

# Classes and Functions

## Introduction

Assignment 6 builds upon previous weeks modules and introduces Classes. Classes are where multiple functions can be grouped. The start script separated the Classes to be used into Data Processing, File Processing, and IO (Input/Output).

## Pseudo Code

Pseudo code was provided by default when unzipping Module 6, which organized the different code blocks. #TODO keywords were spread throughout the starter script to show areas of the code that needed to be changed.

I added my own custom pseudo code where I wanted to make notes to myself and flag an important change.

## Initial Approach

I decided to work through the practice labs to gain some more knowledge after looking through the Assignment 6 starter script. I gathered my notes from Tuesday and Friday Zoom sessions after class questions were discussed to hash out the subject matter further.

## TODOs and Functions

I worked each TODO from top to bottom, using my own comments as place holders to remember where in the script I needed to go back to after creating new function.

It was interesting to see that each function required a slightly different construct. Some functions didn't require any arguments or return statements, while others needed a combination of these.

## A Conversation with the Spyder Console

I had a back and forth with the Spyder Console. I update code, run it, and received feedback through the console. I changed the code until I received the desired outcomes for the areas I was testing.

The script errors were hit at each step of the way when testing the program in runtime. The most valuable thing during smoke testing was that Spyder flagged which lines and the type of errors it encountered. This helped me zero in what part of the script needed to be fixed next.

The very first error was hit because the text file that the script was trying to read did not exist. The console showed in the code to look, which was the read\_file method of the FileProcessor class, Figure 1. Here the 'r' argument highlighted in yellow, only reads the file if it already exists.

```

File "/Users/captainscomm/pythonclass/module6/Mod_06/Assignment06_Starter.py",
line 110, in <module>
    FileProcessor.read_file(strFileName, lstTbl)

File "/Users/captainscomm/pythonclass/module6/Mod_06/Assignment06_Starter.py",
line 40, in read_file
    objFile = open(file_name, 'r')

FileNotFoundError: [Errno 2] No such file or directory: 'CDInventory.txt'

```

Figure 1 - No file error.

To resolve no file error issue for first time users who start the program, I added the 'a+' argument to line 95, Figure 2. This creates a file if one doesn't exist and places the file pointer at the end of the file. In line 96, I close the file.

Then the function proceeds to line 98 where it can begin the process of clearing the table of data and opening the file in read mode to load data from the beginning of the text file if any data exists from preexisting user history.

I retested the read\_function and had no issue with loading data from existing text files, or with first time user scenarios where a text file has not yet been created. No more errors came up again with this function once I updated the code as shown in Figure 2, lines 95-104.

```

95         objFile = open(strFileName, 'a+') #creates text file if it doesn't exist
96         objFile.close()
97
98         table.clear() # this clears existing data and allows to load data from file
99         objFile = open(file_name, 'r') #opens text file for reading
100         for line in objFile:
101             data = line.strip().split(',')
102             dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
103             table.append(dicRow)
104         objFile.close()

```

Figure 2 - read function.

## Function Dependencies

I immediately encountered an interesting dependency between two functions: get\_cd\_input (an IO Class function), and add\_cd (a DataProcess Class function). One function gathers input from the user, and the other function relies on that user input to save a new CD entry into memory.

I was able to gather three user inputs from get\_cd\_input and return these three inputs in the form of a tuple. I wasn't able to get the add\_cd function to recognize the tuple. I found a forum post online that talks about unpacking tuples into separate variables, which was a topic also covered in last Friday's Q&A Zoom session with our instructor team. This jogged my memory and I unpacked the tuple from the get\_cd\_input() in the main body of the code into three variables.

Line 231 of my code, Figure 3, shows how I called the get\_cd\_input() function, while unpacking the tuple into three separate variables: strID, strTitle, and strArtist.

```

227 elif strChoice == 'a':
228     # 3.3.1 Ask user for new ID, CD Title and Artist
229     # DONE: move IO code into function
230     # unpack tuple, variables to be access by add_cd method
231     strID, strTitle, strArtist = I0.get_cd_input()
232     # 3.3.2 Add item to the table
233     # DONE: move processing code into function
234     DataProcessor.add_cd(strID, strTitle, strArtist)
235     I0.show_inventory(lstTbl)

```

Figure 3 - unpacking a tuple from a function.

The next issue was to resolve the add\_cd() function error, Spyder kept complaining that a variable in the function was not declared or defined. That's when I saw that I hadn't created any arguments for add\_cd() to receive the variables from the unpacked tuple in line 231.

I updated add\_cd() with three arguments: val1, val2, val3, see line 33, Figure 4. Once this was done, no more undeclared variable errors were flagged by Spyder.

```

33 def add_cd(val1, val2, val3):
34     """Add CD
35         Accepts 3 arguments that are recorded into a dictionary row
36         and saved into memory, a 2D table.
37
38     Args:
39         val1: used for ID value.
40         val2: used for Title.
41         val3: used for Artist.
42
43     Returns:
44         None.
45
46     """
47     intID = int(val1)
48     dicRow = {'ID': intID, 'Title': val2, 'Artist': val3}
49     lstTbl.append(dicRow)

```

Figure 4 - Receive data in functions with arguments.

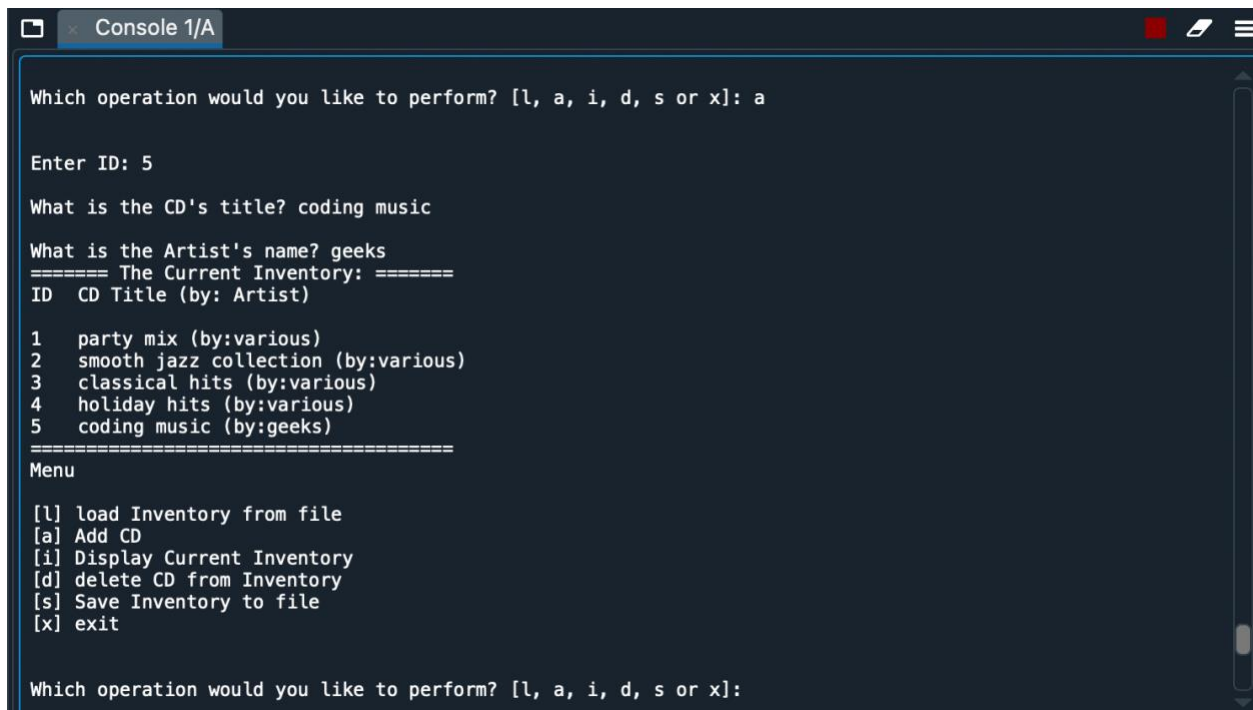
## Summary

My biggest hiccups with working on this assignment are mentioned in the main body of this knowledge document. It took me several hours of working through the errors and review of previous lessons and class sessions to find a solution to the issues that were presented. I saved updating the doc strings as the very last part of my workflow because working on all the errors in script execution were my priority. It felt great to be able to resolve all the issues using my notes and discussion topics from this past week as my foundation. The week 6 assignment was like working on a puzzle, one piece at a time, until finding the right fit.

## Appendix

The following pages in this appendix show screen captures of:

- Figure 5, Spyder Console code execution.
- Figure 6, Mac Terminal code execution.
- Github, Assignment 6 Repository: [https://github.com/jamescisonline/Assignment\\_06](https://github.com/jamescisonline/Assignment_06)



```
Console 1/A

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

Enter ID: 5

What is the CD's title? coding music

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

1  party mix (by:various)
2  smooth jazz collection (by:various)
3  classical hits (by:various)
4  holiday hits (by:various)
5  coding music (by:geeks)
=====
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- Spyder console runtime.

```
Assignment06 — Python CDInventory.py — 84x23

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

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

1       party mix (by:various)
2       smooth jazz collection (by:various)
4       holiday hits (by:various)
5       coding music (by:geeks)
6       meow meows (by:the cat)
=====
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 6 - Mac Terminal runtime.