Matthew Poehler

February 28, 2021

Foundations of Programming: Python

Assignment 07

CD Inventory Script

Introduction

In this document I will demonstrate the steps taken to modify the existing CD Inventory script that was originally provided by Professor Biesinger in Module 06 of the Foundations of Programming: Python course. This will include brief descriptions of the new aspects implemented in the assignment such as Pickling and error handling. These topics were introduced in the Module 07 with more information researched using the internet.

Pickle

Pickling according to the Module 07 material is the process of storing information as binary data. The webpage, https://docs.python.org (external site), is another great source on the pickle function. Inside the pickle module the dump.() function serializes the data into a file and the load.() function de-serializes the data back into the program[1]. Listing 1 and 2 demonstrate this using portions of the CD Inventory script.

```
95          with open(file_name, 'wb') as objFile:
96              pickle.dump(table, objFile)
97          objFile.close()
98
```

Listing 1 - Use of pickling module in CDInventory.py, saving data to a file.

```
81          with open(file_name, 'rb') as objFile:
82              table = pickle.load(objFile)
83          return table
```

Listing 2 - Using of pickling module in CDInventory.py, reading data back into program

- One note to make is that for the pickle module to execute an “import pickle” statement needs to be passed through before the pickle.load() and pickle.dump() functions occur.

Error Handling

When creating programs for humans to use it is wise to understand as many faults in the script as possible and then try to manage them without the program crashing. A way to do this is Error Handling. According to the material in Module 07 and w3schools.com (external site) trapping the error in a

[1] https://docs.python.org/3/library/pickle.html#module-pickle, retrieved 2021-Feb-28

try/except loop is the way to acknowledge the error occurs and then notify the user in a more situationally specific way without stopping the program. Listing 3 and 4 are examples of what this looks like in parts of the CD Inventory script.

```
172    try:
173        lstTbl = FileProcessor.read_file(strFileName, lstTbl)
174    except FileNotFoundError:
175        print('\nFile:', strFileName, ', Does not exist. Please save Inventory data to create file.')
176    except EOFError:
177        print('\nFile is empty. Please save CD Inventory data.')
178
```

Listing 3 - Structured error handling in CDInventory.py, trapping file related errors when program first runs for a user

```
207            try:
208                # 3.3.1 Ask user for new ID, CD Title and Artist
209                datLst = IO.get_data()
210                # 3.3.2 Add item to the table
211                DataProcessor.add_Inv(datLst)
212            except ValueError:
213                print('\nEntry Error!')
214                print('If you would like to add an entry please enter an integer(number) for an ID.')
215            finally:
216                IO.show_inventory(lstTbl)
217                continue  # start loop back at top.
```

Listing 4 - Structured error handling in CDInventory.py, minimizing human error by letting user know if a number was not input

To determine the errors to handle I researched a list of error types on a great website[2], https://www.tutorialsteacher.com (external site)

CD Inventory Script

The purpose of this CD Inventory script is the same as previous assignments, get musical CD information from the user to save to an inventory that can be manipulated or saved to a file. The modifications made in this assignment were changing the file from a text file to a binary file and adding a few error handlings to minimize human error. Figures show the new parts working or managing some errors that may arise.

[2] https://www.tutorialsteacher.com/python/error-types-in-python, retrieved 2021-Feb-28

Figure 1 - CDInventory.py being executed in Spyder IDE, showing the error handling messages when the file does not exist and if letters are entered where numbers should be

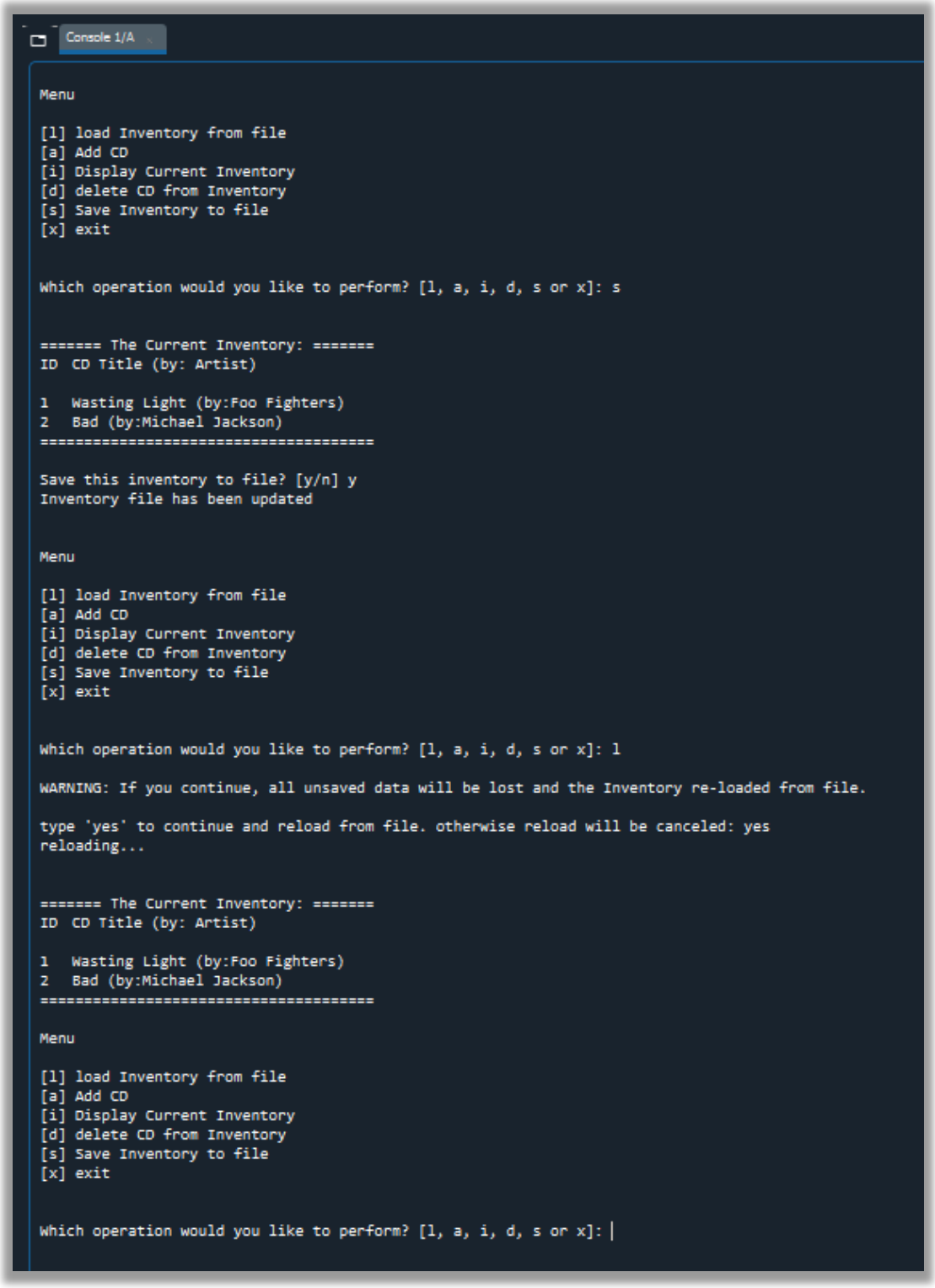

Figure 2 - CDInventory.py continuing in Spyder IDE, demonstrating the saving and re-loading of data

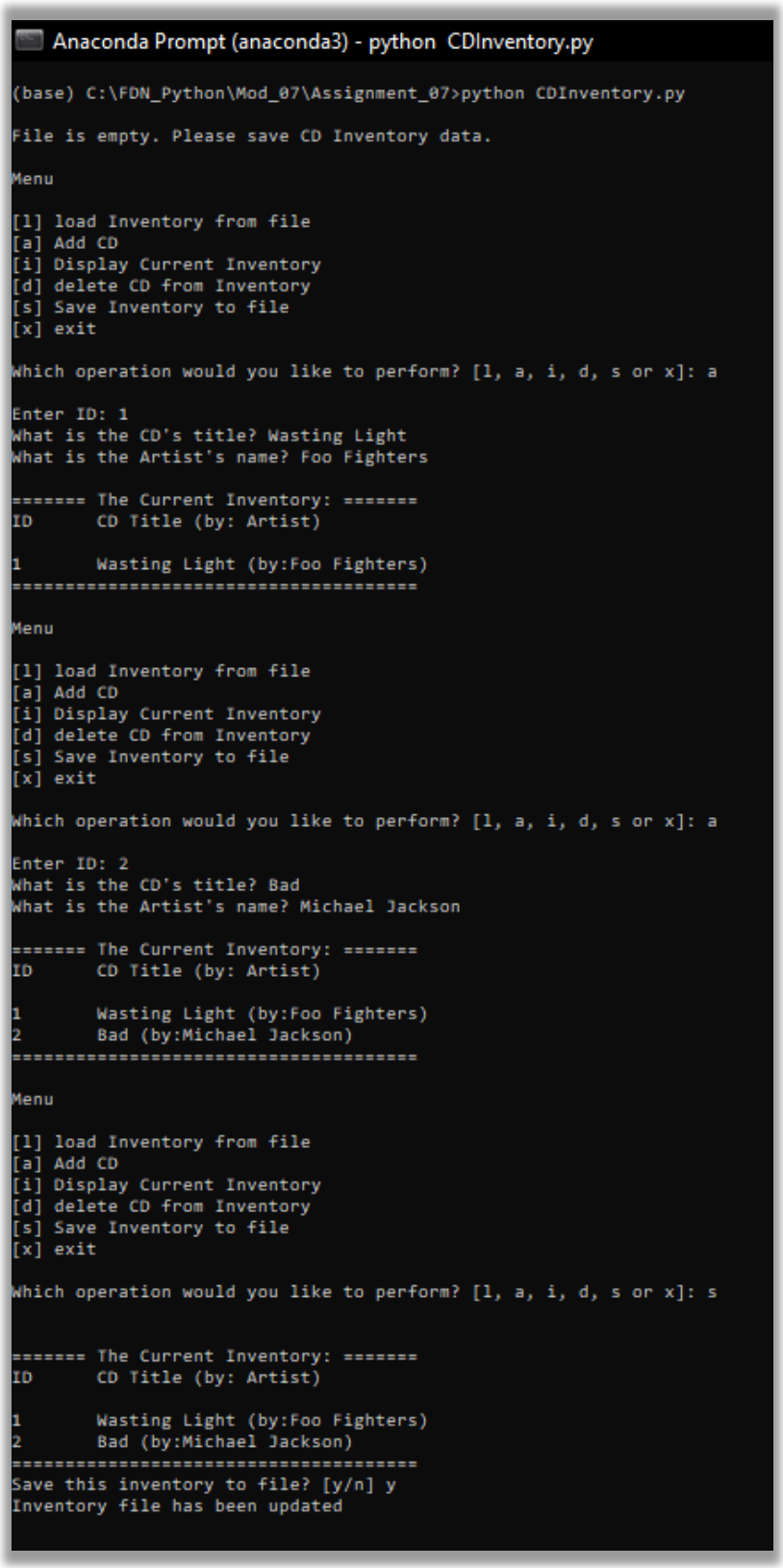

Figure 3 - CDInventory.py being executed in Anaconda Prompt, demonstrating the empty file error handling and saving of data

Summary

With the combination of course material and information from the internet the CD Inventory script has been updated to read and save binary data as well as make sure the user inputs the proper data in when prompted to. If I had more time for this assignment, I would have liked to add more structured error handling, but I am excited to continue to build on my python and programming knowledge.