Josephine Gnanaraj
02\24\2021
Foundations Of Programming: Python
Assignment_06

Introduction

This week's module we learnt working with functions. We also got introduced to Variable Scope and DocString. Learnt about global and local variables, as well as classes.

Working with functions

Information can be passed into functions as arguments. Arguments are specified after the function name, inside the parentheses. we can add as many arguments as you want, just separate them with a comma. Functions are a way of grouping statements and making this group available via a programmer defined name. In Python, the function must be defined before the call of the function. Calling the function executes the code in the function, returns the results and continues at the calling point.

Functions allow the option to pass in parameters. These allow you to pass in values for processing. Generally, Parameters are called arguments.

The return values can either be consumed instantly (as in a print statement or more generally as an Expression) or assigned to a variable.

Returning Tuples

Functions can return tuples as return values. This is very useful — we often want to know some batsman's highest and lowest score, or we want to find the mean and the standard deviation, or we want to know the year, the month, and the day, or if we're doing some ecological modeling we may want to know the number of rabbits and the number of wolves on an island at a given time. In each case, a function (which can only return a single value), can create a single tuple holding multiple elements.

Global and Local Variables

Global variables are the one that are defined and declared outside a function and we need to use them inside a function. They can also be used by functions if those functions have the "global" keyword for those variables. Local variables are local only to the function. They don't have any meaning to code outside of the function. Usually some arguments are passed to functions for processing, and there could be other variables inside the functions that aid the processing, but those variables are not defined or used outside the functions.

Classes

Classes provide a means of bundling data and functionality together. Creating a new class creates a new *type* of object, allowing new *instances* of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by its class) for modifying its state. Python classes provide all the standard features of Object Oriented Programming: the class inheritance mechanism allows multiple base classes, a derived class can override any methods of its base class or

classes, and a method can call the method of a base class with the same name. Objects can contain arbitrary amounts and kinds of data. As is true for modules, classes partake of the dynamic nature of Python: they are created at runtime and can be modified further after creation.

Summary

In this assignment\module we worked with functions and introduced Classes. We also explored Variable Scope and DocString. Learnt about global and local variables, as well as classes. We used Spyder as our IDE. We posted our files on a public GitHub repository so that others can review it.

Appendix

The script below had several TODOs in the code. Resolved and modified these as required and finish organizing the code to create a version of our CD Inventory program that has a menu structure and allows the user to enter CD data, view the current inventory, save data to a CDInventory.txt data file, delete CD from inventory and exit the program.

CDInventory.py

```
#------------------------------------------#
# Title: CDInventory.py
# Desc: Working with classes and functions.
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# Jgnanaraj, 2021-Feb-24, Modified File
#------------------------------------------#
```

```
8.
9.  # -- DATA -- #
10.    strChoice = '' # User input
11.    lstTbl = []  # list of lists to hold data
12.    dicRow = {}  # list of data row
13.    strFileName = 'CDInventory.txt'  # data storage file
14.    objFile = None  # file object
15.
16.
17.    # -- PROCESSING -- #
18.    class DataProcessor:
19.        # TODO add functions for processing here
20.        def input_data_process(idCd, cdTitle, cdArtist):
21.            """Function to add user input data to table
22.
23.            Reads the data from file identified by file_name into a 2D tabl
   e
24.            (list of dicts) table one line in the file represents one dictionar
   y row in table.
25.
26.            Args:
27.                The ID, Title and Artist newly input by the user
28.
29.            Returns:
30.                None.
31.            """
32.            intID = int(idCd)
33.            dicRow = {'ID': intID, 'Title': cdTitle, 'Artist': cdArtist}
34.            lstTbl.append(dicRow)
35.
36.        def delete_row(rowId):
37.            """Function to delete row from the inventory
38.
```

```
        Args:
            The ID of the row intended to be deleted

        Returns:
            None.
        """
        intRowNr = -1
        blnCDRemoved = False
        for row in lstTbl:
            intRowNr += 1
            if row['ID'] == rowId:
                del lstTbl[intRowNr]
                blnCDRemoved = True
                break
        if blnCDRemoved:
            print('The CD was removed')
        else:
            print('Could not find this CD!')
    #pass


class FileProcessor:
    """Processing the data to and from text file"""

    @staticmethod
    def read_file(file_name, table):
        """Function to manage data ingestion from file to a list of dictionaries

        Reads the data from file identified by file_name into a 2D table
        (list of dicts) table one line in the file represents one dictionary row in table.
```

```
69.
70.         Args:
71.             file_name (string): name of file used to read the data from
72.             table (list of dict): 2D data structure (list of dicts) that holds
    the data during runtime
73.
74.         Returns:
75.             None.
76.         """
77.         table.clear()  # this clears existing data and allows to load dat
    a from file
78.         objFile = open(file_name, 'r')
79.         try:
80.             for line in objFile:
81.                 data = line.strip().split(',')
82.                 dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]
    }
83.                 table.append(dicRow)
84.         except:
85.             ("Something happened here - maybe there is nothing in the
    file yet?")
86.         objFile.close()
87.
88.     @staticmethod
89.     def write_file(file_name, table):
90.         """Function to manage data from file to a list of dictionaries
91.
92.         Writes the data from a 2D table (list of dicts) into a long string
     and saved into a text file.
93.
94.         Args:
95.             file_name (string): name of file used to write the data to
```

```
96.             table (list of dict): 2D data structure (list of dicts) that holds
    the data during runtime
97.
98.         Returns:
99.             None.
100.        """
101.
102.        # ADDED - TODO Add code here
103.        new_line = ""
104.        objFile = None
105.
106.        for row in table:
107.            print(row)
108.            for item in row.values():
109.                new_line = new_line + str(item) + ","
110.            new_line = new_line[:-1] + "\n"
111.
112.        objFile = open(strFileName, "w")
113.        objFile.write(new_line)
114.        objFile.close()
115.        print("Data saved!")
116.
117.    def save_file(file_name, table):
118.        """Function to save the text file
119.
120.        Writes the data from a 2D table (list of dicts) into a long string
    and saved into a text file.
121.
122.        Args:
123.            file_name (string): name of file used to write the data to
124.            table (list of dict): 2D data structure (list of dicts) that holds
    the data during runtime
125.
```

```
126.         Returns:
127.             None.
128.         """
129.         objFile = open(file_name, 'w')
130.         for row in table:
131.             lstValues = list(row.values())
132.             lstValues[0] = str(lstValues[0])
133.             objFile.write(','.join(lstValues) + '\n')
134.         objFile.close()
135.         #pass
136.
137.
138. # -- PRESENTATION (Input/Output) -- #
139.
140. class IO:
141.     """Handling Input / Output"""
142.
143.     @staticmethod
144.     def print_menu():
145.         """Displays a menu of choices to the user
146.
147.         Args:
148.             None.
149.
150.         Returns:
151.             None.
152.         """
153.
154.         print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i]
    Display Current Inventory')
155.         print('[d] delete CD from Inventory\n[s] Save Inventory to file
    \n[x] exit\n')
156.
```

```
157.     @staticmethod
158.     def menu_choice():
159.         """Gets user input for menu selection
160.
161.         Args:
162.             None.
163.
164.         Returns:
165.             choice (string): a lower case sting of the users input out of t
    he choices l, a, i, d, s or x
166.
167.         """
168.         choice = ' '
169.         while choice not in ['l', 'a', 'i', 'd', 's', 'x']:
170.             choice = input('Which operation would you like to perform
    ? [l, a, i, d, s or x]: ').lower().strip()
171.         print()  # Add extra space for layout
172.         return choice
173.
174.     @staticmethod
175.     def show_inventory(table):
176.         """Displays current inventory table
177.
178.
179.         Args:
180.             table (list of dict): 2D data structure (list of dicts) that holds
    the data during runtime.
181.
182.         Returns:
183.             None.
184.
185.         """
186.         print('======= The Current Inventory: =======')
```

```
187.         print('ID\tCD Title (by: Artist)\n')
188.         for row in table:
189.             print('{}\t{} (by:{})'.format(*row.values()))
190.         print('======================================')
191.
192.     # TODO add I/O functions as needed
193.     def ask_user_data():
194.         """Asks for user data
195.
196.         Args: None
197.         Returns: The ID, the CD Title and the Artist of the title
198.         """
199.         ID = input('Enter ID: ').strip()
200.         Title = input('What is the CD\'s title? ').strip()
201.         Artist = input('What is the Artist\'s name? ').strip()
202.         return ID, Title, Artist
203.
204.
205. # 1. When program starts, read in the currently saved Inventory
206. FileProcessor.read_file(strFileName, lstTbl)
207.
208. # 2. start main loop
209. while True:
210.     # 2.1 Display Menu to user and get choice
211.     IO.print_menu()
212.     strChoice = IO.menu_choice()
213.
214.     # 3. Process menu selection
215.
216.     # 3.1 process exit first
217.     if strChoice == 'x':
218.         break
219.
```

```
220.    # 3.2 process load inventory
221.    if strChoice == 'l':
222.        print('WARNING: If you continue, all unsaved data will be lost
    and the Inventory re-loaded from file.')
223.        strYesNo = input('type \'yes\' to continue and reload from file
    . otherwise reload will be canceled')
224.        if strYesNo.lower() == 'yes':
225.            print('reloading...')
226.            FileProcessor.read_file(strFileName, lstTbl)
227.            IO.show_inventory(lstTbl)
228.        else:
229.            input('canceling... Inventory data NOT reloaded. Press [ENT
    ER] to continue to the menu.')
230.            IO.show_inventory(lstTbl)
231.        continue  # start loop back at top.
232.
233.    # 3.3 process add a CD
234.    elif strChoice == 'a':
235.        # 3.3.1 Ask user for new ID, CD Title and Artist
236.        # ADDED TODO move IO code into function
237.        strID, strTitle, stArtist = IO.ask_user_data()
238.
239.        # 3.3.2 Add item to the table
240.        # ADDED TODO move processing code into function
241.        DataProcessor.input_data_process(strID, strTitle, stArtist)
242.        IO.show_inventory(lstTbl)
243.        continue  # start loop back at top.
244.
245.    # 3.4 process display current inventory
246.    elif strChoice == 'i':
247.        IO.show_inventory(lstTbl)
248.        continue  # start loop back at top.
249.
```

```
250.    # 3.5 process delete a CD
251.    elif strChoice == 'd':
252.        # 3.5.1 get Userinput for which CD to delete
253.        # 3.5.1.1 display Inventory to user
254.        IO.show_inventory(lstTbl)
255.        # 3.5.1.2 ask user which ID to remove
256.        intIDDel = int(input('Which ID would you like to delete? ').stri
    p())
257.        # 3.5.2 search thru table and delete CD
258.        # ADDED TODO move processing code into function
259.        DataProcessor.delete_row(intIDDel)
260.
261.        IO.show_inventory(lstTbl)
262.        continue  # start loop back at top.
263.
264.    # 3.6 process save inventory to file
265.    elif strChoice == 's':
266.        # 3.6.1 Display current inventory and ask user for confirmatio
    n to save
267.        IO.show_inventory(lstTbl)
268.        strYesNo = input('Save this inventory to file? [y/n] ').strip().lo
    wer()
269.        # 3.6.2 Process choice
270.        if strYesNo == 'y':
271.            # 3.6.2.1 save data
272.            # ADDED TODO move processing code into function
273.            FileProcessor.save_file(strFileName, lstTbl)
274.        else:
275.            input('The inventory was NOT saved to file. Press [ENTER] t
    o return to the menu.')
276.        continue  # start loop back at top.
277.
```

```
278.     # 3.7 catch-
    all should not be possible, as user choice gets vetted in IO, but to be
    save:
279.     else:
280.         print('General Error')
```

Created a script and Compiled to verify output:

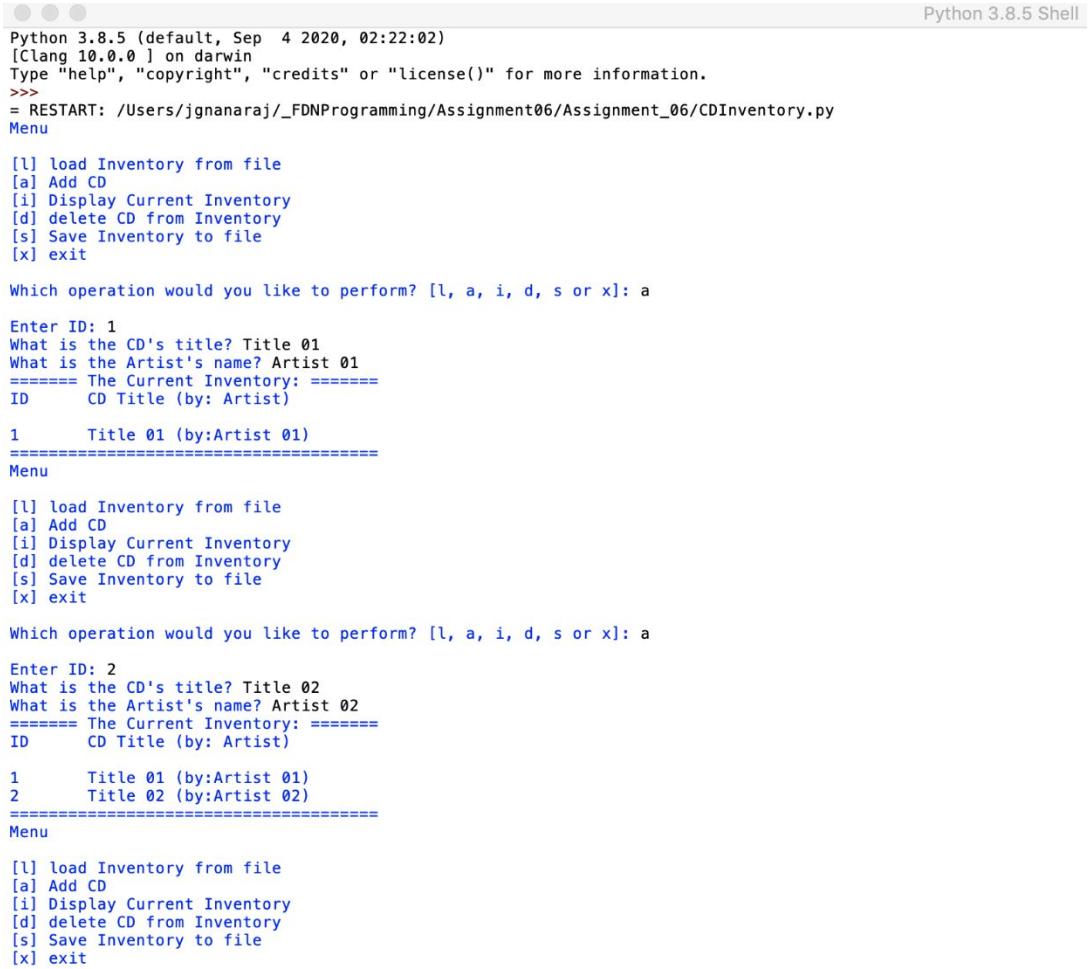

Figure 1 - Screen capture

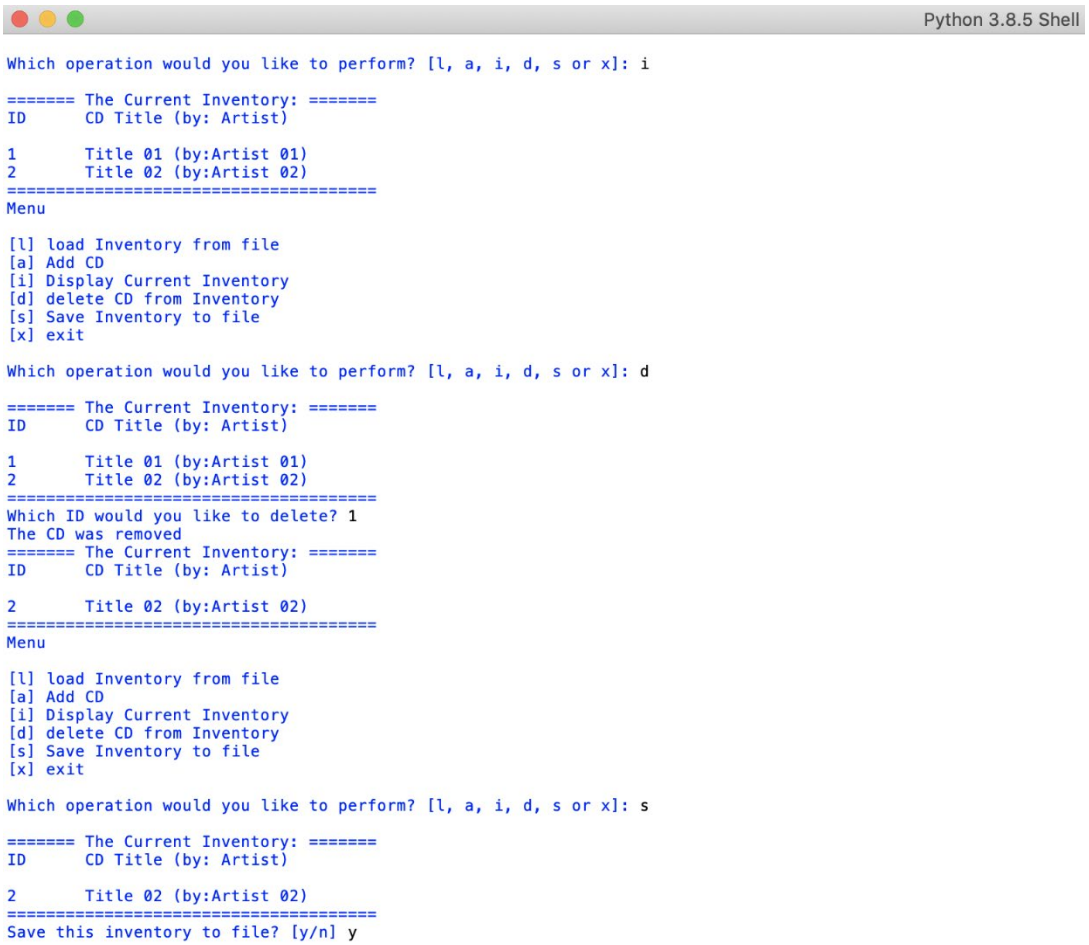

Figure 2 - Screen capture

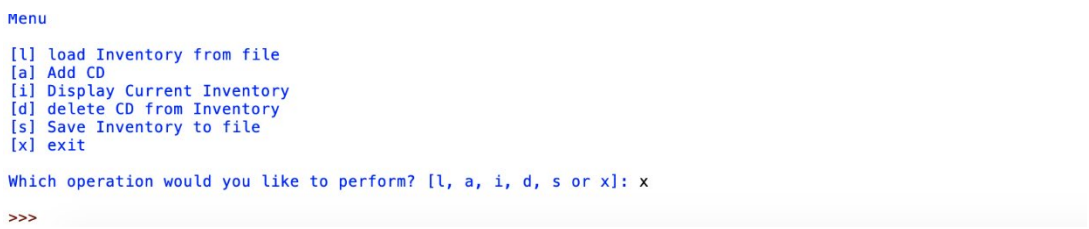

Figure 3 - Screen capture

References:

https://runestone.academy/runestone/books/published/thinkcspy/Lists/TuplesasReturnValues.html
https://www.w3schools.com/python/python_functions.asp
https://docs.python.org/3/tutorial/classes.html
https://www.learnpython.org/en/Functions
FDN_Py_Module_06.pdf