Matthew Poehler

March 07, 2021

Foundations of Programming: Python

Assignment 08

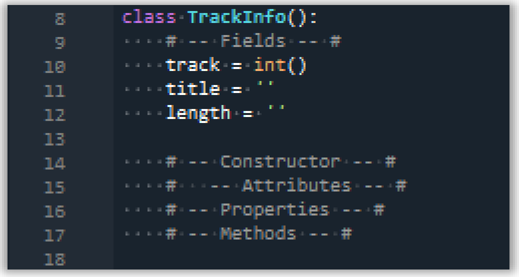
CD Inventory Script

Introduction

In this document I will demonstrate the steps taken to modify a script template created by Professor Biesinger in Module 08 of the Foundations of Programming: Python course. This script was given with a series of To-Do's and pseudocode and the objective was to add a similar CD Inventory script that has been the bases for previous assignments. This document will include brief descriptions of the aspects of object-oriented programming that was discussed in the Module 08 material such as classes and the pieces that go into constructing an object like constructors and their attributes, getters/setters, and methods. I will conclude with images of the CD Inventory script being executed in Spyder IDE and Anaconda Prompt.

Classes/Objects

According to the Module 08 material classes are the blueprints for objects. They contain the data and the functionality of an object. The material in Module 08 layout classes in multiple parts such as fields, constructors, attributes, properties and methods. Listing 1 is an example of what this looks like. All these pieces not only create objects to be used in program but give the object specific qualities that produce unique outcomes when the object is called. A special note introduced in the course material is that classes do not require all the structure parts to exist.

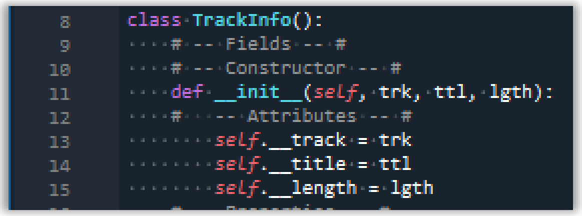


```
class TrackInfo():
    # -- Fields -- #
    track = int()
    title = ''
    length = ''

    # -- Constructor -- #
    #    -- Attributes -- #
    # -- Properties -- #
    # -- Methods -- #
```

Listing 1 - Code from LAB08-A from Module 08 showing the structure of a class

Constructors/Attributes

The fields portion was not utilized in this assignment so the focus will be on those that are such as constructors and the attributes that can be used with them. According to Module 08 material constructors are the dedicated method that is invoked in object creation. The __init__() or dunder init method is Python's constructor method. To activate an object, you just call the class's name as if it were a function. Attributes, according to the course material, are internal variables that hold data. Listing 2 demonstrates what this all looks like.

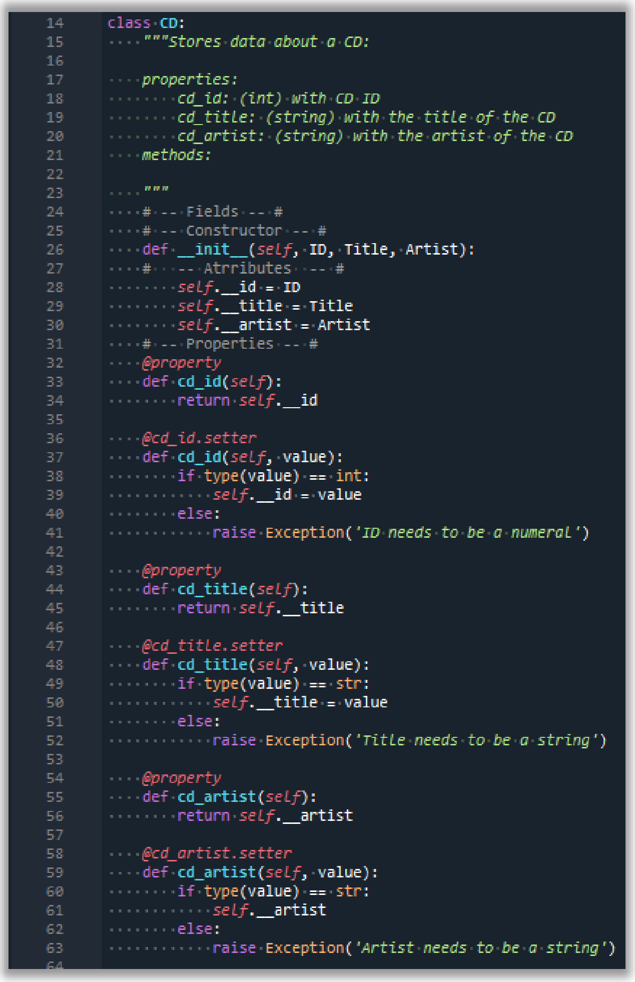


```python
class TrackInfo():
    # -- Fields -- #
    # -- Constructor -- #
    def __init__(self, trk, ttl, lgth):
    #   -- Attributes -- #
        self.__track = trk
        self.__title = ttl
        self.__length = lgth
```

Listing 2 - From LAB08-C from Module 08 showing the implementation of a constructor and attributes inside an object

Getters and Setters

A way to dictate the use of an objects attributes is to implement properties. According to the material in Module 08, properties are a form of value control that enable a programmer to make an object's attributes private and then control how a user interacts with them through built in control mechanisms. Properties are broken up into two forms, getters and setters. Getters or accessor grants access to the attribute and setters allows for a value to be assigned to the attribute if the parameters of the setter are met. Listing 3 is an example of how this may look in a script.

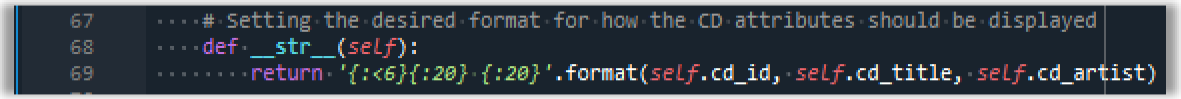


```python
class CD:
    """Stores data about a CD:

    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:

    """
    # -- Fields -- #
    # -- Constructor -- #
    def __init__(self, ID, Title, Artist):
    #   -- Atrributes  -- #
        self.__id = ID
        self.__title = Title
        self.__artist = Artist
    # -- Properties -- #
    @property
    def cd_id(self):
        return self.__id

    @cd_id.setter
    def cd_id(self, value):
        if type(value) == int:
            self.__id = value
        else:
            raise Exception('ID needs to be a numeral')

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

    @cd_title.setter
    def cd_title(self, value):
        if type(value) == str:
            self.__title = value
        else:
            raise Exception('Title needs to be a string')

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

    @cd_artist.setter
    def cd_artist(self, value):
        if type(value) == str:
            self.__artist
        else:
            raise Exception('Artist needs to be a string')
```

Listing 3 - Part of CD_Inventory.py highlighting the CD class and demonstrating the use of properties in an object

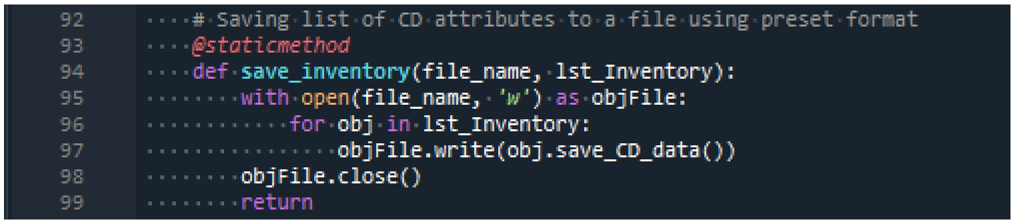
Methods

Methods are like functions in a script where they allow a programmer to organize blocks statements together and when the methods name is called the statements are invoked. Methods, however, submit a reference to the object it is called on. The course material states that this is where the “self” reference comes in. It takes the place of the first attribute supplied to a method. Listing 4, 5, and 6 are examples of methods from the CD Inventory script. Listing 4 is where the __str__() method is implemented. This converts the attributes being passed in into a string. Listing 5 is where objects get passed through and the data stored in them is saved to a file and Listing 6 is where list of objects, which is the inventory, is passed into a method to be printed to the screen.



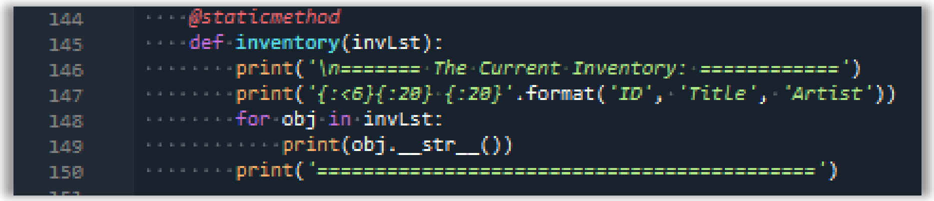
```python
    # Setting the desired format for how the CD attributes should be displayed
    def __str__(self):
        return '{:<6}{:20} {:20}'.format(self.cd_id, self.cd_title, self.cd_artist)
```

Listing 4 - Lines of script from CD_Inventory.py providing an example of a method



```python
    # Saving list of CD attributes to a file using preset format
    @staticmethod
    def save_inventory(file_name, lst_Inventory):
        with open(file_name, 'w') as objFile:
            for obj in lst_Inventory:
                objFile.write(obj.save_CD_data())
        objFile.close()
        return
```

Listing 5 - Lines from CD_Inventory.py providing an example of methods

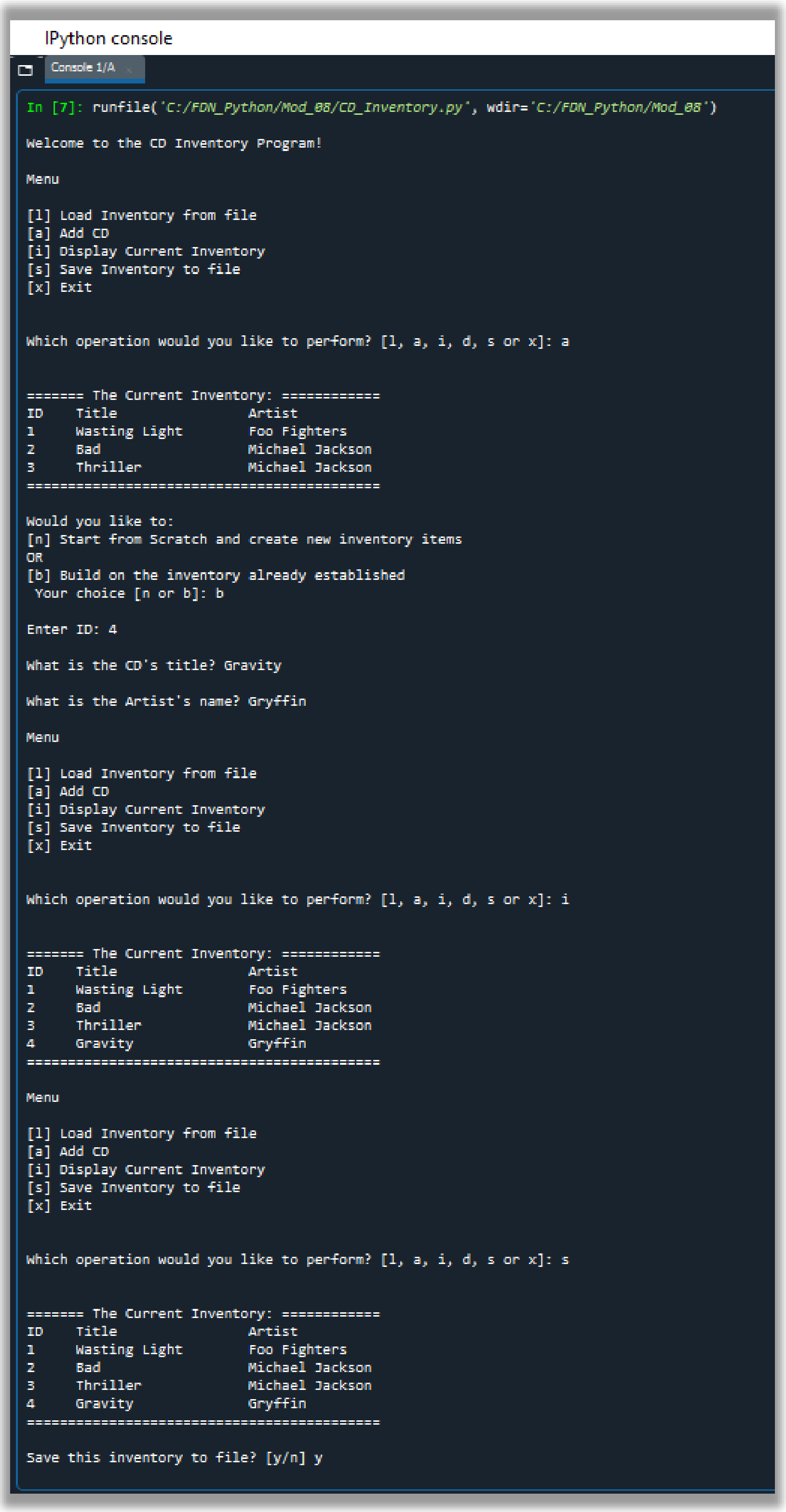


```python
    @staticmethod
    def inventory(invLst):
        print('\n======= The Current Inventory: ============')
        print('{:<6}{:20} {:20}'.format('ID', 'Title', 'Artist'))
        for obj in invLst:
            print(obj.__str__())
        print('==========================================')
```

Listing 6 - Lines from CD_Inventory.py providing an example of a method

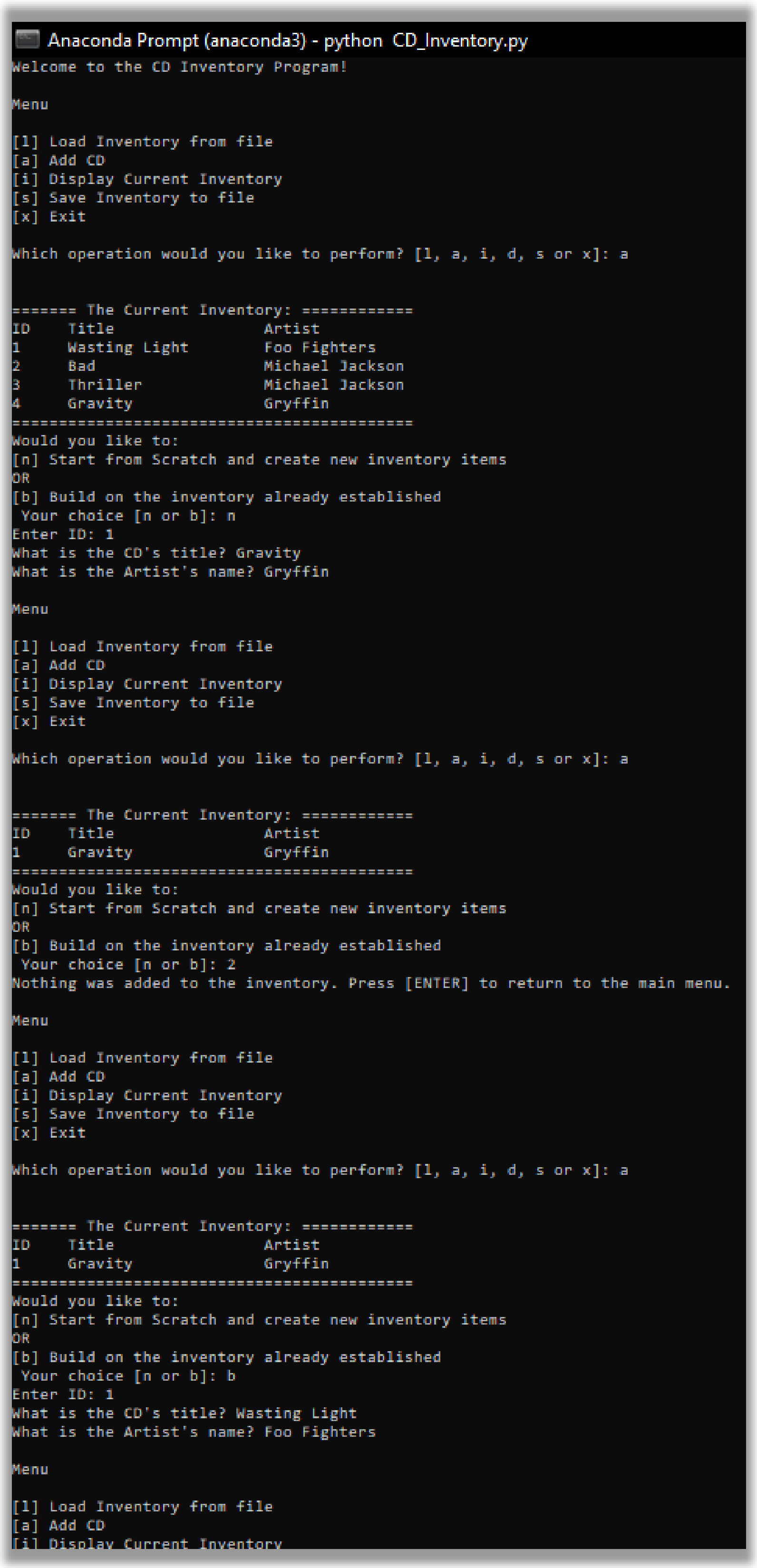
CD Inventory Script

Figures 1 and 2 show examples of the CD Inventory script being executed in an IDE and prompt screen. This script utilized the different structure pieces of an object and methods to produce the different parts of the program.



```
In [7]: runfile('C:/FDN_Python/Mod_08/CD_Inventory.py', wdir='C:/FDN_Python/Mod_08')

Welcome to the CD Inventory Program!

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, d, s or x]: a


======= The Current Inventory: ============
ID    Title                 Artist
1     Wasting Light         Foo Fighters
2     Bad                   Michael Jackson
3     Thriller              Michael Jackson
==========================================

Would you like to:
[n] Start from Scratch and create new inventory items
OR
[b] Build on the inventory already established
 Your choice [n or b]: b

Enter ID: 4

What is the CD's title? Gravity

What is the Artist's name? Gryffin

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, d, s or x]: i


======= The Current Inventory: ============
ID    Title                 Artist
1     Wasting Light         Foo Fighters
2     Bad                   Michael Jackson
3     Thriller              Michael Jackson
4     Gravity               Gryffin
==========================================

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, d, s or x]: s


======= The Current Inventory: ============
ID    Title                 Artist
1     Wasting Light         Foo Fighters
2     Bad                   Michael Jackson
3     Thriller              Michael Jackson
4     Gravity               Gryffin
==========================================

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

Figure 1 - CD_Inventory.py being executed in Spyder IDE.



Anaconda Prompt (anaconda3) - python CD_Inventory.py

```
Welcome to the CD Inventory Program!

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, d, s or x]: a


======= The Current Inventory: ============
ID    Title                 Artist
1     Wasting Light         Foo Fighters
2     Bad                   Michael Jackson
3     Thriller              Michael Jackson
4     Gravity               Gryffin
===========================================
Would you like to:
[n] Start from Scratch and create new inventory items
OR
[b] Build on the inventory already established
 Your choice [n or b]: n
Enter ID: 1
What is the CD's title? Gravity
What is the Artist's name? Gryffin

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, d, s or x]: a


======= The Current Inventory: ============
ID    Title                 Artist
1     Gravity               Gryffin
===========================================
Would you like to:
[n] Start from Scratch and create new inventory items
OR
[b] Build on the inventory already established
 Your choice [n or b]: 2
Nothing was added to the inventory. Press [ENTER] to return to the main menu.

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, d, s or x]: a


======= The Current Inventory: ============
ID    Title                 Artist
1     Gravity               Gryffin
===========================================
Would you like to:
[n] Start from Scratch and create new inventory items
OR
[b] Build on the inventory already established
 Your choice [n or b]: b
Enter ID: 1
What is the CD's title? Wasting Light
What is the Artist's name? Foo Fighters

Menu

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
```

Figure 2 - CD_Inventory.py being executed in Anaconda Prompt

Summary

The objective of this assignment was to create a program that utilized object-oriented programming to collect data from a user to construct a simple inventory that contains titles and artists of musical CDs. This required the creation of a CD object that contained attributes such as ID numbers, titles and artist names. That information is then available to view, save into a text file or be re-loaded from a text file. This program was created using the material learned from Module 08. I look forward to expanding on object-oriented programming as this course continues.