

Classes & The CD Inventory Program Continued

Introduction

In this assignment, I will discuss what classes are and a few essential components of a class. Then, I will discuss the steps I took to modify the CD_Inventory.py program to add a class for creating a CD object.

Classes

In Python, classes are like a blueprint for creating an instance of an object.¹ An object is an abstraction for data, including the properties of the data as well as behaviors applicable to it.² For example, you could create a class that would create a “guitar” object, where it could have properties like ‘guitar strings’, ‘tuning keys’, and ‘body type’, and methods like ‘strum’ and ‘tune’. You could then use this class to instantiate a guitar object, and you could define its individual properties and then ‘strum’ the guitar or ‘tune’ the guitar through the methods available to the object.

The CD Inventory Program Continued

For this assignment, I added code to create a ‘CD’ class that stores information about a CD. This class has three attributes: a cd_id, a cd_title, and a cd_artist. The class allows for accessing and setting each attribute, and there is data validation when setting an attribute. Once I instantiate an object of the CD class, I can then interact with the instance object by setting values of the attributes, like so:

¹ <https://docs.python.org/3/tutorial/classes.html>, retrieved 2020-09-02.

² <https://docs.python.org/3/reference/datamodel.html>, retrieved 2020-09-02.

```
# Instantiate a CD object
cd_data = CD()

# Set properties of the instance object
cd_data.cd_id = 1
cd_data.cd_title = 'Help!'
cd_data.cd_artist = 'The Beatles'
```

Figure 1 - Instantiating Object

Now that I have set those values, I can also retrieve them by their attribute names:

```
print(cd_data.cd_id)
print(cd_data.cd_title)
print(cd_data.cd_artist)
```

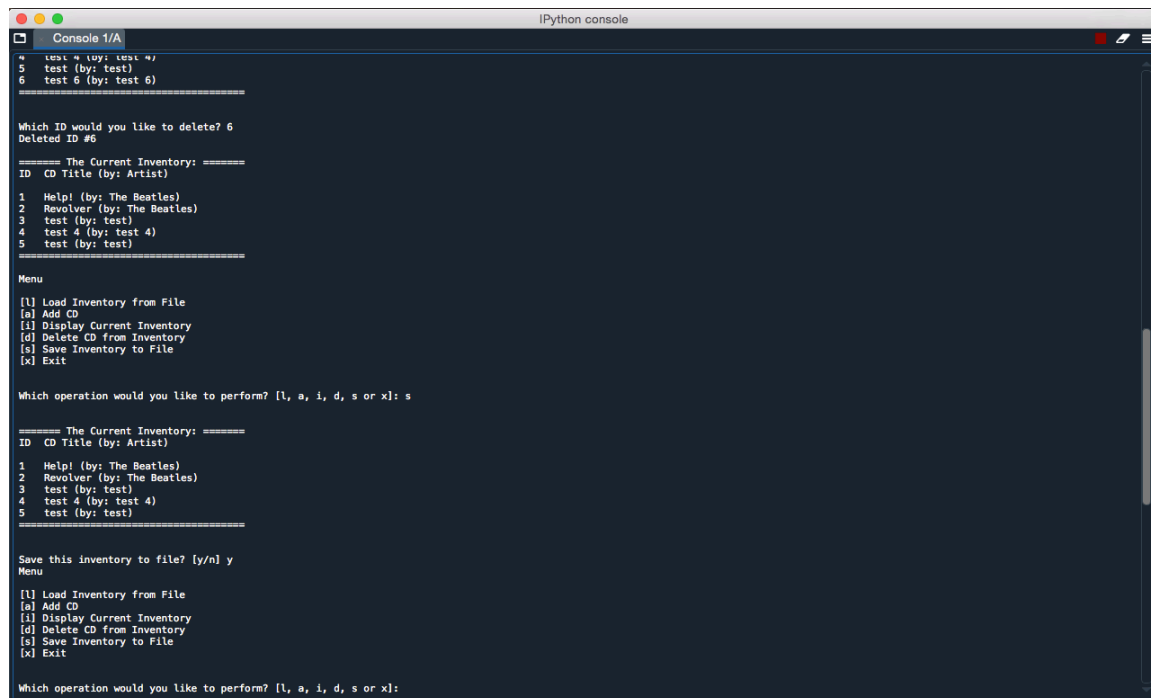
Figure 2 - Accessing Instance Object Attributes

Because these attributes belong to the instance object, it makes passing all of the CD data around in the program a little bit easier, because now I no longer have to pass three individual variables around in the program and to various functions—the `cd_id`, `cd_title`, and `cd_artist`—because I can simply pass the entire `cd_data` instance object into a function, and then within that function I can access the attributes of the object. For example, I was able to update my `append_row_to_table()` function from requiring four arguments to only requiring two, since I collapsed the `cd_id`, `cd_title`, and `cd_artist` arguments into a single argument—a `cd_data` instance object that contains each of those values via attributes.

```
...@staticmethod
...def append_row_to_table(cd_object, list_of_cds):
...    """Function to append a dictionary to a list
...
...    Args:
...        -- cd_object ():
...        -- list_of_cds (list): A list of dicts that represents a collection of rows of CD data
...
...    Returns:
...        -- None.
...    """
...    dict_row = {'id': cd_object.cd_id, 'title': cd_object.cd_title, 'artist': cd_object.cd_artist}
...    list_of_cds.append(dict_row)
```

Figure 3 - Updated Function with Instance Object

Here is the program running in Spyder:

A screenshot of the Spyder Python IDE interface. The main window shows the execution of a Python script. The output includes a list of test items, a prompt to delete an item (ID 6), a display of the current inventory, a menu of operations, and a prompt to save the inventory to a file. The user has entered 's' for save, and the program has prompted for confirmation to save to a file.

```
test 4 (by: test 4)
test (by: test)
test 6 (by: test 6)

Which ID would you like to delete? 6
Deleted ID #6

===== The Current Inventory: =====
ID  CD Title (by: Artist)
1  Help! (by: The Beatles)
2  Revolver (by: The Beatles)
3  test (by: test)
4  test 4 (by: test 4)
5  test (by: test)

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]: s

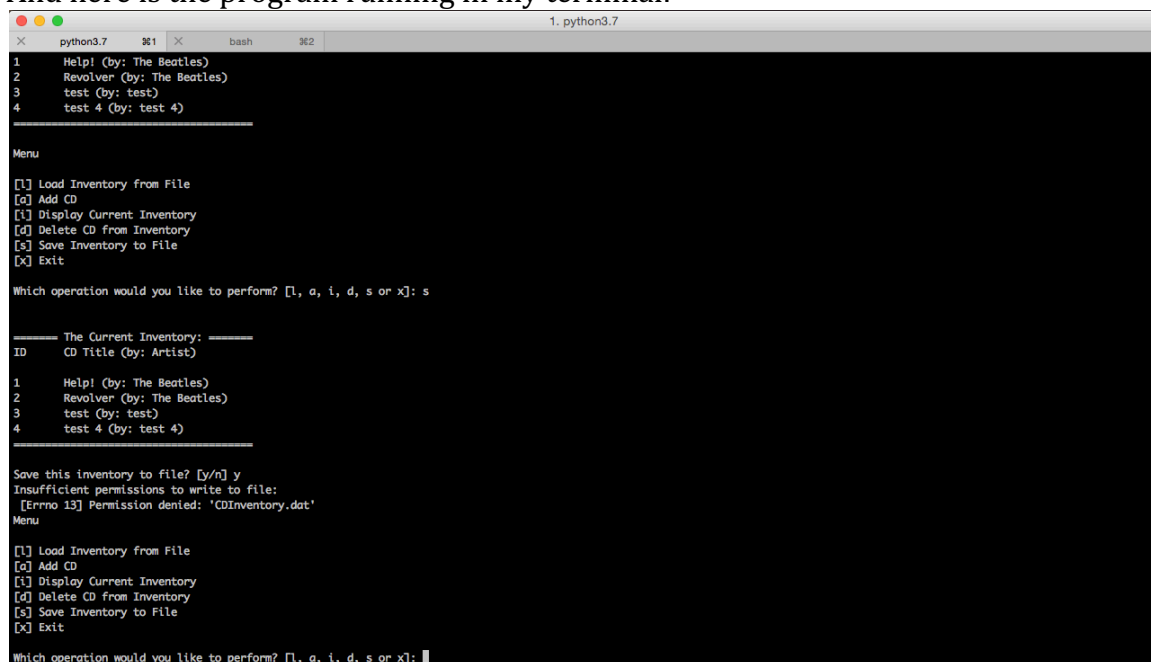
===== The Current Inventory: =====
ID  CD Title (by: Artist)
1  Help! (by: The Beatles)
2  Revolver (by: The Beatles)
3  test (by: test)
4  test 4 (by: test 4)
5  test (by: test)

Save this inventory to file? [y/n] y
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 4 - Running CD_Inventory.py in Spyder

And here is the program running in my terminal:

A screenshot of a terminal window running the CD_Inventory.py program. The output is identical to the Spyder IDE screenshot, but it includes an error message when the user attempts to save the inventory to a file: 'Insufficient permissions to write to file: [Errno 13] Permission denied: 'CDInventory.dat''.

```
1  Help! (by: The Beatles)
2  Revolver (by: The Beatles)
3  test (by: test)
4  test 4 (by: test 4)

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]: s

===== The Current Inventory: =====
ID  CD Title (by: Artist)
1  Help! (by: The Beatles)
2  Revolver (by: The Beatles)
3  test (by: test)
4  test 4 (by: test 4)

Save this inventory to file? [y/n] y
Insufficient permissions to write to file:
[Errno 13] Permission denied: 'CDInventory.dat'
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 5 - Running CD_Inventory.py in Terminal

Summary

For this assignment, I talked about classes in Python. Then, I went over some of the updates I made to the CD_Inventory.py program to utilize a new CD class. Here is the

link to my Github repository for Assignment_08:
https://github.com/palmermbandy/Assignment_08.