Armando Gomez

2021-Nov-28

Foundations of Programming (Python)

Assignment07

Knowledge Document

# Introduction

First I loaded the Assignment06 CDInventory.py and renamed it and put it in the new Assignment07 folder as CDInventory.py, for the code see Appendix. I also updated docstrings and added my involvement in the header. Four basic parts to this assignment. Part 1, research exception handling in python. Part 2, research pickling in python. Part 3, add structured error handling around the areas where there is user interaction, type casting(string to int) or file access operations. Part 4, modify the permanent data store to use binary data.

# Part 1, Research exception handling in python

[Python website](https://docs.python.org/3/tutorial/errors.html) (external reference)[[1]](#footnote-1) was decent but the code is hard to read, very academic, very generalist, not a lot of examples.

[Guru99 website](https://www.guru99.com/python-exception-handling.html) (external reference)[[2]](#footnote-2) was terrible, it appeared that he did the bare minimum to show graphical and codified examples and just copied and pasted information from the python source website.

[Programiz website](https://www.programiz.com/python-programming/exception-handling) (external reference)[[3]](#footnote-3) was my favorite, this included a youtube video that he made, and then went into very deep detail with lots of very easy to read code examples. Then at the end he has related tutorials linked on there, it’s amazing.

# Part 2, Research pickling in python

[Python website](https://docs.python.org/3/tutorial/errors.html) (external reference)[[4]](#footnote-4) was actually my favorite one this time because it was written really well, giving the background of it by comparing it with other protocols and then going in to the formatting and syntax (module interface). I found that knowing how this module works is really the meat and potatoes. But it can’t stand on it’s own, I needed another website for specific examples.

[Geeks for Geeks website](https://www.geeksforgeeks.org/understanding-python-pickling-example/) (external reference)[[5]](#footnote-5) was ok but it forced you to log in or create an account, even with a popup blocker I don’t like that. I did like the easy to read examples of code they had on there and the number of different examples was really good.

[Real Python website](https://realpython.com/python-pickle-module/) (external reference)[[6]](#footnote-6) was really nice, the very different easy to read format has become one of my favorites and I really like the explanations with examples that just keep on coming.

# Part 3, Add Structured Error Handling

Immediately when structured error handling came to my mind, I thought about where errors usually come from (users). Therefore I went directly to inputs. The first input I thought about was the ID number when the user selects to add a new CD. What if the user a letter or a decimal number (not an integer)? Which is what we need for our code to work. In my new\_ID() method (line 190) I added a while True loop with try making the string input by the user (strID) into an integer (intID) and the exception (if it’s not an integer) would print “Please input an integer for the ID number”.

The next thought about error handling came to mind with respect to whether the file existed or not. Therefore I went to the FileProcessor class (line 68) and in the read\_file method (line 72) I added a Try to open the file to read, and exception FileNotFoundError to print the built in error information from files\_09.py example in the lectures. I added the same logic to the write\_file method (line 106).

# Part 4, Modify the permanent data store to use binary data

At first it seemed pretty straight forward, I added import pickle (line 10), I changed the strFileName (line 16) from “.txt” to “.dat”, changed ‘r’ to ‘rb’ (for reading binary in line 89), and changed ‘w’ to ‘wb’ (for writing binary in line 122), then in writing instead of writing the objFile dictionary row into the table (lstTbl) I pickle.dump the values into the file (line 127). All of that was great, then I got to reading the data, boy that was not so straight forward. After many headaches, a classmate suggested I print out the pickled data from the objFile. That’s when I realized and remembered that the pickled data is being read line by line as a list! So I just implemented a while loop to make the dicRow from the pickled data load and append it to the table until done. Bonus points because I implemented another error handling for ignoring when it’s done looping through the indexed data so I don’t get an index error.

# Summary

I really had fun playing with this code because for the second time this was not so difficult but yet still challenging because the code was written already. I really enjoyed bouncing ideas off of classmates when I got stuck. I liked their different ways of thinking about these problems and the different solutions we all tried. It really doesn’t appear to be one and only one best way to do these things. Pickling is really cool, although I still hate pickles.

# Appendix

<https://github.com/exterminar/Assignment_07>

# Figures

A screenshot of a computer

Description automatically generated with medium confidence

Figure 1, Screenshot of my script running in Spyder working on my computer

Text

Description automatically generated

Figure 2, Screenshot of my script running in Terminal working on my computer

1. <https://docs.python.org/3/tutorial/errors.html> Retrieved 2021-Nov-28 [↑](#footnote-ref-1)
2. <https://www.guru99.com/python-exception-handling.html> Retrieved 2021-Nov-28 [↑](#footnote-ref-2)
3. <https://www.programiz.com/python-programming/exception-handling> Retrieved 2021-Nov-28 [↑](#footnote-ref-3)
4. <https://docs.python.org/3/tutorial/errors.html> Retrieved 2021-Nov-28 [↑](#footnote-ref-4)
5. <https://www.geeksforgeeks.org/understanding-python-pickling-example/> Retrieved 2021-Nov 28 [↑](#footnote-ref-5)
6. <https://realpython.com/python-pickle-module/> Retrieved 2021-Nov-28 [↑](#footnote-ref-6)