Tiago Rodrigues

04/12/2022

Foundations of Programming, Python

Assignment 08

# Introduction

# In this module I will talk Constructors, Destructors, Attributes, Properties, Methods and the script requested for this assignment that is a “copy” of the last assignment but work with Object-Oriented Programming.

# Constructors

Constructors are a dedicated method is invoked when creating an object. Python’s constructor method is the dunder init method: \_\_init\_\_().

Python implicity calls the \_\_init\_\_() method and passes any arguments provided when creating an object to the \_\_init\_\_() method.

# Destructors

As the name implies, these methods are used when an object gets destroyed or de-allocated. Depending on the language you’re using, these need to be more or less sophisticated. In most cases, they are responsible of freeing up the memory used by the object, cleaning up references and similar tasks

# Attributes

In Python, Attributes are internal fields or variables that hold data.

# Properties

One common concept of controlling validity of values assigned to attributes in your class is to make the attributes private and enforce the interaction with them thru methods that have control mechanism build in. These special methods are called properties. Typically, you’ll create two for each attribute: One to set it and one to access it. Most often they are called “getter” or “accessor” for reading (or getting) the attribute and ”setter” or “mutator” for writing (or setting) of the attribute. In most programming languages you can use a private keyword to make attributes (also applicable to functions) private to only be accessible from within the class.

# Methods

Methods are like functions in a script: They allow you organize your statements into blocks that can be invoked by calling the method’s name. The one difference is that a method call also submits a reference to the object it’s invoked on, so the first attribute supplied to a method is the “self” reference.

# Assignment08

In this assignment I’ve used the previous script on assignment07 added structured error handling around the areas where there is user interaction, type casting and file access operations and used Object-Oriented Programming.

Here is the code.

|  |  |
| --- | --- |
| 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 | *#------------------------------------------#*  *# Title: CD\_Inventory.py*  *# Desc: Assignnment 08 - Working with classes*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, created file*  *# DBiesinger, 2030-Jan-01, added pseudocode to complete assignment 08*  *# Tiago Rodrigues, 2022-Dez-03, update code to work with classes*  *#------------------------------------------#*  **import** **os**  *# -- DATA -- #*  strFileName = 'cdInventory.txt'  lstOfCDObjects = []  **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:*  *new\_cd: Function that takes new CD data and creates a new CD while appending to list*  *"""*  *# TODone Add Code to the CD class*  *# -- Fields -- #*  cd\_id = ''  cd\_title = ''  cd\_artist = ''  *# -- Construct -- #*  **def** \_\_init\_\_(self, Id, title, artist):  *# -- Atributes -- #*  self.\_\_cd\_id = Id  self.\_\_cd\_title = title  self.\_\_cd\_artist = artist  **def** contents(self):  **return** [self.cd\_id, self.cd\_title, self.cd\_artist]    **def** display(self):  **return** '**{}\t{}** (by:**{}**)'.format(\*self.contents())      *# -- Properties -- #*  @property  **def** cd\_id(self):  **return** self.\_\_cd\_id  @cd\_id.setter  **def** cd\_id(self, value):  **if** str(value).isnumeric():  self.\_\_cd\_id = value  **else**:  **raise** **Exception**('The Cd Id must be an integer!')  @property  **def** cd\_title(self):  **return** self.\_\_cd\_title    @property  **def** cd\_artist(self):  **return** self.\_\_cd\_artist    *# -- Methods -- #*  @staticmethod  **def** new\_cd(cdinfo):  *'''Adds new CD to list of CDs (lstOfCDObjects)*  *Args:*  *cdinfo(CD): Object CD consisting of CD data*  *Data:*  *None*  *'''*  lstOfCDObjects.append(cdinfo)    *# -- PROCESSING -- #*  **class** **FileIO**:  *"""Processes data to and from file:*  *properties:*  *methods:*  *save\_inventory(file\_name, lst\_Inventory): -> None*  *load\_inventory(file\_name): -> (a list of CD objects)*  *"""*  *# TODone Add code to process data from a file*  @staticmethod  **def** read\_from\_file(self, file\_name):  *'''Function to read a file write to a list*  *Args:*  *file\_name(string): name of file used to read the data from*  *Returns:*  *data(object): list of CD objects*  *'''*  lstOfCDObjects.clear()  **if** os.path.exists(file\_name):  **with** open(file\_name, 'r') **as** ObjFile:  **for** line **in** ObjFile.readlines():  data = line[:-1].split(',')  lstOfCDObjects.append(CD(data[0], data[1], data[2]))  **else**:  print('The file **{}** doesn**\'**t exist'.format(file\_name))  **while** **True**:  rsp = input('Would you like to create a new file? (y/n):').strip()  **if** rsp.lower() == 'y':  **with** open(file\_name, 'w') **as** objFile:  print('File created successfully!')  **break**  **elif** rsp.lower() == 'n':  **break**  **else**:  print('Choose one of the two options')  *# TODone Add code to process data to a file*  @staticmethod  **def** add\_to\_file(self, lst, filename):  *'''Function to write into the file*  *Args:*  *file\_name(string): name of file used to read the data from*  *lst (list): data structure that holds the data during runtime*  *Returns:*  *None.*  *'''*  **with** open(filename, 'w') **as** ObjFile:  **for** row **in** lst:  [d1, d2, d3] = str(row.cd\_id), row.cd\_title, row.cd\_artist  ObjFile.write(','.join([d1, d2, d3]) + '**\n**')  ObjFile.close()  print('Data added successfully')  *# -- PRESENTATION (Input/Output) -- #*  **class** **IO**:  *# TODone add docstring*  *# TODone add code to show menu to user*  @staticmethod  **def** show\_menu():  *'''This function shows to user the menu*  *Args:*  *None.*  *Retunrs:*  *None.*  *'''*  print('Menu')  print('[d] Display the current inventory**\n**[a] Add data to the inventory**\n**[s] Save inventory to file**\n**[l] Load inventory from file**\n**[x] Exit')  *# TODone add code to captures user's choice*  @staticmethod  **def** user\_choice():  *'''Simple Function to request the user choice*  *Args:*  *None.*  *Retunrs:*  *Choice (string): a lower case string of the users choice*  *'''*  choice = ' '  **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:  choice = input('Select the operation: ').lower().strip()  print() *# Add extra space for layout*  **return** choice  *# TODone add code to display the acurrent data on screen*  @staticmethod  **def** cd\_list(self, lst):  *'''Display current inventory*  *Args:*  *lst(list of Cds): List of cds*  *Retunrs:*  *None.*  *'''*  print('======= The Current Inventory: =======')  print('ID**\t**CD Title (by: Artist)**\n**')  **for** row **in** lst:  print(row.display())  *#print('{}\t{} (by:{})'.format(\*row))*  **pass**  print('======================================')  *# TODone add code to get CD data from user*  @staticmethod  **def** get\_user\_input(self):  *'''Simple Function to get the user input*  *Args:*  *None.*  *Retunrs:*  *CD info: Object of CD that contains ID, CD Title and CD Title*  *'''*  **while** **True**:  **try**:  strId = int(input('Enter Cd Id:').strip())  **break**  **except** **ValueError**:  print('The Cd Id must be an integer!')  strTitle = input('Enter Cd Title:').strip()  strArtist = input('Enter Cd Artist:').strip()  **return** strId, strTitle, strArtist  *# -- Main Body of Script -- #*  *# TODone Add Code to the main body*  FileIO.read\_from\_file(FileIO, strFileName)*# Load data from file into a list of CD objects on script start*  **while** **True**:  *# Display menu to user*  IO.show\_menu()  strchoice = IO.user\_choice()  *# let user exit program*  **if** strchoice == 'x':  **break**  *# show user current inventory*  **if** strchoice == 'd':  IO.cd\_list(IO, lstOfCDObjects)  *# let user add data to the inventory*  **elif** strchoice == 'a':  cd\_id, cd\_title, cd\_artist = IO.get\_user\_input(IO)  cd = CD(cd\_id, cd\_title, cd\_artist)  lstOfCDObjects.append(cd)  *#CD.new\_cd(IO.get\_user\_input())*  *# let user save inventory to file*  **elif** strchoice == 's':  FileIO.add\_to\_file(FileIO, lstOfCDObjects, strFileName)  **continue**  *# let user load inventory from file*  **elif** strchoice == 'l':  FileIO.read\_from\_file(FileIO, strFileName)  IO.cd\_list(IO, lstOfCDObjects)  **else**:  print('Choose one option of the menu') |

## Using Spyder terminal

Here I’ve added some data into the list.

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 1 - Data added inside the list

Display the data inside the list.

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 2 - Display current data in list

Save data inside the list into .txt File.

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 3 - Save data inside the file

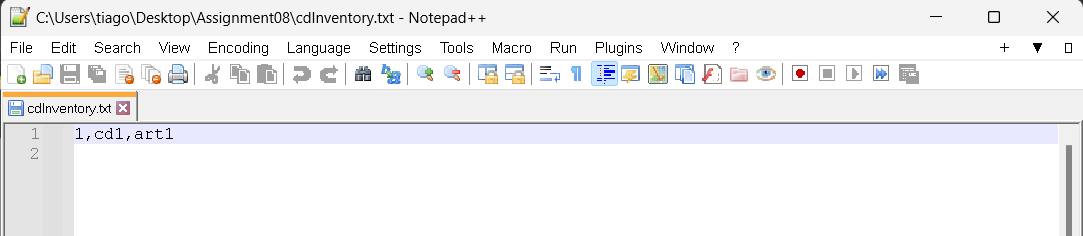


Ilustração 4 - .txt file

Loading the data inside the file

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 5 - Loading data

## Using computer terminal

Adding more data to the list

Uma imagem com texto

Descrição gerada automaticamente

Ilustração - Data added inside the list

Display de data inside list

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 7- Display current data in list

Save the data inside the .txt file.

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 8- Save data inside the file

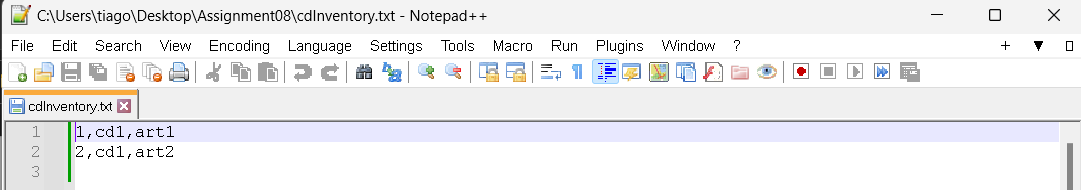


Ilustração 9 - .txt file

Loading all the data inside the .txt file.

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 10 - Loading data

# Summary

In this assignment we introduce Object-Oriented Programming.Here is the GitHub url:

<https://github.com/Ehz10/Assignment_07.git>