Amanda Larson

November 21, 2021

IT FDN 110 – Foundations of Programming: Python

Assignment 06

Module 06 – Assignment 06: Working with classes and functions

Introduction

Assignment 06 in the foundations of python course has us working with classes and functions. We started with a "starter script" and the task was to move blocks of code from the main body of the script into functions thereby organizing the program using the design principle: SoC – Separation of Concerns. We were reminded to not overlook the docstrings for each new function we created. This paper will detail how I completed the assignment, what went well and what didn't go as well.

What went well

Overall, this assignment was a bit less complicated than the last few assignments. And it took about 3-4 hours less time than I spent on other assignments. The idea of moving blocks of code from one location in the program to another was simple enough: cut and paste. Remembering to format the function properly with docstrings and also remembering to replace the code block you cut by a new line of code which calls that function were fundamental to this assignment.

There were 3 or 4 different areas of code with #TODOs which identified them as code blocks that should be moved into functions. Each block of code that was moved was moved into a class. Each class corresponded with the SoC principle. There was an I/O class, a File Processing Class and a Data Processing class. Its important to know which class you created your new function in so that you can call it properly in the main body.

I had no problem understanding the goal of the assignment and it took very little time to actually move the code blocks up into functions and set them up with docstrings. For some reason I knew intuitively how to set up the first few with the correct arguments and had no errors at all. It wasn't until I started working with the get_CD() function and add_CD() function that things started to go sideways.

What didn't go well

On Saturday, I worked on moving all the code into functions. Laurel and Thomas and I met on a zoom call to discuss the best way to move forward and we remembered that Laura had mentioned it was easier to start from the bottom of the main body and work your way up. This was a great tip and worked well for us. Laurel and I were able to move all of our code blocks into functions fairly quickly but both were having errors arise while setting up the I/O function of get_CD() which gets the users input and the Data Processing function of add_CD() which adds the users data to the data table. The error was that the variable wasn't defined but then when we defined it, whether inside or outside of the while loop, or function it didn't like that it was set to an empty space ''. I tried changing it to None and 0 and moving them in and out of the function but it still didn't like it.

Luckily, other students were having similar issues and when I posted on the discussion forum James was able to help me out. On Sunday, when I started back at troubleshooting the errors, I worked with Andrea to solve the problem. Part of the problem was that we didn't have a return statement in the get_CD() function. Another issue was that we weren't sure how to get the results/return from the get_CD() function into the add_CD() function. I did some research and tried nesting them and passing one function as an argument to the other function but eventually was just creating more

errors than I was solving. After seeing James' post is when it occurred to me to unpack AND call the function at the same time so that I could pass those arguments into the next function.

```
elif strChoice == 'a':
    # 3.3.1 Ask user for new ID, CD Title and Artist
    strID, strTitle, stArtist = IO.get_CD() #unpack and call at the same time
    # 3.3.2 Add item to the table
    DataProcessor.add_CD(strID, strTitle, stArtist)
    IO.show_inventory(lstTbl)
    continue  # start loop back at top.
```

Figure 1 - Displays how to unpack the return and call the function at the same time.

This worked well, but I then had to go back and adjust the number of arguments for both functions and make sure that everything made sense. Once I had figured this step out, my code ran perfectly and I was able to both get and save the new CDs from the user input. The other functions worked well already, so I was done! I tried the program in both Spyder and in Anaconda prompt.

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


Enter ID: 4

What is the CD's title? Lover

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

1   Bad (by:Michael)
3   Folklore (by:Taylor)
2   Rep (by:Tay)
4   Lover (by:Taylor)
======================================
```

Figure 2- Showing the program adding a CD using Spyder.

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

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

1   Bad (by:Michael)
3   Folklore (by:Taylor)
4   Lover (by:Taylor)
======================================

Which ID would you like to delete? 4
The CD was removed
======= The Current Inventory: =======
ID  CD Title (by: Artist)

1   Bad (by:Michael)
3   Folklore (by:Taylor)
======================================
```

Figure 3- Showing the program deleting a CD from the inventory using Spyder.

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

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

1   Bad (by:Michael)
3   Folklore (by:Taylor)
2   Rep (by:Tay)
======================================

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
```

Figure 4 - Showing the program saving to file using Spyder.

```
(base) C:\Users\alarson>cd C:\PythonFundementals\Mod_06\Assignment06

(base) C:\PythonFundementals\Mod_06\Assignment06>python  Assignment06.py
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]: 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 canceledyes
reloading...
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Bad (by:Michael)
3       Folklore (by:Taylor)
2       Rep (by:Tay)
======================================
```

Figure 5 - Showing the program loading data in from the text file using Anaconda prompt.

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

Enter ID: 4
What is the CD's title? Lover
What is the Artist's name? Taylor
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Bad (by:Michael)
3       Folklore (by:Taylor)
2       Rep (by:Tay)
4       Lover (by:Taylor)
======================================
```

Figure 6 - Showing adding a new CD to the inventory using Anaconda prompt.

Summary

I enjoyed learning some basic design principles which will help me to create more readable and less repetitive programs. I liked that this assignment was a bit easier but still had a tricky element to it. Its more than just rearranging – you have to make sure that it makes sense and that the functions are able to return what you need. I am finally able to successfully run my programs in anaconda with no errors or in the wrong version of python on the first try and that feels

good. I think I'm still fuzzy on the purpose of unpacking and why that was the solution for connecting the two functions. I also am not sure why the read function used file_name and table as variables and the write function used strFileName and lstTbl. I would think they would need to be the same in order to refer to the same text file. Also file_name was never defined. Not sure how that works or why.

Appendix

I used a site called (Highlight your source code, 2021)[1] to format the code into the colored version. See below:

```
#------------------------------------------#
# Title: Assignment06.py
# Desc: Working with classes and functions.
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# ALarson, 2021-Nov-20, Moved code blocks into functions and created doctext
# ALarson, 2021-Nov-21, Troubleshooted new code errors after moving into functions
#------------------------------------------#

# -- DATA -- #
strChoice = '' # User input
lstTbl = []  # list of lists to hold data
dicRow = {}  # list of data row
strFileName = 'CDInventory.txt'  # data storage file
objFile = None  # file object
strTitle = ''
stArtist = ''
lstValues = []

# -- PROCESSING -- #
class DataProcessor:
    """The DataProcessor class processes the user data."""
    @staticmethod
    def add_CD(strID, strTitle, stArtist):
        """Function to add a new user-defined CD to the table in-memory.

        Args:
            intID: the CD ID number as provided by the user
            strTitle: the CD title as provided by the user
            stArtist: the CD artist as provided by the user

        Returns:
            None.
        """

        intID = int(strID)
        dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}
        lstTbl.append(dicRow)
```

[1] https://highlight.hohli.com/index.php

```
39.
40.
41.     @staticmethod
42.     def delete_Row(intIDDel):
43.         """ Function to delete row in table.
44.
45.         Args:
46.             intIDDel: the ID of the CD to delete
47.
48.         Returns:
49.             None.
50.
51.         """
52.         intRowNr = -1
53.         blnCDRemoved = False
54.         for row in lstTbl:
55.             intRowNr += 1
56.             if row['ID'] == intIDDel:
57.                 del lstTbl[intRowNr]
58.                 blnCDRemoved = True
59.                 break
60.         if blnCDRemoved:
61.             print('The CD was removed')
62.         else:
63.             print('Could not find this CD!')
64.
65.
66. class FileProcessor:
67.     """Processing the data to and from text file"""
68.
69.     @staticmethod
70.     def read_file(file_name, table):
71.         """Function to manage data ingestion from file to a list of dictionaries
72.
73.         Reads the data from file identified by file_name into a 2D table
74.         (list of dicts) table one line in the file represents one dictionary row in table.
75.
76.         Args:
77.             file_name (string): name of file used to read the data from
78.             table (list of dict): 2D data structure (list of dicts) that holds the data
    during runtime
79.
80.         Returns:
81.             None.
82.         """
83.         table.clear()  # this clears existing data and allows to load data from file
84.         objFile = open(file_name, 'r')
85.         for line in objFile:
86.             data = line.strip().split(',')
87.             dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
```

```
            table.append(dicRow)
        objFile.close()


    @staticmethod
    def write_file(file_name, table):
        """Function to manage the saving of data to a textfile.

        Args:
            file_name (string): name of file to be saved, presumably the same as has already been loaded in.
            table (list of dictionaries): 2D data structure (list of dicts) that holds the data during runtime

                Returns:
                    None.
                    """
                # DONE Add code here
                objFile = open(strFileName, 'w')
                for row in lstTbl:
                    lstValues = list(row.values())
                    lstValues[0] = str(lstValues[0])
                    objFile.write(','.join(lstValues) + '\n')
                objFile.close()



    # -- PRESENTATION (Input/Output) -- #

    class IO:
        """Handling Input / Output"""

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

            Args:
                None.

            Returns:
                None.
            """

            print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
            print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')

        @staticmethod
        def menu_choice():
            """Gets user input for menu selection
```

```
135.
136.             Args:
137.                 None.
138.
139.             Returns:
140.                 choice (string): a lower case sting of the users input out of the choices
    l, a, i, d, s or x
141.
142.             """
143.             choice = ' '
144.             while choice not in ['l', 'a', 'i', 'd', 's', 'x']:
145.                 choice = input('Which operation would you like to perform? [l, a, i, d, s
    or x]: ').lower().strip()
146.             print()  # Add extra space for layout
147.             return choice
148.
149.         @staticmethod
150.         def show_inventory(table):
151.             """Displays current inventory table
152.
153.
154.             Args:
155.                 table (list of dict): 2D data structure (list of dicts) that holds the
    data during runtime.
156.
157.             Returns:
158.                 None.
159.
160.             """
161.             print('======= The Current Inventory: =======')
162.             print('ID\tCD Title (by: Artist)\n')
163.             for row in table:
164.                 print('{}\t{} (by:{})'.format(*row.values()))
165.             print('=======================================')
166.
167.
168.         @staticmethod
169.         def get_CD():
170.             """Function to get user input/information for the CD ID number, the CD title,
    and the CD artist.
171.
172.             Args: None
173.
174.             Returns:
175.                 a tuple of the user's input: strID, strTitle, stArtist
176.
177.             """
178.             strID = input('Enter ID: ').strip()
179.             strTitle = input('What is the CD\'s title? ').strip()
180.             stArtist = input('What is the Artist\'s name? ').strip()
```

```
181.             return strID, strTitle, stArtist #returns a tuple
182.
183.     # 1. When program starts, read in the currently saved Inventory
184.     FileProcessor.read_file(strFileName, lstTbl)
185.
186.     # 2. start main loop
187.     while True:
188.         # 2.1 Display Menu to user and get choice
189.         IO.print_menu()
190.         strChoice = IO.menu_choice()
191.
192.         # 3. Process menu selection
193.         # 3.1 process exit first
194.         if strChoice == 'x':
195.             break
196.         # 3.2 process load inventory
197.         if strChoice == 'l':
198.             print('WARNING: If you continue, all unsaved data will be lost and the
    Inventory re-loaded from file.')
199.             strYesNo = input('type \'yes\' to continue and reload from file. otherwise
    reload will be canceled')
200.             if strYesNo.lower() == 'yes':
201.                 print('reloading...')
202.                 FileProcessor.read_file(strFileName, lstTbl)
203.                 IO.show_inventory(lstTbl)
204.             else:
205.                 input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue
    to the menu.')
206.                 IO.show_inventory(lstTbl)
207.             continue  # start loop back at top.
208.         # 3.3 process add a CD
209.         elif strChoice == 'a':
210.             # 3.3.1 Ask user for new ID, CD Title and Artist
211.             strID, strTitle, stArtist = IO.get_CD() #unpack and call at the same time
212.             # 3.3.2 Add item to the table
213.             DataProcessor.add_CD(strID, strTitle, stArtist)
214.             IO.show_inventory(lstTbl)
215.             continue  # start loop back at top.
216.         # 3.4 process display current inventory
217.         elif strChoice == 'i':
218.             IO.show_inventory(lstTbl)
219.             continue  # start loop back at top.
220.         # 3.5 process delete a CD
221.         elif strChoice == 'd':
222.             # 3.5.1 get Userinput for which CD to delete
223.             # 3.5.1.1 display Inventory to user
224.             IO.show_inventory(lstTbl)
225.             # 3.5.1.2 ask user which ID to remove
226.             intIDDel = int(input('Which ID would you like to delete? ').strip())
227.             # 3.5.2 search thru table and delete CD
```

```
            blnCDRemoved = False
            DataProcessor.delete_Row(intIDDel)
            IO.show_inventory(lstTbl)
            continue  # start loop back at top.
        # 3.6 process save inventory to file
        elif strChoice == 's':
            # 3.6.1 Display current inventory and ask user for confirmation to save
            IO.show_inventory(lstTbl)
            strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
            # 3.6.2 Process choice
            if strYesNo == 'y':
                FileProcessor.write_file(strFileName, lstTbl)
            else:
                input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
            continue  # start loop back at top.
        # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
        else:
            print('General Error')
```