Palmer Bandy
August 26, 2020
IT FDN 110 B Su 20
Assignment 07

Pickling, Exception Handling, & The CD Inventory Program Continued

Introduction

In this assignment, I will discuss what pickling is and why it is useful. I will also go over exception handling in Python and one way to implement it. Then, I will discuss the steps I took to modify the CDInventory.py program to add pickling and exception handling.

Pickling

In Python, "pickling" refers to the process of serializing complex data via the "Pickle" module and storing it in binary format.[1] One example of how this is useful is when you want to store data types specific to Python, like lists or dictionaries—you could theoretically store the characters that represent those data types in a text file, but that could result in unforeseen issues: pickling actually serializes that data type and stores it in binary format such that when you read it back—"unpickle" it—the data type is the same.[2] I found the following articles helpful for learning about Python pickling:

- https://docs.python.org/3/library/pickle.html
- https://realpython.com/python-pickle-module/
- https://www.tutorialspoint.com/python-pickling

Exception Handling

In any programming language, it is inevitable that there will be some kind of error in the program execution. Many languages have built-in messages that are printed out when an error occurs, but it is often desirable to customize error messages, especially for user-facing programs, and you can do this by handling possible exceptions.[3] A common pattern in Python is to use a `try` and `except` block, where the code you intend to execute under "happy path" conditions is placed in the `try`

[1] https://realpython.com/python-pickle-module/, retrieved 2020-08-26
[2] Ibid., retrieved 2020-08-26
[3] https://realpython.com/python-exceptions/, retrieved 2020-08-26
[3] https://realpython.com/python-exceptions/, retrieved 2020-08-26

block, and the potential errors are handled in one or more `except` blocks. I found the following articles helpful for learning about Python exception handling:

- https://docs.python.org/3/library/exceptions.html#Exception
- https://realpython.com/python-exceptions/
- https://www.tutorialspoint.com/python/python_exceptions.htm

The CD Inventory Program Continued

This assignment's task was to modify last week's file to handle exceptions, and also to read and write the data stored to disk in binary form rather than plain text. I started out by changing the `read_file_into_memory` and `write_file` functions to read and write the data in binary format using the Pickle module. It was straightforward to modify the `write_file` function to do this, but I ran into a few errors while trying to make sure the `read_file` function worked properly—I was trying to open the file in binary form and then loop through it line by line, which resulted in an EOFError. I went to Stack Overflow and found an alternate way to read in the data "line by line" by using a `while` loop combined with a `try` and `except` block[4]; the result looks like this:

```
try:
    # Approach for loading pickled data adapted from
    # https://stackoverflow.com/questions/20716812/saving-and-loading-multiple-objects-in-pickle-file
    with open(strFileName, 'rb') as objFile:
        while True:
            try:
                unpickledLine = pickle.load(objFile)
                data = unpickledLine.strip().split(',')
                dictRow = {'id': data[0], 'title': data[1], 'artist': data[2]}
                table.append(dictRow)
            except EOFError:
                break
except FileNotFoundError as e:
    print(f'No file named {strFileName} was found.\n', e)
return table
```

Figure 1 - Unpickling Multi-line File

Once I had that working properly, I moved on to adding exception handling in various places in my program. The first place I chose to add some exception handling was in the `ask_user_for_input` function—I also implemented some feedback about this function from last week's assignment. I actually struggled a bit to think of what kind of actual execution errors might occur, since all inputs are strings and it seems unlikely that a user would type something that would cause an error at the time of input. One thing that I did consider was that the user may type CMD + C (or CTRL + C on Windows) to stop program execution, so I added an exception to handle that case. Then I decided to add a custom exception (I am not sure this is truly a custom exception, but I call it that because I am forcing an exception based on conditions that I defined) that prohibits certain characters and

[4] https://stackoverflow.com/questions/20716812/saving-and-loading-multiple-objects-in-pickle-file, retrieved 2020-08-26

raises a `ValueError` exception if that error case occurs. Specifically, if someone were to include a double quotation mark or a comma in their inputs—two characters that merit some kind of special handling when treating the data file as if it were in CSV format, which is what I assume elsewhere in the program—I raise an error and tell them that these are invalid characters.

I also added exception handling to the `read_file` and `write_file` functions that catches either a `FileNotFound` error for reading when the file does not exist, or a `PermissionError` when the user does not have sufficient permissions to write to the file in question (e.g., if the file has been created by the program and then the permissions for that file are changed to be read-only, the program will no longer be able to write to the file anymore).

Here is the program running in Spyder:

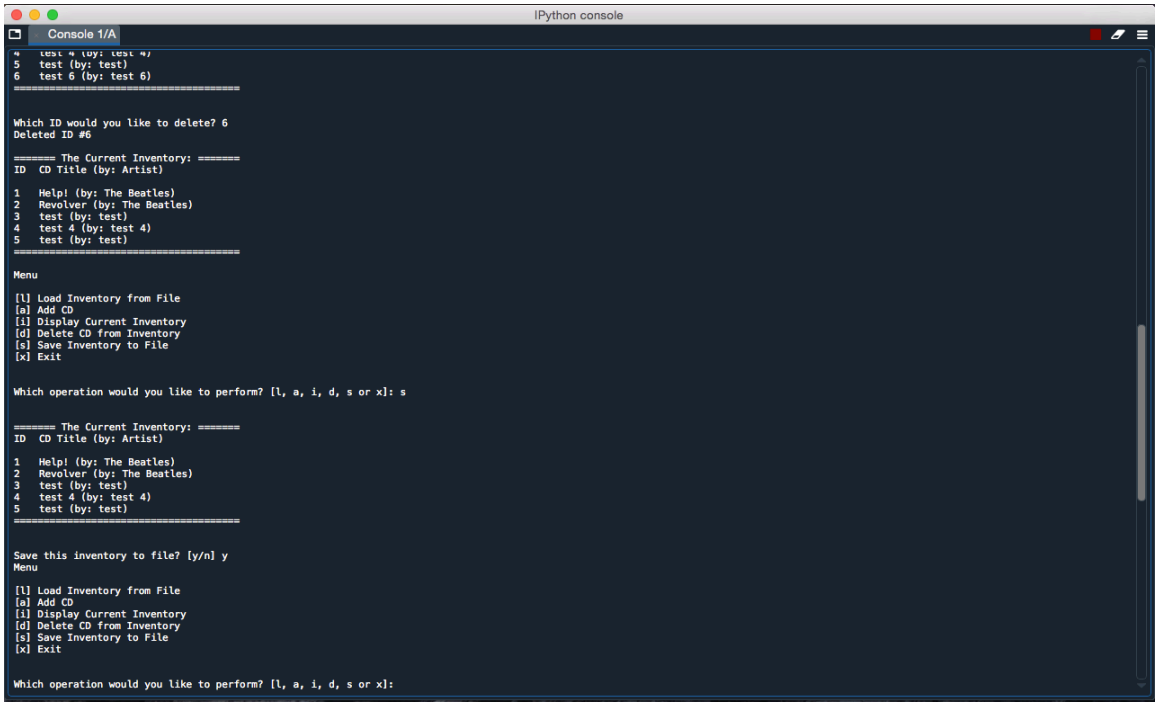
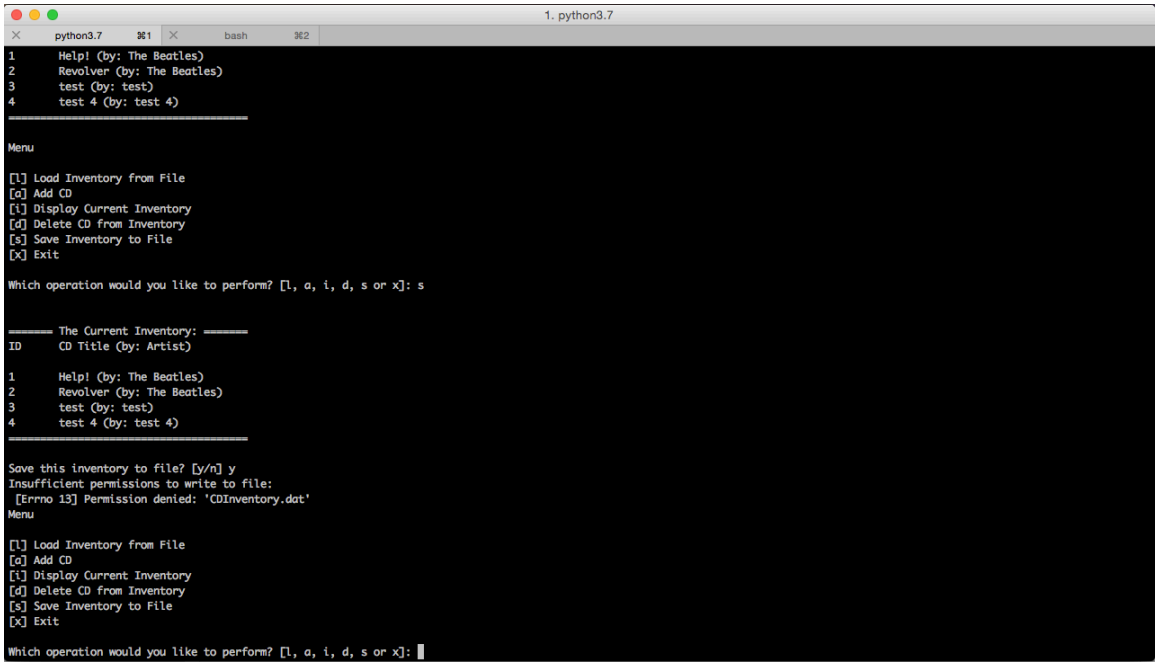


Figure 2 - Running CDInventory.py in Spyder

And here is the program running in my terminal:



```
1. python3.7
python3.7 ⌘1    bash ⌘2
1       Help! (by: The Beatles)
2       Revolver (by: The Beatles)
3       test (by: test)
4       test 4 (by: test 4)
======================================

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]: s


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

1       Help! (by: The Beatles)
2       Revolver (by: The Beatles)
3       test (by: test)
4       test 4 (by: test 4)
======================================

Save this inventory to file? [y/n] y
Insufficient permissions to write to file:
 [Errno 13] Permission denied: 'CDInventory.dat'
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 - Running CDInventory.py in Terminal

Summary

For this assignment, I talked about what pickling is and why it is useful. I also briefly covered what exception handling is and one way to implement it in Python. Then, I discussed the steps I took to update the CDInventory.py program to use pickling, store the data in binary format, and also implement exception handling in a few places. Here is the link to my Github repository for Assignment_07: https://github.com/palmermbandy/Assignment_07.