James Crockett
2021-Nov-28
Foundations of Programming (Python)
Assignment 07

Exception Handling and Pickling

Introduction

This week added new concepts of exception handling and how this can be implemented to avoid terminating a program prematurely and losing data from memory for that program session. Week 7 has provided the advantages of pickling and how it can be utilized with file access operations when dumping or loading objects from a binary file.

Pseudo Code

I added #TODO comments in areas of code where exception handling would need to be implemented. This served as visual markers of areas that I needed to work through as I was implementing and testing the code changes. Additional comments were added for learning moments where I tried a new argument after reading about it from reference material on the web or a Youtube video covering the subject matter I wanted to consume for a particular challenge that was presented by Spyder console output.

Initial Approach

I decided to work through the practice labs to gain some more knowledge before starting Assignment 7. I was able to complete Lab A, and Lab C. I implemented most of Lab B, but never dove into implementing the program argument requirements because I wanted to shift my time into working through the final Assignment for week 7. I plan on going back to system arguments usage at a later time.

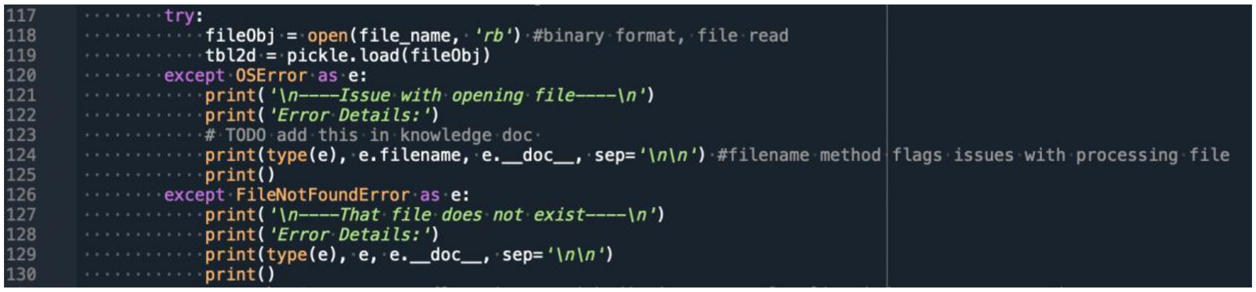
Exception Handling

Exception handling is a way for to incorporate error handling for different events that would normally crash or terminate a program, this normally encapsulated in try, exception blocks, or variations of these blocks. The website https://www.tutorialsteacher.com/python/exception-handling-in-python [1] demonstrates the different ways these exception blocks can be constructed to achieve the preferred results. I found the code blocks with "except" and "finally" keywords helpful to demonstrate that I can continue with a certain logic if exceptions were or weren't triggered depending on the keyword used as I constructed these different blocks within my own scripts.

Examples were already offered on general type error exceptions from the class materials, but I hadn't read through references on how to provide exception handling for file access operations such as .open() or .close(). The website https://docs.python.org/3/library/exceptions.html [2] was helpful is showing the

[1] Retrieved 2021-Nov-28.
[2] Retrieved 2021-Nov-28.

OSError filename attribute handles processing issues with files with open or close actions. Figure 1 below shows implementation of OSError handling in lines 120 & 124.



```
117         try:
118             fileObj = open(file_name, 'rb') #binary format, file read
119             tbl2d = pickle.load(fileObj)
120         except OSError as e:
121             print('\n----Issue with opening file----\n')
122             print('Error Details:')
123             # TODO add this in knowledge doc
124             print(type(e), e.filename, e.__doc__, sep='\n\n') #filename method flags issues with processing file
125             print()
126         except FileNotFoundError as e:
127             print('\n----That file does not exist----\n')
128             print('Error Details:')
129             print(type(e), e, e.__doc__, sep='\n\n')
130             print()
```

Figure 1 - Exception handling for open() using OSError

Pickling, Binary Packaging

Pickle is a python object that can be used to package and unpackage different types of data or objects into or from binary formats. Once something is pickled, it is placed into a binary format that is typically more compact that the non-binary format. Binary format is not as easily human readable. Pickling is not a secure way to save information because it can still easily be reconstructed.

I enjoyed the website: https://blog.quantinsti.com/pickle-python/[3] because it conveyed the practical uses of pickling along with the advantages for real world scenarios. Saving large amounts of data from memory into a compact binary data file to be access later saves space, access time, and provides the convenience of work with that object at a later date.

Challenges of Changing Existing Code

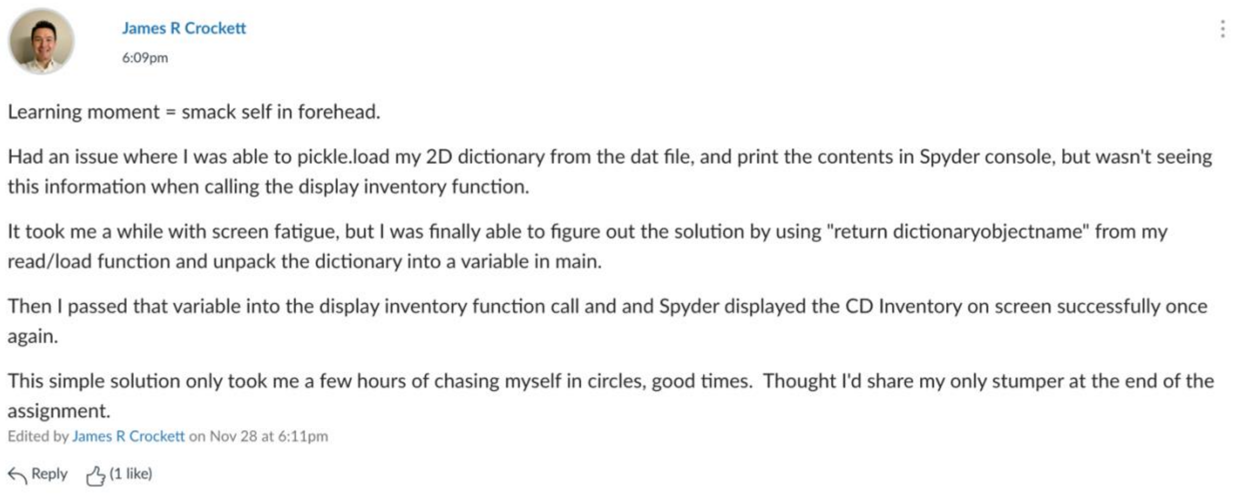
The main challenges of changing existing code is going in with the assumption that all other parts of the script will behave the same once new changes are introduce, in this case the pickle.dump and pick.load of binary data.   The original script from week 6 was designed to work with string data from a text file and to convert that data into a table when reading.

The difference with the changes in week 7 to the week 6 script was that pickle.load "unpickled" the object as a complete 2D dictionary, or list of dictionaries that contained the information of the CD Inventory. The other portion inside of the read function where the pickle.load was contained still had logic to take string content from a text file, so this was causing errors to display in runtime because there was no text file for this logic to work with.

Amanda from Discord had mentioned to me in chat that this unpickled data was already a 2D dictionary, and that set me on the right path and allowed me to step back and reassess the bigger picture of the issues I needed to address with the recent code changes introduced into the week 7 script.

I posted my findings in the Canvas discussion forums to share my learning experience with other students.

[3] Retrieved 2021-Nov-28.


James R Crockett

6:09pm

Learning moment = smack self in forehead.

Had an issue where I was able to pickle.load my 2D dictionary from the dat file, and print the contents in Spyder console, but wasn't seeing this information when calling the display inventory function.

It took me a while with screen fatigue, but I was finally able to figure out the solution by using "return dictionaryobjectname" from my read/load function and unpack the dictionary into a variable in main.

Then I passed that variable into the display inventory function call and and Spyder displayed the CD Inventory on screen successfully once again.

This simple solution only took me a few hours of chasing myself in circles, good times. Thought I'd share my only stumper at the end of the assignment.

Edited by James R Crockett on Nov 28 at 6:11pm

Figure 2-Code Change Challenge and Resolution.

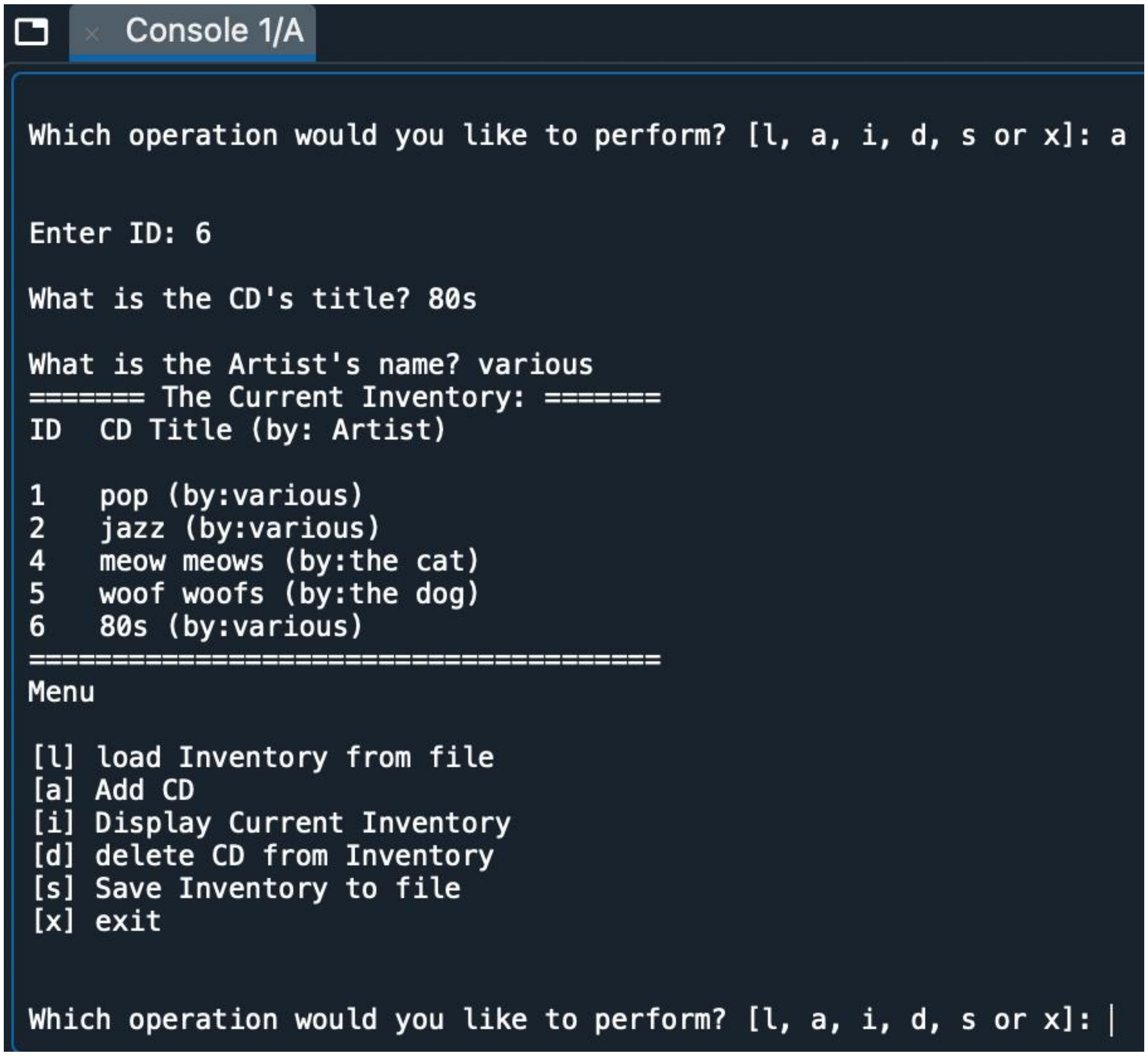
Summary

The biggest take away from the week 7 assignment is to be aware that even what can be considered a minor code change to an existing script can have a ripple effect on other parts of the program that can have unintended outcomes. Working through this part of the process of troubleshooting, and stepping through the code really requires quite a bit of thought with an understanding of what each part of the code is doing and what other parts of the code is "expecting" to receive. Format of data, how that data is passed, and received all factor in the end behavior.

Appendix

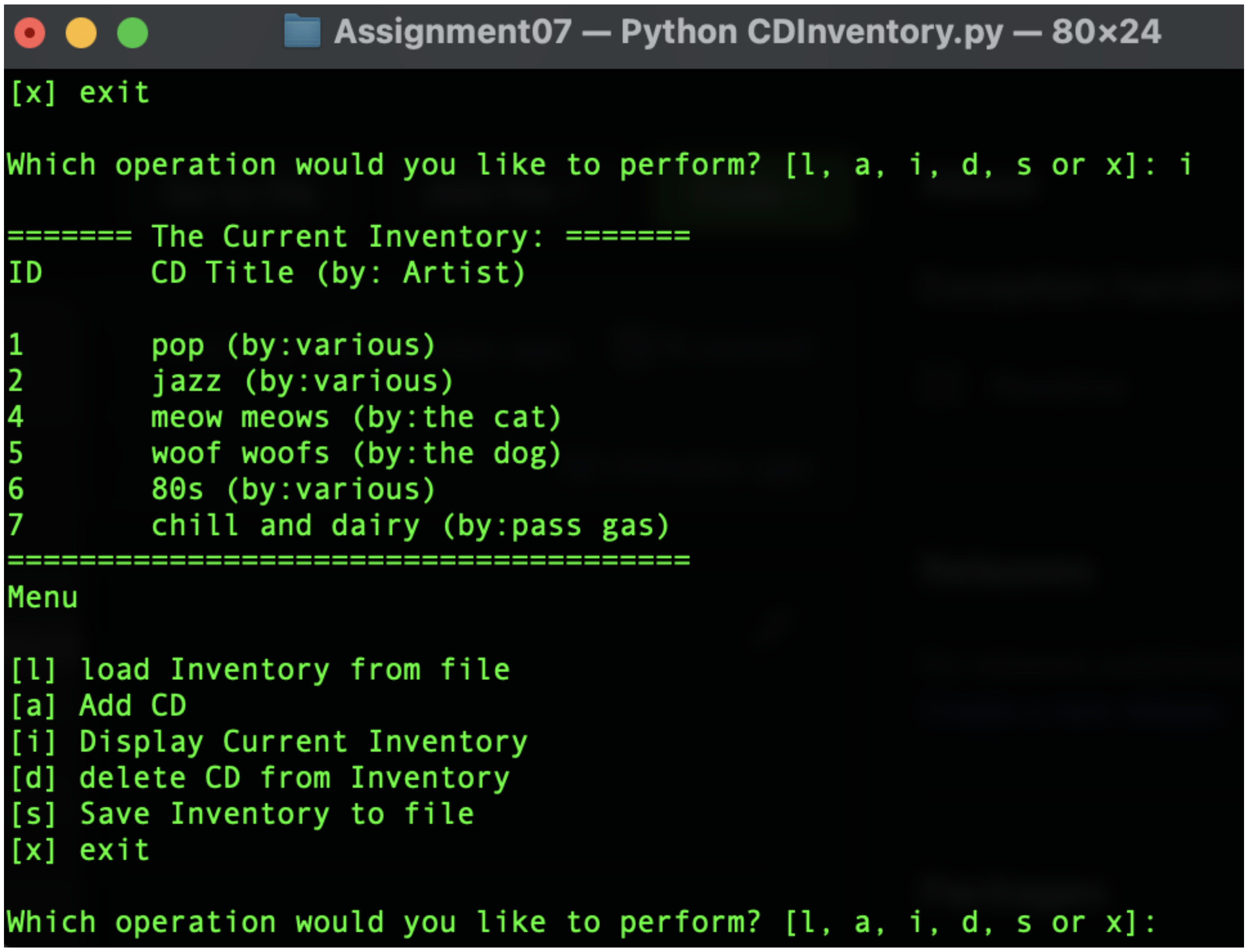
The following pages in this appendix show screen captures of:

- Figure 5, Spyder Console code execution.
- Figure 6, Mac Terminal code execution.
- Github, Assignment 7 Repository: https://github.com/jamescisonline/Assignment_07



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


Enter ID: 6

What is the CD's title? 80s

What is the Artist's name? various
======= The Current Inventory: =======
ID  CD Title (by: Artist)

1   pop (by:various)
2   jazz (by:various)
4   meow meows (by:the cat)
5   woof woofs (by:the dog)
6   80s (by:various)
======================================
Menu

[l] 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]:
```

Figure 3- Spyder console runtime.



```
[x] exit

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

======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       pop (by:various)
2       jazz (by:various)
4       meow meows (by:the cat)
5       woof woofs (by:the dog)
6       80s (by:various)
7       chill and dairy (by:pass gas)
======================================
Menu

[l] 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]:
```

Figure 4 - Mac Terminal runtime.[4]