Tiago Rodrigues

27/11/2022

Foundations of Programming, Python

Assignment 07

# Introduction

# In this module I will talk about binary files, structured error handling and the script requested for this assignment that is a “copy” of the last assignment but work with binary files

# Binary files

Binary Files are intended to only be read by a computer system. When you write a computer program, data is held in variables and in more complex data structures such as arrays, dictionaries, or lists. A Binary file allows you to store this data in a form that preserves the structures used in your program.

# Structured Error Handling

Structured w error handling is an error handling model, and it is characterized by the following features:

* Represents all errors as objects
* Allows to define error types
* Allows to explicitly raise (throw) an error
* Allow to handle error types in a particular context
* Allow to propagate (re-throw) errors from the current context (inner block) to immediate outer context (outer block).
* Allow to specify code that executes ate the conclusion of some associated code

# Assignment07

In this assignment I’ve used the previous script on assignment06 replaced the last method to save data with binary files and added structured error handling around the areas where there is user interaction, type casting and file access operations.

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  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298 | *#------------------------------------------#*  *# Title: Assignment06\_Starter.py*  *# Desc: Working with classes and functions.*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *# Tiago Rodrigues, 2022-Nov-17. Updated File*  *# Tiago Rodrigues, 2022-Nov-27, Modified code to add binary data and error code handling*  *#------------------------------------------#*  *#TODone docstrings for every function*  *#TODone add error handling in functions*  **import** **pickle**  *# -- DATA -- #*  strChoice = '' *# User input*  lstTbl = [] *# list of lists to hold data*  dicRow = {} *# list of data row*  strFileName = 'CDInventory.dat' *# data storage file*  objFile = **None** *# file object*  *# -- PROCESSING -- #*  **class** **DataProcessor**:  *# TODone add functions for processing here*  @staticmethod  **def** add\_data(ID, Title, Artist):  *''''*  *Function to add data inside the table*    *Args:*  *ID (String): id*  *Title(String): cd Title*  *Artist(String): cd Artist*    *Returns:*  *lstTbl (list)*  *'''*  *# 3.3.2 Add item to the table*  **try**:  intID = int(ID)  dicRow = {'ID': intID, 'Title': Title, 'Artist': Artist}  lstTbl.append(dicRow)  **return** lstTbl  **except** **Exception** **as** e:  print('There was a error!')  print('Build in error info:')  print(type(e), e, e.\_\_doc\_\_, sep='**\n**')    @staticmethod  **def** del\_inventory():  *'''*  *Function to delete data from the table*    *Args:*  *None*    *Returns:*  *None*    *'''*  **try**:  intIDDel = int(input('Which ID would you like to delete? ').strip())  **except** **ValueError** **as** e:  print('That is not an integer!')  print('Build in error info:')  print(type(e), e, e.\_\_doc\_\_, sep='**\n**')  *# 3.5.2 search thru table and delete CD*  intRowNr = -1  blnCDRemoved = **False**  **for** row **in** lstTbl:  intRowNr += 1  **if** row['ID'] == intIDDel:  **del** lstTbl[intRowNr]  blnCDRemoved = **True**  **break**  **if** blnCDRemoved:  print('The CD was removed')  **else**:  print('Could not find this CD!')  **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.*  *Args:*  *file\_name (string): name of file used to read the data from*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*  *Returns:*  *None.*  *"""*  table.clear() *# this clears existing data and allows to load data from file*  **try**:  objFile = open(file\_name, 'rb')  **except** **FileNotFoundError** **as** e:  print('File not found.')  print('Build in error info:')  print(type(e), e, e,\_\_doc\_\_, sep='**\n**')  **try**:  data = pickle.load(objFile)  **for** line **in** range(0, len(data)):  table.append(data[line])  **except** **Exception** **as** e:  print('There was an error.')  print('Build in error info:')  print(type(e), e, e,\_\_doc\_\_, sep='**\n**')    objFile.close()    @staticmethod  **def** write\_file(file\_name, table):  *'''*  *#Function to write inside the file file\_name*    *Args:*  *file\_name: the name of the file*  *table: the table with data to insert into the file*    *Returns:*  *None*  *'''*  IO.show\_inventory(lstTbl)  strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()  *# 3.6.2 Process choice*  **if** strYesNo == 'y':  *# 3.6.2.1 save data*  *# TODone move processing code into function*  **try**:  objFile = open(strFileName, 'wb')  **except** **FileExistsError** **as** e:  print('File not found')  print('Build in error info:')  print(type(e), e, e.\_\_doc\_\_, sep='**\n**')  **for** row **in** lstTbl:  lstValues = list(row.values())  **try**:  lstValues[0] = str(lstValues[0])  **except** **Exception** **as** e:  print('There was a error!')  print('Build in error info:')  print(type(e), e, e.\_\_doc\_\_, sep='**\n**')  pickle.dump(lstTbl,objFile)  objFile.close()  **else**:  input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')  **pass**      *# -- 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*  *Args:*  *None.*  *Returns:*  *choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x*  *"""*  choice = ' '  **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:  choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()  print() *# Add extra space for layout*  **return** choice  @staticmethod  **def** show\_inventory(table):  *"""Displays current inventory table*  *Args:*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.*  *Returns:*  *None.*  *"""*  print('======= The Current Inventory: =======')  print('ID**\t**CD Title (by: Artist)**\n**')  **for** row **in** table:  print('**{}\t{}** (by:**{}**)'.format(\*row.values()))  print('======================================')  *# TODone add I/O functions as needed*  @staticmethod  **def** add\_iventory():  *'''*  *Function that request three inputs to user to add data in table*    *Args:*  *None*    *Returns:*  *strID (String): id*  *strTitle (String): cd Title*  *strArtist (String): cd Artist*  *'''*  **try**:  *# 3.3.1 Ask user for new ID, CD Title and Artist*  strID = input('Enter ID: ').strip()  strTitle = input('What is the CD**\'**s title? ').strip()  strArtist = input('What is the Artist**\'**s name? ').strip()  **except** **ValueError** **as** e:  print('Incorrect Character')  print('Build in error info:')  print(type(e), e, e.\_\_doc\_\_, sep='**\n**')  **return** strID, strTitle, strArtist        *# 1. When program starts, read in the currently saved Inventory*  *# check if the file exists before start the program*  **try**:  FileProcessor.read\_file(strFileName, lstTbl)  **except** **FileExistsError**:  print('The file **{}** doesn**\'**t exist!'.format(strFileName))  *#FileProcessor.read\_file(strFileName, lstTbl)*  *# 2. start main loop*  **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':  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...')  FileProcessor.read\_file(strFileName, lstTbl)  IO.show\_inventory(lstTbl)  **else**:  input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')  IO.show\_inventory(lstTbl)  **continue** *# start loop back at top.*  *# 3.3 process add a CD*  **elif** strChoice == 'a':  *# 3.3.1 Ask user for new ID, CD Title and Artist*  *# TODone move IO code into function*  *# 3.3.2 Add item to the table*  *# TODone move processing code into function*  strID, strTitle, strArtist = IO.add\_iventory()  lstTbl = DataProcessor.add\_data(strID, strTitle, strArtist)  IO.show\_inventory(lstTbl)  **continue** *# start loop back at top.*  *# 3.4 process display current inventory*  **elif** strChoice == 'i':  IO.show\_inventory(lstTbl)  **continue** *# start loop back at top.*  *# 3.5 process delete a CD*  **elif** strChoice == 'd':  *# 3.5.1 get Userinput for which CD to delete*  *# 3.5.1.1 display Inventory to user*  IO.show\_inventory(lstTbl)  *# 3.5.1.2 ask user which ID to remove*  DataProcessor.del\_inventory()  *# 3.5.2 search thru table and delete CD*  *# TODone move processing code into function*  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*  *# start loop back at top.*  FileProcessor.write\_file(strFileName, lstTbl)  *# 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:*  **else**:  print('General Error') |

## 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

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 4 - .txt file

Loading the data inside the file

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 5 - Loading data

Removing the select id from the list

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 6 - remove cd

## Using computer terminal

Adding more data to the list

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 7- Data added inside the list

Display de data inside list

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 8- Display current data in list

Save the data inside the .txt file.

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 9- Save data inside the file

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 10 - .txt file

Loading all the data inside the .txt file.

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 11 - Loading data

Removing the cd.

Uma imagem com texto

Descrição gerada automaticamente

Ilustração 12- remove cd

# Summary

In this assignment we introduce binary files and structured error handling which is very interesting because we can create our errors and display the message, we want instead of exiting the program. Here is the GitHub url:

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