**Kasisi Harris**

**2022-Aug-20**

**IT FDN 110 A Su 22: Foundations of Programming**

**Assignment\_07**

**GitHub link:** [kasisi79/Assignment\_07 (github.com)](https://github.com/kasisi79/Assignment_07)

Pickles, Errors, & Exceptions

# Introduction

This week, we were tasked with incorporating exception handling into our previous CD Inventory script. This is to increase the efficiency of potentially troubleshooting errors as well as allowing a more user-friendly response to any potential errors that may occur. I’ve chosen to place try and exception statements in areas where the file is opened as well as where the type of data that the user enters could generate an error.

# Error and Exception Handling Research

I found the courses description of error handling quite confusing at times, so in order to achieve some sort of clarity I consulted the oracle, aka the Google Search Engine, to find other potential sources of information on the topic. Corey Schafer, or CoreyMS.com on YouTube, offers a series of tutorials for several programming languages including python. His video, [*Python Tutorial: Using Try/Except Blocks for Error Handling*](https://www.youtube.com/watch?v=NIWwJbo-9_8), explained the use of Try, Except, Else, and Finally in an easy-to-understand way. I’ve bookmarked this channel for future consultation and highly recommend it for anyone else needing supplementary material for the course.

# Pickling Research

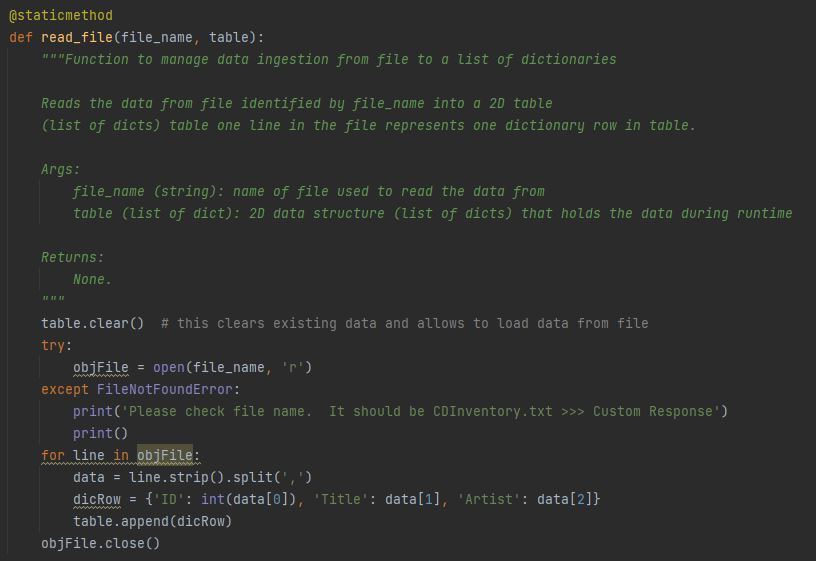
Another great supplement to the course material can be found via Tech Watt’s YouTube Channel. Two months ago, the content creator uploaded the video, [*Python Pickle Module. Save and Read From a File*](https://www.youtube.com/watch?v=nBYoVG0YL7A). Tech Watt does a great job taking his time to build on how to use pickling in python. I especially like this video because he makes a simple mistake in it, and easily rectifies, (concerning reading and writing). Again, I recommend this video for a quick understanding of the topic.

# Script Creation in PyCharm

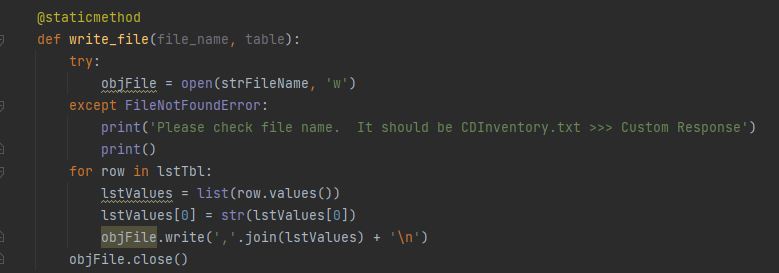
Besides the header change to denote the addition of exception handling, I’ve added code three parts of last week’s script: the File Processing functions read\_file, write\_file, and within the deletion choice as detailed below.

## File Errors

There could exist an instance where the file that the script is referring to could have changed. This exception could be caught by placing exception coding in the area of code that opens the file. Following this train of thought, I’ve placed exception statements that identify File Not Found Errors should the file name not match what is identified in the coding.



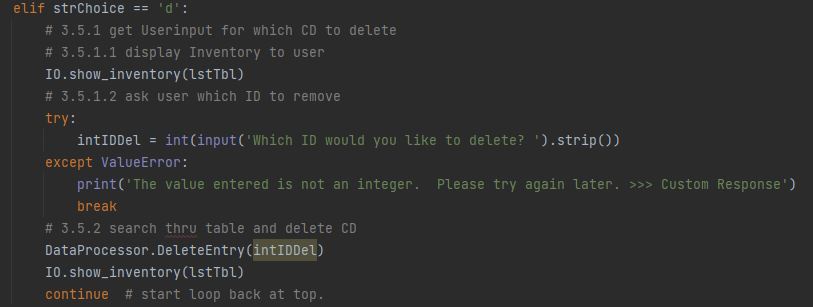
Listing - CDInventory.py - Exception - read\_file



Listing - CDInventory.py - Exception - write\_file

## Type Errors

Thinking about where the user might interact with the code that could result in an error was interesting. Much of last weeks code guards against error thwarting the program. For instance, should the user enter an item that is not identified in the menu select, the program will not continue and prompt the user to once again enter a selection. There is however, one area of code where the user could potentially cause an error as a result of their choice, the deletion selection. The user is asked to enter the ID of the record they wish to delete. The script expects an integer, and should the user supply a different value type, it will result in an error. To identify this I’ve incorporated a Value Error exception as depicted below.



Listing - CDInventory.py - Exception - Delete Entry

# Py Charm & Terminal Run Images

The below images detail both the PyCharm execution of the code as well as the Terminal Execution of the code.

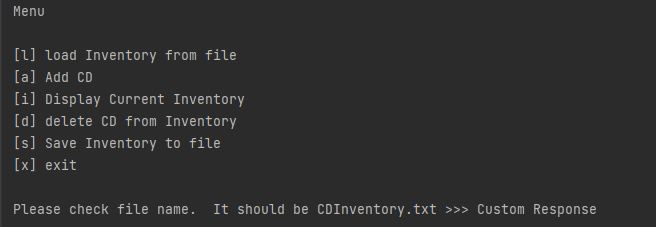


Figure - PyCharm Run - File Exception

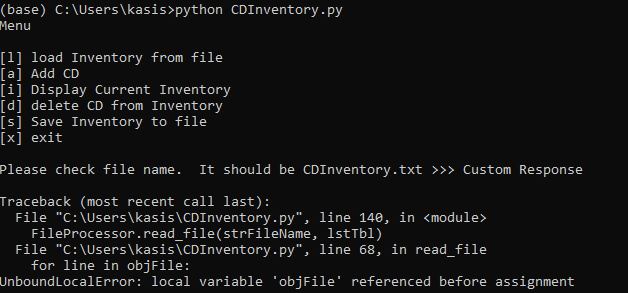


Figure - Terminal Run - File Exception

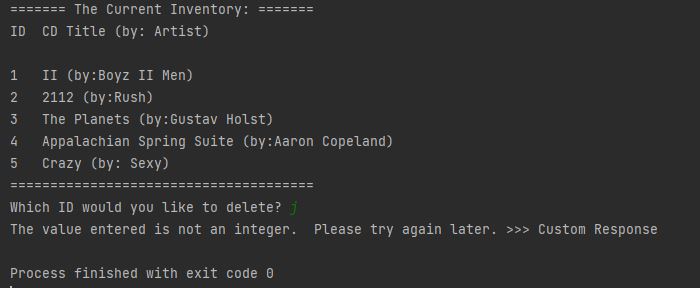


Figure - PyCharm Run - Type Exception

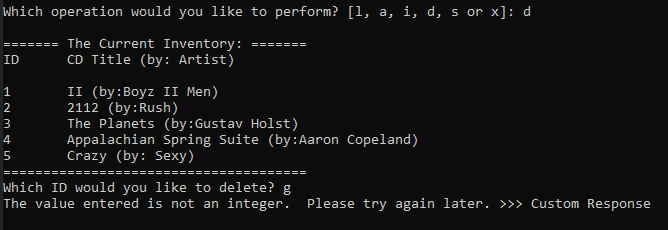


Figure - Terminal Run - Type Exception

# Summary

It is easy to see how Error & Exception Handling provides a more user-friendly troubleshooting experience. On a much higher level, it could provide verbiage that would help identify the exact reasoning and location of an error to expedite the error’s eradication. I look forward to building on this in future weeks.

# Appendix

## Listing of CDInventory.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 | *#------------------------------------------#*  *# Title: Assignment06\_Starter.py*  *# Desc: Working with classes and functions.*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *# KHarris, 2022-Aug-13, Added code to replace TODO tasks*  *# KHarris, 2022-Aug-20, Added Exception Handling to File Open and Write lines as well as*  *# Interger verification for deleting records.*  *#------------------------------------------#*  *# -- 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*  value1 = **None**  value2 = **None**  value3 = **None**  *# -- PROCESSING -- #*  **class** **DataProcessor**:  **def** AddInfoInput (value1, value2, value3):  strID = input('Enter ID: ').strip().format(value1)  strTitle = input('What is the CD**\'**s title? ').strip().format(value2)  strArtist = input('What is the Artist**\'**s name? ').strip().format(value3)  intID = int(strID)  dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}  lstTbl.append(dicRow)  **return** strID, strTitle, strArtist  **def** DeleteEntry(response):  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, 'r')  **except** **FileNotFoundError**:  print('Please check file name. It should be CDInventory.txt >>> Custom Response')  print()  **for** line **in** objFile:  data = line.strip().split(',')  dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}  table.append(dicRow)  objFile.close()  @staticmethod  **def** write\_file(file\_name, table):  **try**:  objFile = open(strFileName, 'w')  **except** **FileNotFoundError**:  print('Please check file name. It should be CDInventory.txt >>> Custom Response')  print()  **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*  *Args:*  *None.*  *Returns:*  *Return strChoice*  *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('======================================')  *# 1. When program starts, read in the currently saved Inventory*  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*  *# 3.3.2 Add item to the table*  DataProcessor.AddInfoInput(value1,value2, value3)  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*  **try**:  intIDDel = int(input('Which ID would you like to delete? ').strip())  **except** **ValueError**:  print('The value entered is not an integer. Please try again later. >>> Custom Response')  **break**  *# 3.5.2 search thru table and delete CD*  DataProcessor.DeleteEntry(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':  *# 3.6.2.1 save data*  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 safe:*  **else**:  print('General Error') |

# References

Python Tutorial: Using Try/Except Blocks for Error Handling. (2015). [YouTube Video]. In *YouTube*. https://www.youtube.com/watch?v=NIWwJbo-9\_8

‌*Python Pickle Module. Save and Read From a File.* (n.d.). Www.youtube.com. Retrieved August 20, 2022, from https://www.youtube.com/watch?v=nBYoVG0YL7A

‌