**Dimitrios Tsakalos**

Sunday, March 7, 2021

IT FDN 110 A Wi 21: Foundations of Programming: Python

[Assignment 08](https://github.com/tsakdimi/Assignment_08)

**CD Inventory program working with a list of objects**

# Introduction

In Assignment 08 for this course, I reviewed the material for Module 08 which included watching the module videos, reading book chapter 08 and reading webpages explaining object-oriented programming (OOP). I read through the Foundations of Programming (Python) PDF and completed the assignment by applying my knowledge to create a script in Python that collects input from the user, organizes the code in classes, constructors, attributes, properties and methods so the data can be stored as a list of objects. The script saves the data in a comma separated values text file and accounts for handling errors. The knowledge document and Python file were uploaded to [GitHub](https://github.com/tsakdimi/Assignment_08) (external reference)[[1]](#footnote-1).

# Reviewing the material

The material I reviewed for Module 08 included reading book chapter 08 and searching online for information about OOP in Python. After looking up information online I then worked on the Foundations of Programming (Python) PDF provided in module 08, which built on Assignment 06 and Assignment 07. After reviewing the materials and the assignment requirements, I started working on the labs step by step. The labs helped and I gained an understanding of the new concepts. This week’s challenges were:

* Understanding OOP
* Learning to work on an assignment that contains mainly lines of pseudocode

# Working on the Foundations of Programming (Python) Module 07

After reviewing the above materials, I proceeded to work on the Foundations of Programming Module 08 Labs. I started with LAB-08A. I created a file and added code to create the class TrackInfo, added code to create fields for position, title, length with data types int, string, string, respectively (see Figure 1). I then proceeded to run it in the Spyder console (see Figure 2). The code performs the following:

* Line 8 creates the class TrackInfo
* Lines 11 to 13 create the fields for position, title, length with the relevant data types
* Lines 20 to 31 instantiate an object on the first line of the block, followed by referring to that instance using the fields as methods to assign the wanted values to the corresponding object.
* Lines 33 to 35 print using the objects as keyword arguments.

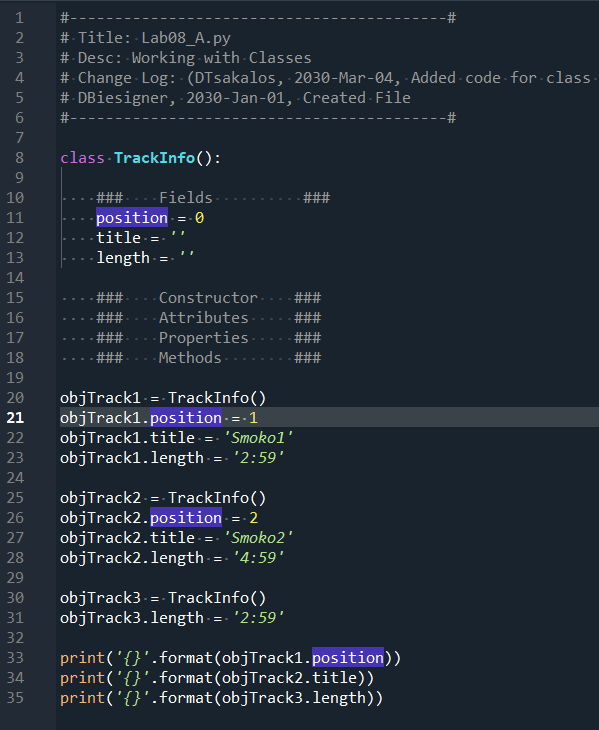


Figure - LAB08-A code

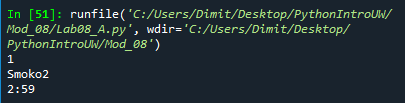


Figure – LAB-08A console

I used the code in LAB-08A as input to LAB-08B and modified it to work with constructors (see Figure 3). I then confirmed the program functioned correctly (see Figure 4). The code performs the following:

* Line 8 creates the class TrackInfo
* Lines 11 to 13 create the fields for position, title, length with the relevant data types
* Lines 16 defines the constructor with the relevant arguments
* Lines 19 to 21 assign the attributes to the arguments
* Lines 26 to 28 create the instances for one track listing. I only populated the relevant strings
* Lines 31 to 33 print the information using the methods as keyword arguments.

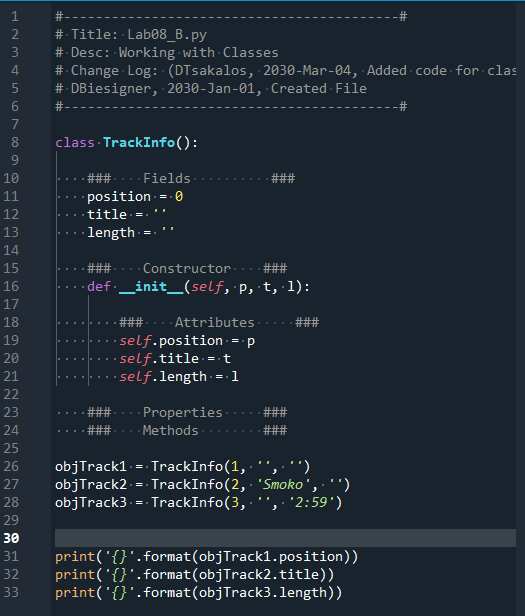


Figure – LAB-08B code

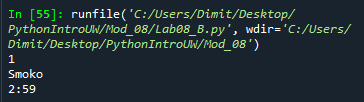


Figure – LAB-08B console

For LAB-08C I made a copy of the script from LAB-08B. I then added code to populate the class attributes (see Figure 5), followed by testing the script (see Figure 6). The code performs the following:

* Line 8 creates the class TrackInfo
* Lines 11 to 13 create the fields for position, title, length with the relevant data types
* Lines 16 defines the constructor with the relevant arguments
* Lines 19 to 21 assign the attributes to the arguments
* Lines 32 creates the instance for one track listing
* Lines 38 to 40 print the information using the methods as keyword arguments.

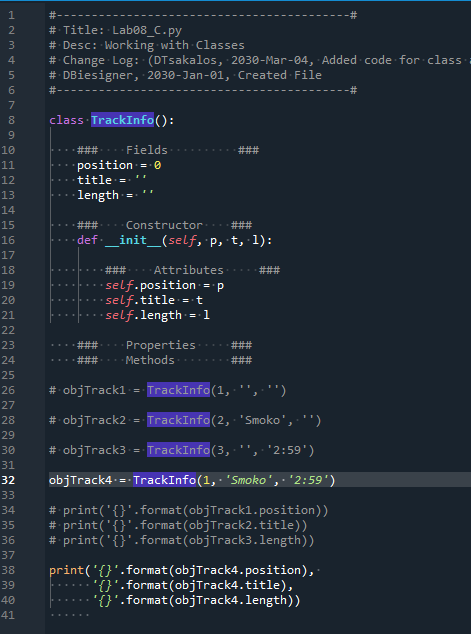


Figure – LAB-08C code

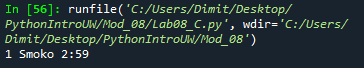


Figure – LAB-08C console

Moving on to LAB-08D, I added code to create properties and code to verify the validity of the values (see Figure 7). I then tested the code (see Figure 8). The code performs the following:

* Line 8 creates the class TrackInfo
* Line 12 defines the constructor with the relevant arguments
* Lines 15 to 17 assign the attributes to the arguments
* Lines 20 to 51 define the getter and setter methods
* Lines 53 creates an object with an assigned message.
* Line 54 print the information using the methods as keyword arguments.



Figure – LAB-08D code



Figure – LAB-08D console

For the final LAB-08E, I added a method to format the content of the attributes, and code to verify the proper functioning of the method (see Figure 9). I finished by testing the script (see Figure 10). The code performs the following:

* Line 8 creates the class TrackInfo
* Line 12 defines the constructor with the relevant arguments
* Lines 15 to 17 assign the attributes to the arguments
* Lines 20 to 51 define the getter and setter methods
* Lines 54 to 56 use the static method to create a method that when evoked returns a formatted string.
* Line 54 creates an object with an assigned message.
* Line 55 print the information evoking the \_\_str\_\_ method with the object created above passed in as an argument.

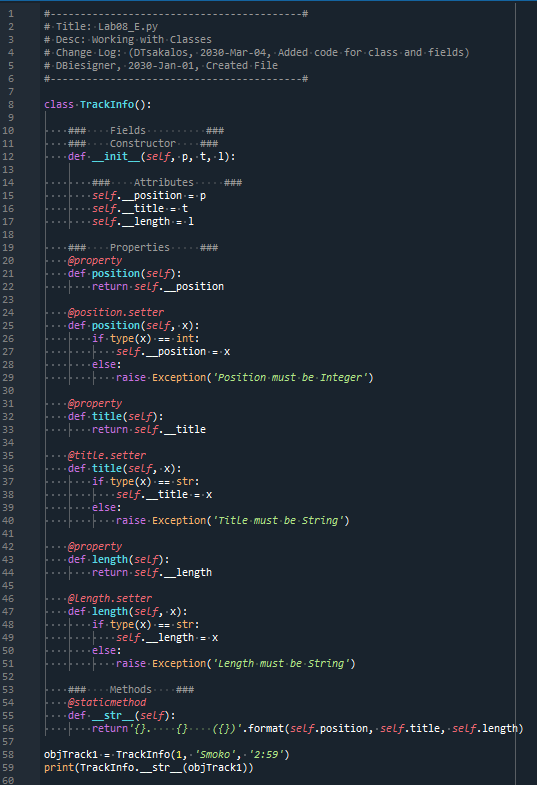


Figure – LAB-08E code



Figure – LAB-08E console

# Creating a CD Inventory program working with a list of objects

After completing the labs, I moved onto Assignment08. After going through all the steps of folder creation and header modifications I started working on the code (see Appendix Listing CD\_Inventory.py). The code performs the following:

* Line 10 to 11 declare the global variables
* Line 15 to 26 create the class CD and its doctring that stores data about a music track
* Lines 28 to 34 create the constructor with attributes for the class
* Lines 36 to 68 create the getters and setters for the properties of the class
* Lines 71 to 72 define a method for the class that returns a formatted version of our data for printing to screen
* Lines 74 to 75 define a method for the class that returns a formatted version of our data for saving to file
* Lines 80 to 86 create the FileIO class and docstring that saves from file and loads from file
* Lines 89 to 93 define the save\_inventory method that saves to file
* Lines 95 to 109 define the load\_inventory method that loads the data from the file, formats it and returns it as a list of objects
* Lines 114 to 123 create the class IO and its doctring that presents the information to the user and collects input
* Lines 127 to 131 define the print\_menu method that prints a menu for the user
* Lines 133 to 140 define the menu\_choice method that captures and returns user input
* Lines 143 to 150 define the show\_inventory method that displays the inventory to the user
* Lines 154 to 163 define the load\_choice method that allows the user to load from file data
* Lines 166 to 192 define the get\_new\_cd\_data method that collects the data about a new music track added by the user and returns it as the object track
* Lines 197 to 198 create an instance and then load existing saved data from file by using the appropriate method
* Lines 201 to 240 call the appropriate methods within classes to perform the operations the user selected according to the menu

After that, I ensured the program ran correctly in the Spyder console (see Figure 11), then I ran a check in the Anaconda Prompt (see Figure 12) and finished with checking the cdInventory.txt file (see Figure 13).

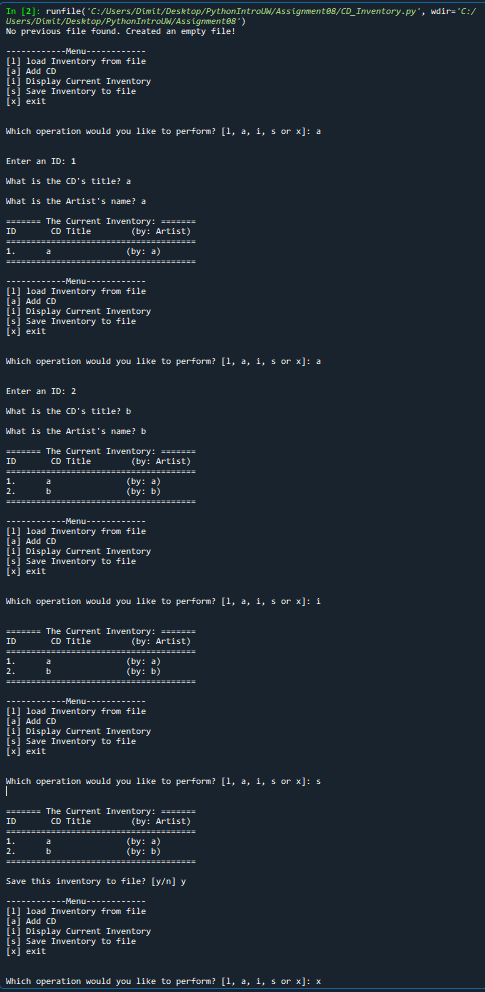


Figure - CD\_Inventory.py console

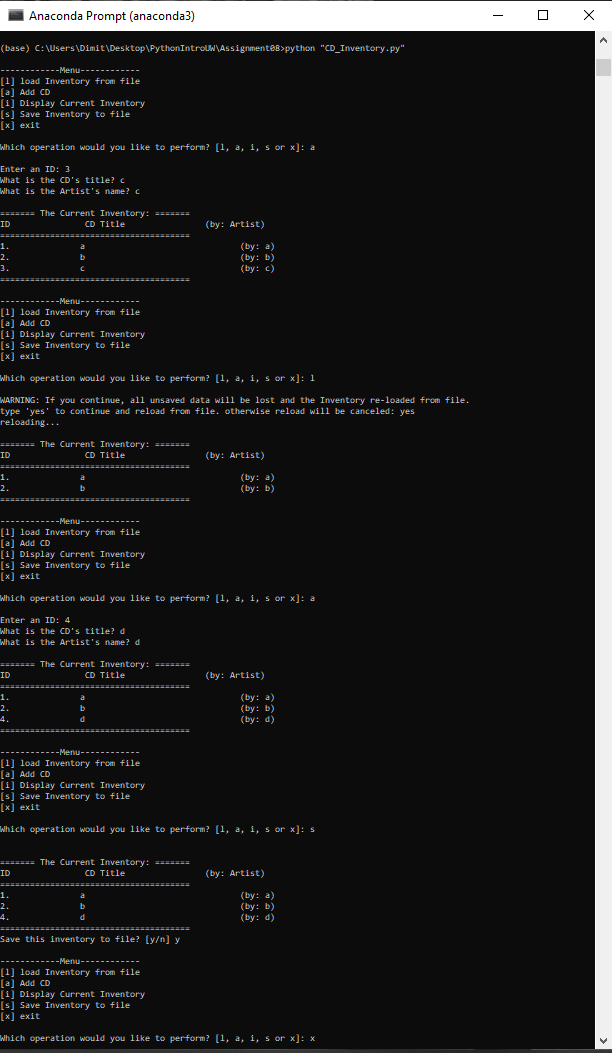


Figure - CD\_Inventory.py in Anaconda Prompt

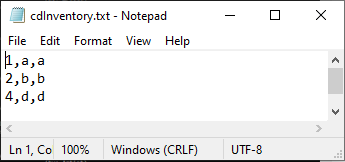


Figure - cdInventory.txt saving correctly

# Summary

After reviewing the material and going through the Labs, I worked on Assignment 08. The challenges this week were understanding the basics of OOP and how to implement it based on pseudocode. The main errors I encountered were missing positional arguments. The labs were very helpful, as was Wednesday’s session for clearing up the concepts in my mind and helping me with labs D and E. I also had some mistakes with using @staticmethod in areas where it created problems. For example, when using it in the load and save methods my code would not run because it was missing a positional argument. The hardest part of the assignment was feeling comfortable with the two previous assignments, so that I had a good foundation of information to build off of and be able to write the code in a clean and correct manner. I expected this week to be harder than what it was, but still want to improve on my syntax memorization and pseudocode writing.

# Appendix

## Listing Github

<https://github.com/tsakdimi/Assignment_08>

## Listing CD\_Inventory.py

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240 | *#------------------------------------------#*  *# Title: CD\_Inventory.py*  *# Desc: Assignnment 08 - Working with classes*  *# Change Log: (DTsakalos, 2021-Mar-07, Added code in place of pseudocode)*  *# DBiesinger, 2030-Jan-01, created file*  *# DBiesinger, 2030-Jan-01, added pseudocode to complete assignment 08*  *#------------------------------------------#*  *# -- DATA -- #*  strFileName = 'cdInventory.txt'  lstOfCDObjects = []  *# -- CD Track Data -- #*  **class** CD:  """Stores data about a CD:  properties:  cd\_id: (int) with CD ID  cd\_title: (string) with the title of the CD  cd\_artist: (string) with the artist of the CD  methods:  \_\_str\_\_: -> (string) with cd\_id, cd\_title, cd\_artist formatted for screen display  file\_str: -> (string) with cd\_id, cd\_title, cd\_artist formatted for saving to file  """  *### Contructor ###*  **def** \_\_init\_\_(self, cd\_id, cd\_title, cd\_artist):  *### Attributes ###*  self.\_\_cd\_id = cd\_id  self.\_\_cd\_title = cd\_title  self.\_\_cd\_artist = cd\_artist  *### Properties ####*  @property  **def** cd\_id(self):  **return** self.\_\_cd\_id  @cd\_id.setter  **def** cd\_id(self, cd\_id):  **if** type(cd\_id) == int:  self.\_\_cd\_id = cd\_id  **else**:  **raise** Exception('CD ID must be an integer')  @property  **def** cd\_title(self):  **return** self.\_\_cd\_title  @cd\_title.setter  **def** cd\_title(self, cd\_title):  **if** type(cd\_title) == str:  self.\_\_cd\_title = cd\_title  **else**:  **raise** Exception('CD Title must be a string')  @property  **def** cd\_artist(self):  **return** self.\_\_cd\_artist  @cd\_artist.setter  **def** cd\_artist(self, cd\_artist):  **if** type(cd\_artist) == str:  self.\_\_cd\_artist = cd\_artist  **else**:  **raise** Exception('CD Artist must be a string')  *### Methods ###*  **def** \_\_str\_\_(self):  **return** '{}.\t\t{}\t\t\t\t(by: {})'.format(self.cd\_id, self.cd\_title, self.cd\_artist) *# using getter*  **def** file\_str(self):  **return** '{},{},{}\n'.format(self.cd\_id, self.cd\_title, self.cd\_artist) *# using getter*  *# -- PROCESSING -- #*  **class** FileIO:  """Processes data to and from file:  methods:  save\_inventory(file\_name, lst\_Inventory): -> None  load\_inventory(file\_name): -> (a list of CD objects)  """  *### Methods ###*  **def** save\_inventory(self, file\_name, lst\_Inventory):  objFile = open(file\_name, 'w')  **for** track **in** lst\_Inventory:  objFile.write(track.file\_str())  objFile.close()  **def** load\_inventory(self, file\_name):  lst\_Inventory = []  **while** True:  **try**:  objFile = open(file\_name, 'r')  **break**  **except** FileNotFoundError:  objFile = open(file\_name, 'w')  **print**('No previous file found. Created an empty file!')  **for** line **in** objFile:  data = line.strip().split(',')  track = CD(int(data[0]), data[1], data[2]) *# using constructor*  lst\_Inventory.append(track)  objFile.close()  **return** lst\_Inventory  *# -- PRESENTATION (Input/Output) -- #*  **class** IO:  """ Presents the information to use and collects input:    methods:  print\_menu: Prints menu for user -> None  menu\_choice: -> (string) of the choice the user selects  show\_inventory(table): (string) Prints a list of Objects -> None  load\_choice: -> (boolean) to notify of data loss if loading  get\_new\_cd\_data: -> (object) of the data relating to the song -> None  """  *### Methods ###*  *#show menu to user*  @staticmethod  **def** print\_menu():  **print**('\n------------Menu------------')  **print**('[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')  **print**('[s] Save Inventory to file\n[x] exit\n')  *#captures user's choice*  @staticmethod  **def** menu\_choice():  choice = ' '  **while** choice **not** **in** ['l', 'a', 'i', 's', 'x']:  choice = input('Which operation would you like to perform? [l, a, i, s or x]: ').lower().strip()  **print**() *# Add extra space for layout*  **return** choice  *#display the current data on screen*  @staticmethod  **def** show\_inventory(table):  **print**('\n======= The Current Inventory: =======')  **print**('ID\t\t CD Title \t\t (by: Artist)')  **print**('======================================')  **for** track **in** table:  **print**(track)  **print**('======================================')  *#allow user to load*  @staticmethod  **def** load\_choice():  load\_file = False  **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')  strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled: ')  **if** strYesNo.lower() == 'yes':  **print**('reloading...')  load\_file = True  **else**:  input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')  **return** load\_file  *#get CD data from user*  @staticmethod  **def** get\_new\_cd\_data():  *# 3.3.1 Ask user for new ID, CD Title and Artist*  **while** True:  **try**:  intID = int(input('Enter an ID: ').strip())  **break**  **except** ValueError: *#making sure the program does not crash with string as input*  **print**('Invalid Input! Try again.')  **while** True:  **try**:  strTitle = input('What is the CD\'s title? ').strip()  **if** strTitle == '':  **raise** ValueError()  **break**  **except** ValueError: *#making sure the program does not contain empty string*  **print**('Invalid Input! Try again.')  **while** True:  **try**:  strArtist = input('What is the Artist\'s name? ').strip()  **if** strArtist == '':  **raise** ValueError()  **break**  **except** ValueError: *#making sure the program does not contain empty string*  **print**('Invalid Input! Try again.')  track = CD(intID, strTitle, strArtist)  **return** track  *# -- Main Body of Script -- #*  *# 1. Load data from file into a list of CD objects on script start*  fileIO\_obj = FileIO()  lstOfCDObjects = fileIO\_obj.load\_inventory(strFileName)  *# 2. Display menu to user*  **while** True:  *# 2.1 Display Menu to user and get choice*  IO.print\_menu()  strChoice = IO.menu\_choice()  *# 3. Process menu selection*  *# 3.1 process exit first*  **if** strChoice == 'x':  **break**  *# 3.2 process load inventory*  **if** strChoice == 'l':  reload\_file = IO.load\_choice()  **if** reload\_file:  lstOfCDObjects = fileIO\_obj.load\_inventory(strFileName)  IO.show\_inventory(lstOfCDObjects)  *# 3.3 process add a CD*  **elif** strChoice == 'a':  track = IO.get\_new\_cd\_data()  lstOfCDObjects.append(track)  IO.show\_inventory(lstOfCDObjects)  *# 3.4 process display current inventory*  **elif** strChoice == 'i':  IO.show\_inventory(lstOfCDObjects)  *# 3.5 process save inventory to file*  **elif** strChoice == 's':  IO.show\_inventory(lstOfCDObjects)  strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()  *# 3.5.2 Process choice*  **if** strYesNo == 'y':  fileIO\_obj.save\_inventory(strFileName, lstOfCDObjects)  **else**:  input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')  *# 3.6 catch-all should not be possible, as user choice gets vetted in IO, but to be safe:*  **else**:  **print**('General Error') |

1. Retrieved 2021-Mar-07 [↑](#footnote-ref-1)