

Muhammad Mannan

August 24 2021

IT FDN 110

Assignment07

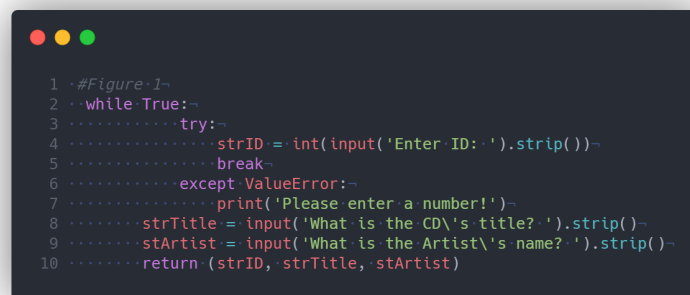
Module 07

Introduction:

The goal of this assignment was to get familiar with structured error handling as well learn about a new way of storing data permanently in the form of a .dat file or in binary. We were then asked to implement these features into our project from the last assignment

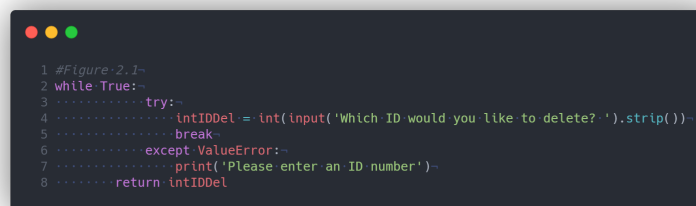
Implementing structured error handling:

In many places in our python script the user is asked to input various pieces of information which is used to adjust the inventory of our CD list. Structured error handling allows us to prevent our code from breaking or throwing a script ending error when the user enters a response that may not be the type of data we are asking them for. The main areas we are concerned with error handling in our script are where we ask for the user's input in adding the CD's entry number, asking which number CD we would like to remove, and whether or not our script can find and access our inventory file. When asking for a user to enter an ID or CD entry number we are only looking for an integer value so we need to make sure that our user doesn't enter a string or any other non-numeric character which breaks the code. We can use structured error handling to deal with issues by throwing a ValueError (see figure 1).



```
1 #Figure-1~
2 while True:~
3     try:~
4         strID = int(input('Enter ID: ').strip())~
5         break~
6     except ValueError:~
7         print('Please enter a number!')~
8     strTitle = input('What is the CD\'s title? ').strip()~
9     strArtist = input('What is the Artist\'s name? ').strip()~
10    return (strID, strTitle, strArtist)
```

We fall into the same issue when asking the user which entry CD they would like to delete. We can also solve this by throwing a ValueError as well when asking for the entry number they would like to delete (see figure 2.1).



```
1 #Figure-2.1~
2 while True:~
3     try:~
4         intIDDel = int(input('Which ID would you like to delete? ').strip())~
5         break~
6     except ValueError:~
7         print('Please enter an ID number')~
8     return intIDDel
```

We also need to consider the fact that the user may try and enter an entry to delete that already isn't in the inventory. We can deal with this by throwing an UnboundLocalError (see figure 2.2)

```

1 #Figure 2.2~
2 try:~
3 .....intRowNr -= 1~
4 .....for row in cdList:~
5 .....intRowNr += 1~
6 .....if row['ID'] == cdNumber:~
7 .....del cdList[intRowNr]~
8 .....CDRemoved = True~
9 .....break~
10 .....return CDRemoved~
11 .....except UnboundLocalError:~
12 .....print("Please pick a valid entry to remove")

```

The final part where we need to add structured error handling is when we need to access the file. We can't always know whether or not the script has already run and this means we can't always know if the file containing our permanently stored data already exists. In order to do this we can add structured error handling to check and see if the file already exists, and if it doesn't it should then create the file. We can do this by throwing `FileNotFoundError` (see figure 3).

```

1 #Figure 3~
2 try:~
3 ....FileProcessor.read_file(strFileName, lstTbl)~
4 except FileNotFoundError:~# the file is created if it doesn't already exist~
5 ....FileProcessor.write_file(strFileName, lstTbl)

```

Saving data in binary:

We also cover and utilize a new way of permanently storing data in this assignment and we do by storing it in a .dat file in binary form. Since the majority of the time the client/user side is viewing the data while the script is running and editing the data while it's in memory we can store the data in a smaller binary file. However in order to do this we need to be able to read and write our data to a file in binary form. In order for us to read a binary file we need to import `pickle` which is a dictionary that helps us work with binary data (see figure 4.1)

```

1 #Figure 4.1~
2 import pickle

```

and after that we can use a for loop to loop through all our lines of binary data and have our script update/load up the data for the user in a more readable format (see figure 4.2).

```
1 #Figure-4,2~
2 objFile = open(file_name, 'rb')~
3 ..... pcklFile = pickle.load(objFile)~
4 ..... for line in pcklFile:~
5 ..... data.append(line)~
6 ..... objFile.close()
```

Next we need to be able to write our file in binary form as well. We can do so by dumping our data into the file (see figure 5).

```
1 #Figure-5~
2 objFile = open(file_name, 'wb+')~
3 ..... pickle.dump(data, objFile)~
4 ..... objFile.close()
```

Notice how the way we open the file is followed by what we want to do with it, either read (r) or write (w) followed by (b) to signify we are doing this in binary.

Summary:

After putting all our code together here is what it looks like running in spyder (see figure 6) and in terminal (see figure 7) as well as their outputs (see figure 8).

```
Python 3.8.8 (default, Apr 12 2021, 15:00:02) [MC v.1016 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 7.22.0 -- An enhanced Interactive Python.
>>> runfile('E:/UW PYTHON/Mod_07/Assignment07.py', wdir='E:/UW PYTHON/Mod_07')
Menu

[1] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

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

Enter ID: 1
What is the CD's title? uno
What is the Artist's name? green day
===== The Current Inventory: =====
ID CD Title (by: Artist)
1 uno (by:green day)
=====
Menu

[1] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

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

Enter ID: 2
What is the CD's title? bleed america
What is the Artist's name? jimmy eat world
===== The Current Inventory: =====
ID CD Title (by: Artist)
1 uno (by:green day)
2 bleed america (by:jimmy eat world)
=====
Menu

[1] load Inventory from file
[a] Add CD
[i] Display Current Inventory
```

Figure 7

```

Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\usman>E:

E:\>cd "UM PYTHON"

E:\UM PYTHON>cd Mod_07

E:\UM PYTHON\Mod_07>python Assignment07.py
Menu

[1] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [1, a, i, d, s or x]: 1

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)
1      uno (by:green day)
3      gone (by:royals)
=====
Menu

[1] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [1, a, i, d, s or x]: d

===== The Current Inventory: =====
ID      CD Title (by: Artist)
1      uno (by:green day)
3      gone (by:royals)
=====
Which ID would you like to delete? 3
The CD was removed.
===== The Current Inventory: =====
ID      CD Title (by: Artist)
1      uno (by:green day)

```

Figure 8

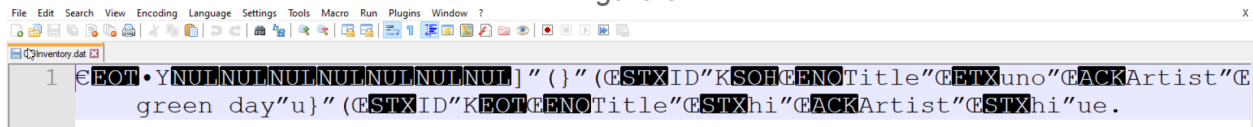


Figure 9

In the end adding structured error handling helps our code run smoother on the client side and utilizing the process of reading and writing our data to a file in binary form helps better optimize out code and provides a more efficient way of storing data.