Nary Hang

March 11, 2022

Introduction to Programming (Python)

Assignment 07

How to modify an existing CD Inventory program with Structured Error Handling and Pickling

# Introduction

The objective for this seventh assignment is focused on modifying an existing CD Inventory program with structured error handling and pickling. The process to complete the assignment includes using existing code from last week assignment 6 and we were tasks to manipulate it to add structured error handling and pickling. We are also, required to research online on exception handling and pickling features. This knowledge document will detail my research and the steps taken to modify the program.

# Research

In researching exception handling, I’ve found a website that explained it in a simplest way. Reference website: <https://www.geeksforgeeks.org/python-exception-handling/>. There are two different statements: try/except statement and try/else statement. The try/except statement catches specific exceptions and in return prints out an error statement. The try/else statement is when try does not trigger an exception and in return prints out else statement. I use the following link for a list of built-in exceptions: <https://docs.python.org/3/library/exceptions.html>.

After research pickling, the following YouTube video provided a great explanation: <https://www.youtube.com/watch?v=Pl4Hp8qwwes>. According to the video, pickling is taking an object (list or dictionary) saving it to a serializing file. The file can be unpickled. The host provided a great example on how to use the code. See screenshot below. I will be using this example for the assignment and reference Friday lecture.

**Figure 1:**

*Screenshot from a YouTube video showing an example of pickling*

Text

Description automatically generated

# Steps Involved

After opening script in Anaconda Spyder, we were tasked with modifying an existing CD Inventory program with structured error handling and pickling.

The following steps were taken:

1. The first step that I took was adding error handling to the I/O of def user\_inputs when asked to enter an ID. I added a float() to the input. A float raises an exception, ValueError. If an integer is not entered in, it will prompt a statement. See below for the following code:

**def** user\_inputs():

validID = **False**

**while** **not** validID:

**try**:

*# used float for non digits*

strID = float(input('Enter ID: ').strip())

**except** **ValueError**: *# inappropriate value error*

print('That was not a number!')

**else**:

validID = **True**

strTitle = input('What is the CD**\'**s title? ').strip()

strArtist = input('What is the Artist**\'**s name? ').strip()

intID = int(strID)

**return** intID, strTitle, strArtist

**Figure 2:**

*Results after selecting a to add cd and entering a non integer*

*Graphical user interface, text

Description automatically generated*

1. Next, I added a structured error handling to def write\_file by using try/except statement. The exception that I used was the FileNotFoundError. FileNotFoundError print statement appears when attempting to open a file that doesn’t exist. I also included the unpickling code. See below for the following code:

@staticmethod

**def** write\_file(file\_name, table):

**try**:

**with** open(file\_name, 'w') **as** objFile:

**for** row **in** table:

lstValues = list(row.values())

lstValues[0] = str(lstValues[0])

objFile.write(','.join(lstValues) + '**\n**')

*# unpickling*

*# read from a binary file*

**with** open(strFileName,'wb') **as** file:

pickle.dump(table, file)

objFile.close()

**except** **FileNotFoundError**: *# opening file does not exist*

print('Error: No such file or directory')

**return** table

1. Next, I added a structured error handling to def read\_file by using try/except statement. The exception that I used was the FileNotFoundError. The error appears when it can’t find the file referencing in the code. I also included the pickling code. See below for the following code:

@staticmethod

**def** read\_file(file\_name, table):

*# adding structured error handling for read file*

**try**:

**with** open(file\_name, 'r') **as** objFile:

*#objFile = open(file\_name, 'r')*

*# read file name*

table.clear() *# this clears existing data and allows to load data from file*

**for** line **in** objFile:

data = line.strip().split(',')

dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}

table.append(dicRow)

**with** open(file\_name, 'rb') **as** file:

table = pickle.load(file)

objFile.close()

**break**

**except** **FileNotFoundError**: *# can't find file referencing*

print('The file is not found, check file location')

**return** table

**Figure 3:**

*Results after deleting CDInventory.dat file*

*﻿Graphical user interface, text

Description automatically generated*

1. Next, I worked on structured error handling for deleting a CD from a table. I used try/except with ValueError code and included a customize print statement on the error. When the user inputs an ID that is not in the table, exception triggers. The exception part of the code is not working. See below for the following code:

**def** deleting\_cd(table):

intRowNr = -1

blnCDRemoved = **False**

**try**:

**for** row **in** lstTbl:

intRowNr += 1

**if** row['ID'] == intIDDel:

**del** table[intRowNr]

blnCDRemoved = **True**

**except** **ValueError**:

print('Could not find this CD!')

**else**:

**if** blnCDRemoved:

print('The CD was removed')

**return** table

1. The last modification was on the menu selection. I raise an exception and added try/except statement. When a user entered an incorrect menu selection, exception ValueError triggers and print out, “enter a valid menu selection”. See below for the code:

choice = ' '

**try**:

*# adding structered error handling*

choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()

**while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:

**raise** **ValueError**

**except** **ValueError** **as** e:

print(type(e))

print('Enter a valid menu selection')

print() *# Add extra space for layout*

**return** choice

**Figure 4:**

*Results after entering invalid menu selection*

﻿ Graphical user interface, text

Description automatically generated

1. After successfully running the program in Terminal, see below for screenshots:

**Figure 5:**

*Results after running python script in Terminal*

*Text

Description automatically generated*

**Figure 5.1:**

*Results after entering invalid menu selection in Terminal*

Text

Description automatically generated

**Figure 5.2:**

*Results after selecting a to add cd and entering a non integer in Terminal*

*Text

Description automatically generated*

**Figure 5.3:**

*Results after deleting CDInventory.dat file*

*Text

Description automatically generated*

1. After completing this assignment, the files are posted in GitHub. Here is the link to the repository:

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

# Summary

I had a setback in this assignment. I’m unable to make the exception work when user input an incorrect ID to delete a CD inventory.

In completing this assignment, I gained experience modifying existing code to add try/except statements with different type of errors. Also, I learned how to to use pickling and how vulnerable it is to unpickle someone else’s data. Overall, I thought this assignment was challenging. I had a difficult time on where to place the except statement in order for it to work.

# Appendix