actual data representing the lst_Inventory table. Figure 3 below displays the data displayed in the text file.

CDInventory.txt - Notepad

File Edit Format View Help

```
<property object at 0x000002595CCA09F0>,<property object at 0x000002595CCA06D0>,<
<property object at 0x000002595CCA09F0>,<property object at 0x000002595CCA06D0>,<
<property object at 0x000002595CCA09F0>,<property object at 0x000002595CCA06D0>,<
```

Figure 3 – Unexpected data saved in the CDInventory.txt file.

I played around with the code for the save_inventory function and made sure I had the exact reference for the objects. Eventually, I landed on a fix of making sure that in line 117 in Figure 3 below that class reference ‘cd’ was not capitalized. I’m still not sure why that fix worked since the class CD is in caps.

```python
    # TODOne Add code to process data to a file

    @staticmethod
    def save_inventory(file_name, lst_Inventory):

        """Function to save current CD Inventory list into text file

        Args:
            file_name (string): name of file to write data to
            lst_Inventory (list of dict): 2D data structure (list of dicts) that holds the data during runtime

        Returns:
            None.
        """
        try:
            with open(file_name, 'w') as objFile:
                for cd in lst_Inventory:
                    objFile.write("{},{},{}\n".format(cd.cd_id, cd.cd_title, cd.cd_artist))
        except FileNotFoundError:
            print('File not found: ', file_name)
```

Figure 4 – code to save_inventory to txt file

Next I added code to the class IO which were focused on defining the methods for processing the user input and output. Figure 5 below displays the methods defined in the IO class.

```python
# -- PRESENTATION (Input/Output) -- #
class IO:
    """Processes input and outputs from user and menu ptions

    properties:
        none

    methods:
        print_menu(): -> None
        menu_choice(): -> choice
        show_inventory(table): -> None
        get_cd_data(): -> ID, title, artist
        add_inventory(strID, strTitle, strArtist, lst_Inventory): -> Adds objects to a list
```

Figure 5 – methods defined in the IO class

Lastly, I added code to the main loop to load the data from file at the start of the script and displaying (and implementing) the menu options available in the CDInventory.py program. Figure 6 below displays the code developed for the main loop.

```python
# 1. When program starts, read in the currently saved Inventory
FileIO.load_inventory(strFileName, lstOfCDObjects)

# DONE: Display menu to user

# 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


    # DONE: let user exit program
    # 3.1 process exit first
    if strChoice == 'x':
        break


    # DONE: let user load inventory from file
    # 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.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.


    # DONE: let user add data to the inventory
    # 3.3 process add a CD
    elif strChoice == 'a':
        # 3.3.1 Ask user for new ID, CD Title and Artist
        intID, strTitle, strArtist = IO.get_cd_data()

        # 3.3.2 Add item to the table
        IO.add_inventory(intID, strTitle, strArtist, lstOfCDObjects)
        IO.show_inventory(lstOfCDObjects)

        continue  # start loop back at top.


    # DONE: let user save inventory to file
    # 3.4 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
            # TODOne move processing code into function
            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.
```

Figure 6 – code for the Main loop

This was another area that I struggled with. I found inconsistencies in my use of the keywords which resulted in errors when calling out functions. As an example, I would sometimes interchange the method get_cd_data() with get_cd_info(). I had to go back to the code and fix multiple errors like that when I was testing the code.

Running the Python Script

After saving the file, I ran the CDInventory script in both Spyder and the Terminal. Figures 7 and 8 below display the script working on the computer ensuring that all options are running correctly.

```
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.22.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/_FDProgramming/Mod_08/Assignment08/Assignment_08.py', wdir='C:/_FDProgramming/
Mod_08/Assignment08')
Menu

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] Exit


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

WARNING: 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:  yes
reloading...
======= The Current Inventory: =======
ID  CD Title (by: Artist)

1   The Big Wheel (by: Runrig)
2   Bad (by: Michael Jackson)
======================================
Menu

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] Exit


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


Enter ID: 3

What is the CD's title? Abbey Road

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

1   The Big Wheel (by: Runrig)
2   Bad (by: Michael Jackson)
3   Abbey Road (by: The Beattles)
======================================
Menu
```

Figure 7 – Running CDInventory.py in Spyder

```
=====================================
Menu

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] Exit


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

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

1   The Big Wheel (by: Runrig)
2   Bad (by: Michael Jackson)
3   Abbey Road (by: The Beattles)
=====================================
Menu

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] Exit


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

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

1   The Big Wheel (by: Runrig)
2   Bad (by: Michael Jackson)
3   Abbey Road (by: The Beattles)
=====================================

Save this inventory to file? [y/n] y
Menu

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] Exit


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


In [2]:
```

Figure 7a – Running CDInventory.py in Spyder – continued

```
Anaconda Prompt (anaconda3)

(base) C:\> cd C:\_FDProgramming\Mod_08\Assignment08

(base) C:\_FDProgramming\Mod_08\Assignment08> python CDInventory.py
Menu

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] Exit

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

WARNING: 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:  yes
reloading...
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       The Big Wheel (by: Runrig)
2       Bad (by: Michael Jackson)
======================================
Menu

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] Exit

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

Enter ID: 3
What is the CD's title? Abbey Road
What is the Artist's name? The Beattles
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       The Big Wheel (by: Runrig)
2       Bad (by: Michael Jackson)
3       Abbey Road (by: The Beattles)
======================================
Menu

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] Exit

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

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

1       The Big Wheel (by: Runrig)
2       Bad (by: Michael Jackson)
3       Abbey Road (by: The Beattles)
======================================
```

Figure 8 – Running CDInventory.py in Anaconda

Anaconda Prompt (anaconda3)

```
=====================================
Menu

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] Exit

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

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

1       The Big Wheel (by: Runrig)
2       Bad (by: Michael Jackson)
3       Abbey Road (by: The Beattles)
=====================================
Menu

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] Exit

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

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

1       The Big Wheel (by: Runrig)
2       Bad (by: Michael Jackson)
3       Abbey Road (by: The Beattles)
=====================================
Save this inventory to file? [y/n] y
Menu

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to file
[x] Exit

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


(base) C:\_FDProgramming\Mod_08\Assignment08>
```

Figure 8a – Running CDInventory.py in Anaconda – continued

Figures 9 below displays saved table in the CDInventory.txt file.

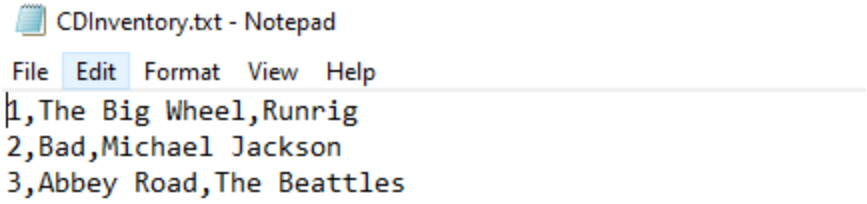

Figure 9 –CDInventory.txt

GitHub

The assignment files including CDInventory.py, CDInventory.txt, and the Know_Doc_08.pdf have been upload to GitHub - https://github.com/jlmagat/Assignment_08 .

Summary

For this assignment, we continue to build on the various concepts we have learned since the beginning of this class up to Module 8. We have applied the basics of Object Oriented Programming in the CDInventory.py script where we applied what we learned in developing classes, objects, constructors and methods.

Appendix

Listing CDInventory.py – part 1

```python
#------------------------------------------#
# Title: CDInventory.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
# JMagat, 2021-Aug-29, modified to add code to complete assignment_08
#------------------------------------------#

# -- 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:
        None.

    """
    # TODOne Add Code to the CD class


    # -- Constructor -- #
    def __init__(self, cd_id, cd_title, cd_artist):
        self.__cd_id = cd_id
        self.__cd_title = cd_title
        self.__cd_artist = cd_artist

    @property
    def cd_id(self):
        return self.__cd_id

    @cd_id.setter
    def cd_id(self, new_cd_id):
        self.__cd_id = new_cd_id

    @property
    def cd_title(self):
        return self.__cd_title


    @cd_title.setter
    def cd_title(self, new_cd_title):
        self.__cd_title = new_cd_title

    @property
    def cd_artist(self):
```

Listing CDInventory.py – part 2

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


    @cd_artist.setter
    def cd_artist(self, new_cd_artist):
        self.__cd_artist = new_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)

    """
    # TODOne Add code to process data from a file
    @staticmethod
    def load_inventory(file_name, lst_Inventory):
        """Function to load inventory from a text file

        Reads the data from file identified by file_name into a 2D table

        Args:
            file_name (string): name of file used to read the data from
            lst_Inventory: 2D data structure holding data

        Returns:
            None.

        Raises:
            FileNotFoundError
        """
        try:
            with open(file_name, 'r') as objFile:
                lst_Inventory.clear()
                for line in objFile:
                    data = line.strip().split(',')
                    addCD = CD(int(data[0]), data[1], data[2])
                    lst_Inventory.append(addCD)
        except FileNotFoundError:
            print('File not found: ', file_name)



    # TODOne Add code to process data to a file

    @staticmethod
    def save_inventory(file_name, lst_Inventory):
```

Listing CDInventory.py – part 3

```python
    def save_inventory(file_name, lst_Inventory):

        """Function to save current CD Inventory list into text file

        Args:
            file_name (string): name of file to write data to
            lst_Inventory (list of dict): 2D data structure (list of dicts) that holds the data during runtime

        Returns:
            None.
        """
        try:
            with open(file_name, 'w') as objFile:
                for cd in lst_Inventory:
                    objFile.write("{},{},{}\n".format(cd.cd_id, cd.cd_title, cd.cd_artist))
        except FileNotFoundError:
            print('File not found: ', file_name)


# -- PRESENTATION (Input/Output) -- #
class IO:
    """Processes input and outputs from user and menu ptions

    properties:
        none

    methods:
        print_menu(): -> None
        menu_choice(): -> choice
        show_inventory(table): -> None
        get_cd_data(): -> ID, title, artist
        add_inventory(strID, strTitle, strArtist, lst_Inventory): -> Adds objects to a list
    """

    # TODOne add docstring
    # -- Fields -- #
    # -- Constructor -- #
    # -- Attributes -- #
    # -- Properties -- #

    # TODOne add code to show menu to user

    @staticmethod
    def print_menu():
        """Displays a menu of choices to the user

        Args:
            None.

        Returns:
            None.
        """
```

Listing CDInventory.py – part 4

```
        """

        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')


    # TODO add code to captures user's choice
    @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


    # TODOne add code to display the current data on screen
    @staticmethod
    def show_inventory(lst_Inventory):
        """Displays current inventory table


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

        Returns:
            None.

        """
        print('======= The Current Inventory: =======')
        print('ID\tCD Title (by: Artist)\n')
        for CD in lst_Inventory:
            print('{}\t{} (by: {})'.format(CD.cd_id, CD.cd_title, CD.cd_artist))
        print('======================================')


    # TODOne add code to get CD data from user
    @staticmethod
    def get_cd_data():

        """Function to collect CD Data from the user: CD ID, Album, Artist

        Args:
```

Listing CDInventory.py – part 5

```
        Args:
            None.

        Returns:
            ID (integer): this is the CD ID entered by the user
            title (string): this is the CD's title
            artist (string): this is the Artist of the CD

        Raises:
            ValueError: When value entered is not a number

        """
        while True:
            try:
                ID = int(input('Enter ID: ').strip())
                break
            except ValueError:
                print('That is not valid CD ID! Please enter an integer')
        title = input('What is the CD\'s title? ').strip()
        artist = input('What is the Artist\'s name? ').strip()
        return ID, title, artist

    @staticmethod
    def add_inventory(strID, strTitle, strArtist, lst_Inventory):
        """Function to add a new entry to the inventory

        Args:
            strID (int): this is the CD ID entered by the user
            strTitle (string): this is the CD's title
            strArtist (string): this is the Artist of the CD
            lst_Inventory: 2D data structure holding data

        Returns:
            None.
        """
        try:
            intID = int(strID)
            addCD = CD(intID, strTitle, strArtist)
            lst_Inventory.append(addCD)
        except ValueError:
            print('That is not valid CD ID! Please enter an integer')


# -- Main Body of Script -- #
# TODO Add Code to the main body
# DONE: Load data from file into a list of CD objects on script start
# 1. When program starts, read in the currently saved Inventory
FileIO.load_inventory(strFileName, lstOfCDObjects)

# DONE: Display menu to user

```

Listing CDInventory.py – part 6

```python

# 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


    # DONE: let user exit program
    # 3.1 process exit first
    if strChoice == 'x':
        break


    # DONE: let user load inventory from file
    # 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.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.


    # DONE: let user add data to the inventory
    # 3.3 process add a CD
    elif strChoice == 'a':
        # 3.3.1 Ask user for new ID, CD Title and Artist
        intID, strTitle, strArtist = IO.get_cd_data()

        # 3.3.2 Add item to the table
        IO.add_inventory(intID, strTitle, strArtist, lstOfCDObjects)
        IO.show_inventory(lstOfCDObjects)

        continue  # start loop back at top.


    # DONE: let user save inventory to file
    # 3.4 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':
```

Listing CDInventory.py – part 7

```
        if strYesNo == 'y':
            # 3.6.2.1 save data
            # TODOne move processing code into function
            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.


    # DONE: show user current inventory
    # 3.5 process display current inventory
    elif strChoice == 'i':
        IO.show_inventory(lstOfCDObjects)
        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:
        print('General Error')
```