Palmer Bandy
August 19, 2020
IT FDN 110 B Su 20
Assignment 06

Functions, Classes, & The CD Inventory Program Continued

Introduction

In this assignment, I will discuss what a function is and how they help to organize code. Additionally, I will go over the basics of what a class is in Python, and how it relates to functions. Then, I will talk about how I utilized functions and classes to modify the CDInventory.py program by organizing it according to the principle of the "Separation of Concerns".

Functions

In Python, a "function" is a way of grouping together a set of statements (a block of code) under a name defined by the person writing the code.[1] To create a function, you first define the name of the function using the keyword `def`, followed by the function name, a set of parentheses, and a colon, like so:

```
def my_new_function():
    print('This is a function')
```

Figure 1 - Creating a Function

In Figure 1, `my_new_function` is the name of the function. In order to use a function, you need to "call" it somewhere in the program (after you have first defined it, so somewhere below where the function has been defined), and you call it by its name. To call this function, then, you would include a line like `my_new_function()`, which would execute the code in the function from the place where you called the function.[2]

Functions can take inputs and do something with them, which is how they are often used. These inputs are called "arguments", and you include the arguments that a function requires within the parentheses that follow the function name. For

[1] https://www.learnpython.org/en/Functions, retrieved 2020-08-19
[2] https://www.tutorialspoint.com/python/python_functions.htm, retrieved 2020-08-19

example, `my_new_function(arg1, arg2)` takes two arguments, and within the block of code that the function executes, you could use these two arguments like variables.[3]

Classes

A "class" in Python is a way to group together "functions, variables, and constants."[4] Similarly to functions, classes help you organize your code. To create a class, you use the `class` keyword, followed by the name of your class. For example, this creates a new class named `MyNewClass`, under which is placed `my_new_function()`:

```
class MyNewClass:

    @staticmethod
    def my_new_function():
        print('This is a function')

```

Figure 2 - Creating MyNewClass

In order to call a function that is part of a class, you prepend the class name onto the function, separated by a period, like so: `MyNewClass.my_new_function()`. In this way, you could theoretically have differently named classes that have all the same function names inside of them, but the functions would be unique based on the fact that they are part of different classes. So calling `MyOldClass.my_new_function()` would be different than calling `MyNewClass.my_new_function()`, even though both functions are of the same name.[5]

The CD Inventory Program

This assignment's task was primarily to modify the starter file to make use of functions and classes. I started out by adding a function named `append_row_to_table` that takes two arguments—the 2D table that is a list of dictionaries, and an individual dictionary row—and simply appends the dictionary row to the list. Because this is a data processing task, I placed this function under the `DataProcessor` class that was already set up in the starter file:

---

[3] Ibid.

[4] FDN_Py_Module_06.pdf, 21, retrieved 2020-08-19

[5] Ibid., 22

```python
class DataProcessor:

    @staticmethod
    def append_row_to_table(lstTbl, dictRow):
        """Function to append a dictionary to a list

        Args:
            - lstTbl (list): A list of dicts that represents a collection of rows of CD data
            - dictRow (dictionary): A dictionary that represents a single row of CD data

        Returns:
            - None.
        """
        lstTbl.append(dictRow)
```

Figure 3 - append_row_to_table() Function

Once I had this function defined and working, I added the docstring to explain what it does, the arguments that it takes, and what it returns.

Then I took the code that was provided for deleting a row and placed it into its own function named `delete_row_from_table`. This function also takes two arguments—the lstTable and then a single ID that represents the ID of a row to delete—and deletes that row from the in-memory table if the ID exists. I then simplified the code a little to enumerate the list and delete an entry based on the enumeration counter if it finds an ID to delete.

Next, I added code to the `write_file` function to actually write data to the file from the `lstTbl`. This function reads in data from the `lstTbl`, which is a list of dictionaries, loops through each dictionary in the list, and constructs a single comma-separated string for each dictionary, and then it writes that string to the specified file.

In order to gather user input for adding new CDs, I created a new function named `ask_user_for_input` that takes one argument—a CD ID—and asks the user to input two other strings—a CD Title and Artist. The argument that this function takes is for a CD ID, and right now you could technically pass any string value into this, but it is meant to be used with the `DataProcessor.generate_new_id()` function that I also created, which generates a new CD ID that is unique as long as it has consistently been used to generate the IDs stored in the data file or in-memory table.

Here is the program running in Spyder:

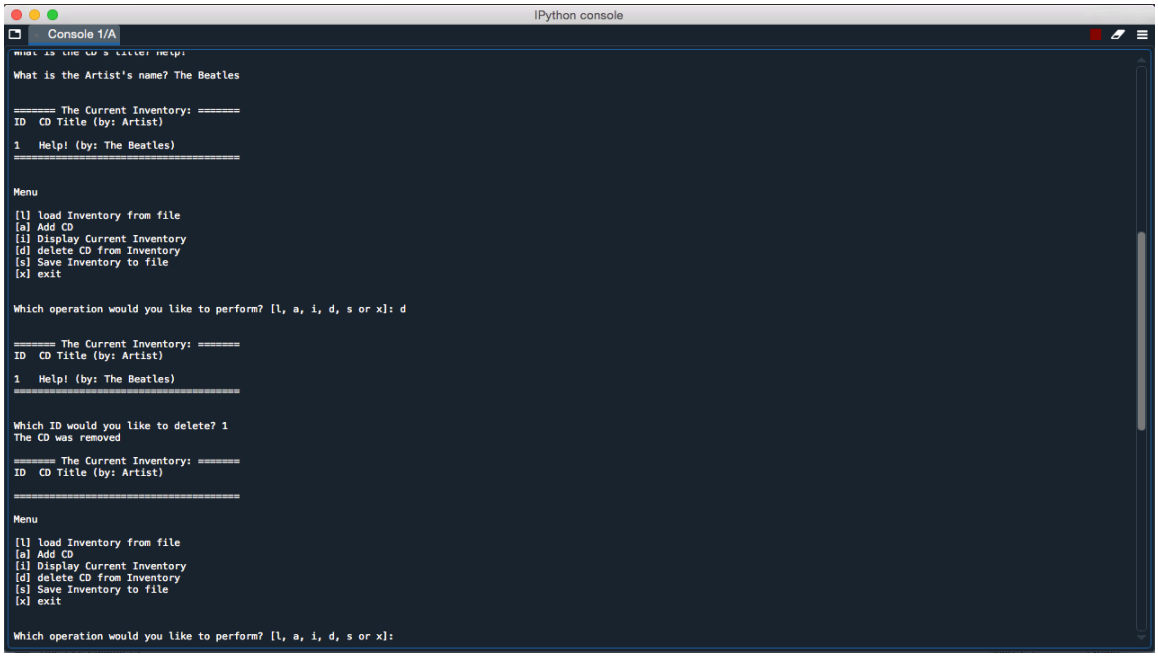

Figure 4 - Running CDInventory.py in Spyder

And here is the program running in my terminal:

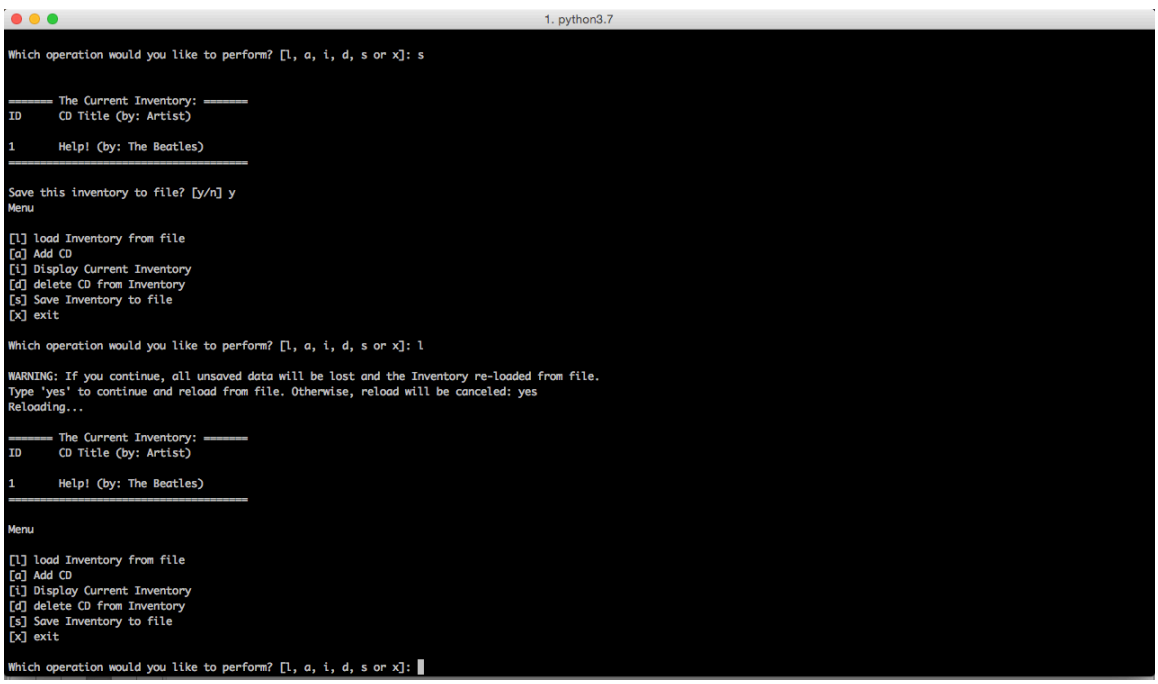

Figure 5 - Running CDInventory.py in my Terminal

Summary

For this assignment, I discussed what functions and classes are in Python, and how they are useful for organizing code. Then, I walked through the steps that I took to

modify the starter code in order to utilize functions and classes for organizing my code into manageable blocks. Here is the link to my Github repository for Assignment_06: https://github.com/palmermbandy/Assignment_06.