Evan Anderson

12/1/2021

Foundations of Programming, Python

Assignment07

# CD Inventory Program (Structured Error Handling)

## Introduction

The task of this assignment was modify the previous iteration of the CD Inventory Program by adding structured error handling around the areas where there is user interaction, type casting (string to int) or file access operations. Error handling is useful because it prevents data within the program from being lost (program terminating) and allows for custom responses to errors (i.e. a message stating why the input is not valid). In this document I will go through each error that was mitigated and the code used to do so.

## FileNotFoundError

The first error I addressed was the FileNotFoundError since the first action of the program is to open the 'CDInventory' text file and read it. In order to mitigate this error, I built out a try-except block that creates a 'CDInventory' text file if one does not already exist:

```
178 # 1. When program starts, read in the currently saved Inventory. If no inventory exists, a new file is create
179 try:
180     FileProcessor.read_file(strFileName, lstTbl)
181 except FileNotFoundError:
182     print('Text file does not exist! New text file named \'CDInventory.txt\' created.\n')
183     objFile = open(strFileName, 'w')
184     objFile.close()
185     FileProcessor.read_file(strFileName, lstTbl)
```

*Figure 1: FileNotFoundError #1*

Although this try-except block above makes sure a text file exists once the program begins, it does not prevent a FileNotFoundError if the text document gets deleted while the program is running. To mitigate this, I added a try-except block to the "Load Inventory from File" portion of the code:

```
199     # 3.2 process load inventory
200     if strChoice == 'l':
201         print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
202         strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled: ')
203         if strYesNo.lower() == 'yes':
204             print('reloading...')
205             try:
206                 FileProcessor.read_file(strFileName, lstTbl)
207                 IO.show_inventory(lstTbl)
208             except FileNotFoundError:
209                 print('Text file does not exist! New text file named \'CDInventory.txt\' created.\n')
210                 objFile = open(strFileName, 'w')
211                 objFile.close()
```

*Figure 2: FileNotFoundError #2*

This try-except block has the same function in that it creates a new text document when one is not found. This block is executed when the user chooses the 'Load Inventory from File' option. If the user tries loading a file that doesn't exist, it loads inventory from a newly created blank text document.

## Invalid Option (User-Defined Exception)

The next error that I addressed was a user-defined error caused by the user entering an input that is not included in the list of options. I completed the try-except block for this in the menu_choice class using a Boolean value:

```
132         choice = ' '
133         while True:
134             choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
135             OneOfMenuOptions = choice in ['l', 'a', 'i', 'd', 's', 'x']
136             if OneOfMenuOptions:
137                 break
138             try:
139                 if not OneOfMenuOptions:
140                     raise Exception('Invalid Option')
141             except Exception as e:
142                     print (e)
143         return choice
```

*Figure 3: User-Defined Exception*

## ValueError

The last error that needed addressing in the program was the ValueError resulting from a string being inputted as a CD ID (which the program requires to be an integer). To address this error, the program uses a try-except block to determine whether the user inputs are valid in both the 'Add CD' block and the 'Delete CD' block:

```
218     # 3.3 process add a CD
219     elif strChoice == 'a':
220         # 3.3.1 Ask user for new ID, CD Title and Artist
221         try:
222             strID, strTitle, stArtist = IO.input_cd() #unpack and call at the same time
223             # 3.3.2 Add item to the table
224             DataProcessor.add_cd(strID,strTitle,stArtist)
225             continue  # start loop back at top.
226         except ValueError:
227             print('\nINVALID INPUT DATA\nID MUST BE A NUMBER\n')
```

*Figure 4: ValueError #1*

```
234     # 3.5 process delete a CD
235     elif strChoice == 'd':
236         # 3.5.1 get Userinput for which CD to delete
237         # 3.5.1.1 display Inventory to user
238         IO.show_inventory(lstTbl)
239         # 3.5.1.2 ask user which ID to remove
240         try:
241             intIDDel = int(input('Which ID would you like to delete? ').strip())
242         # 3.5.2 search thru table and delete CD
243             blnCDRemoved = False
244             DataProcessor.delete_cd()
245         except ValueError:
246             print('\nINVALID INPUT DATA\nID MUST BE A NUMBER\n')
247         IO.show_inventory(lstTbl)
248         continue  # start loop back at top.
```

*Figure 5: ValueError #2*

## Testing the Program (In Spyder)

Before starting the program, I deleted the text document from the Assignment_07 folder to see how the program would respond. As expected, the program notifies the user that no source file exists and creates a new one to work off of:

```
In [46]: runfile('C:/_FDProgramming/Mod_07/Assignment_07/CDInventory.py', wdir='C:/_FDProgramming/Mod_07/
Assignment_07')
Text file does not exist! New text file named 'CDInventory.txt' created.

Menu
```

*Figure 6: Starting the program with no source file*

While the program is running, I then deleted the text file to see what would happen when I try loading it to the program. As expected, the program does not run into an error. Instead it creates a new text document and loads the data to the program:

```
Which operation would you like to perform? [l, a, i, d, s or x]: l
WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.

type 'yes' to continue and reload from file. otherwise reload will be canceled: yes
reloading...
======= The Current Inventory: =======
ID  CD Title (by: Artist)

=======================================
Menu
```

*Figure 7: Loading a text document that doesn't exist*

The last error that needed addressing was the ValueError resulting from a non-integer being inputted as a CD ID within the 'Add CD' and 'Delete CD' functions of the program. When I entered 'Test' as the ID for adding a CD, the program showed the following response:

```
Which operation would you like to perform? [l, a, i, d, s or x]: a

Enter ID: TEST

What is the CD's title? Flower Boy

What is the Artist's name? Tyler the Creator

INVALID INPUT DATA
ID MUST BE A NUMBER

Menu
```

*Figure 8: ValueError #1*

When I entered 'Test' as the ID for deleting a CD, the program showed the following response:

```
Which ID would you like to delete? TEST

INVALID INPUT DATA
ID MUST BE A NUMBER
```

*Figure 9: ValueError #2*

## Testing in Terminal

I started the program with no source text document and the program created a new one automatically.

```
(base) C:\_FDProgramming\Mod_07\Assignment_07>python CDInventory.py
Text file does not exist! New text file named 'CDInventory.txt' created.

Menu
```

*Figure 10: Creating a new file at start of program*

I then deleted the source document and tried loading it to the program.

```
Which operation would you like to perform? [l, a, i, d, s or x]: l
WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.
type 'yes' to continue and reload from file. otherwise reload will be canceled: yes
reloading...
Text file does not exist! New text file named 'CDInventory.txt' created.

Menu
```

*Figure 11: Creating file when there is none to load*

I then tried adding a CD with a string as the ID:

```
Which operation would you like to perform? [l, a, i, d, s or x]: a
Enter ID: TEST
What is the CD's title? Flower Boy
What is the Artist's name? Tyler the Creator

INVALID INPUT DATA
ID MUST BE A NUMBER

Menu
```

*Figure 12*

Lastly, I tried deleting a CD with a string as the ID:

```
Which operation would you like to perform? [l, a, i, d, s or x]: d
======= The Current Inventory: =======
ID      CD Title (by: Artist)

======================================
Which ID would you like to delete? TEST

INVALID INPUT DATA
ID MUST BE A NUMBER
```

*Figure 13*

All done! All the errors have been successfully avoided and the program never needed to force quit.

## Summary

In summary, we avoided various forms of errors by building out try-except blocks. This allowed the program to continuously run despite incorrect user inputs or absent source files.