Artem Aleshin
August 21st, 2022
IT FDN 110 A
Assignment 07

# Assignment 07 Knowledge Document

## Introduction

The Knowledge Document covers Points of Interest from Module 06, pages that do and do not help with Exception Handling and Pickling, the work done as part of the Seventh set of Laboratories and the completed Assignment. As part of working through the tasks, I learned about working with Binary Files, Structured Error Handling and more about working with Text Files.

## Module 06 Points of Interest

A topic that I found interesting from Module 07 is pickling. It was interesting to learn from the book that Python does not pickle to binary files by default, but that can be adjusted by changing bin to true. Furthermore, it is convenient that Python includes the pickling module in it by default. I found the concept of shelving somewhat confusing and could use more clarification in class. In addition, I would be interested to learn more about the with keyword. Lastly, the structured error handling concept seems to be fairly straightforward.

## Exception Handling

A page that I found to be helpful with understanding Exception Handling is https://www.programiz.com/python-programming/exception-handling. The page gives clear the examples and the accompanying video on the page does a good job of explaining Exception Handling further. Another page that I found somewhat helpful is https://www.geeksforgeeks.org/python-exception-handling/ because it provides some good examples. A page that I did not find as helpful is https://docs.python.org/3/tutorial/errors.html because there is an overwhelming amount of content on it.

## Pickling

One page I found helpful with pickling is https://www.tutorialspoint.com/python-pickling because it is fairly straightforward in its explanations. A video that I found helpful is https://www.youtube.com/watch?v=2Tw39kZIbhs because it gives a simple example with a dictionary and provides a clear explanation of what pickling does. Similarly, a page that I did not find as helpful is https://docs.python.org/3/library/pickle.html because there is a lot of technical jargon on it that I am not familiar with.

# Module 07 Laboratories

## LAB 07_A

1. The script was duplicated.
2. The code was modified and is as follows:

```python
#------------------------------------------#
# Title: LAB07_A.py
# Desc: Demonstrates reading and writing to a file
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# aaleshin, 2022-Aug-21, Modified File to Read and Write Text Files
#------------------------------------------#

# -- DATA -- #
strFileInput = 'mathIn.txt'
strFileOutput = 'mathOut.txt'

# -- PROCESSING -- #
class SimpleMath:
    """A collection of simple math processing functions """

    @staticmethod
    def get_sum(val1 = 0.0, val2 = 0.0):
        """Function for adding two values


        Args:
            val1: the first number to add
            val2: the second number to add


        Returns:
            A float corresponding to the sum of val1 and val2
        """
        return float(val1 + val2)

    @staticmethod
    def get_diffference(val1 = 0.0, val2 = 0.0):
        """Function for subtracting two values


        Args:
            val1: the number to subtract from
            val2: the number to subtract


        Returns:
            A float corresponding to the difference of val1 and val2
        """
        return float(val1 - val2)

    @staticmethod
    def get_product(val1 = 0.0, val2 = 0.0):
        """Function for multiplying two values


        Args:
            val1: the first number to multiply
            val2: the second number to multiply

```

*Figure 1: Lab 07_A Script Part 1.*

```
        Returns:
            A float corresponding to the product of val1 and val2
        """
        return float(val1 * val2)

    @staticmethod
    def get_quotient(val1 = 0.0, val2 = 0.0):
        """Function for dividing two values


        Args:
            val1: the number to divide
            val2: the number to divide by


        Returns:
            A float corresponding to the quotient of val1 and val2
        """
        return float(val1 / val2)


class IO:
    """A collection of the Input / Output operations """

    def read_file(fileName):
        """
        function to read in two numbers from file fileName and return these

        Args:
            fileName (string): file name to read the numbers from

        Returns:
            numA (int): first number in file fileName.
            numB (int): second number in file fileName.

        """

        with open(fileName, 'r') as fileObj:
            data = fileObj.readline()
            lstInput = data.split(',')
            numA = float(lstInput[0])
            numB = float(lstInput[1])
        return numA, numB



    def write_file(fileName, results):
        """
        function to write the math results to file fileName

        Args:
            fileName (string): file Name to write the results to.
            results (list): The results

```

*Figure 2: Lab 07_A Script Part 2.*

```
        Returns:
            None.

        """

        with open(strFileOutput, 'w') as fileObj:
            strResults = str(results)
            strResults = strResults[1:-1]
            fileObj.write(str(strResults))

# -- PRESENTATION (Input/Output) -- #
print('Basic Math script. Calculating the Sum, Difference, Product and Quotient of two numbers.')
IO.read_file(strFileInput)
intNumA, intNumB = IO.read_file(strFileInput)
lstResults = []
lstResults.append(SimpleMath.get_sum(intNumA, intNumB))
lstResults.append(SimpleMath.get_diffference(intNumA, intNumB))
lstResults.append(SimpleMath.get_product(intNumA, intNumB))
lstResults.append(SimpleMath.get_quotient(intNumA, intNumB))
IO.write_file(strFileOutput, lstResults)
```

*Figure 3: Lab 07_A Script Part 3.*

3. The script was tested.
    a. The files were read and written correctly and no errors were produced.
    b. The script was modified to read and write files by both modifying the functions as well as by modifying the script at the bottom of the page to remove and replace the user inputs. A loop would have to be used for the files where the data gets appended similar to the CDInventory program.

## LAB 07-B

1. A copy of LAB07-A was made.
2. The script was modified and is as follows:

```python
#-----------------------------------------#
# Title: LAB07_B.py
# Desc: Demonstrates reading and writing to a file
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# aaleshin, 2022-Aug-21, Modified File to Read and Write Text Files
# aaleshin, 2022-Aug-21, Modified File to use Arguments and Binary Files

#-----------------------------------------#

import sys, pickle

# -- DATA -- #
strFileInput = 'numbers.dat'
strFileOutput = 'results.dat'
argument = sys.argv[1]

file = open(strFileInput, 'ab')
file.close()

file = open(strFileOutput, 'ab')
file.close()

defaultInp = [1,2]

with open(strFileInput, 'ab') as fileObj:
    pickle.dump(defaultInp, fileObj)

# -- PROCESSING -- #
class SimpleMath:
    """A collection of simple math processing functions """

    @staticmethod
    def get_sum(val1 = 0.0, val2 = 0.0):
        """Function for adding two values


        Args:
            val1: the first number to add
            val2: the second number to add


        Returns:
            A float corresponding to the sum of val1 and val2
        """
        return float(val1 + val2)

    @staticmethod
    def get_diffference(val1 = 0.0, val2 = 0.0):
        """Function for subtracting two values


        Args:
            val1: the number to subtract from
            val2: the number to subtract


        Returns:
            A float corresponding to the difference of val1 and val2
        """
        return float(val1 - val2)

    @staticmethod
    def get_product(val1 = 0.0, val2 = 0.0):
        """Function for multiplying two values
```

*Figure 4: Lab 07_B Script Part 1.*

```
66
67
68          Args:
69              val1: the first number to multiply
70              val2: the second number to multiply
71
72
73          Returns:
74              A float corresponding to the product of val1 and val2
75          """
76          return float(val1 * val2)
77
78      @staticmethod
79      def get_quotient(val1 = 0.0, val2 = 0.0):
80          """Function for dividing two values
81
82
83          Args:
84              val1: the number to divide
85              val2: the number to divide by
86
87
88          Returns:
89              A float corresponding to the quotient of val1 and val2
90          """
91          return float(val1 / val2)
92
93
94  class IO:
95      """A collection of the Input / Output operations """
96
97      def read_file(fileName):
98          """
99          function to read in two numbers from file fileName and return these
100
101         Args:
102             fileName (string): file name to read the numbers from
103
104         Returns:
105             numA (int): first number in file fileName.
106             numB (int): second number in file fileName.
107
108         """
109
110         with open(fileName, 'rb') as fileObj:
111             data = pickle.load(fileObj)
112             lstInput = data.split(',')
113             numA = float(lstInput[0].strip('\''))
114             numB = float(lstInput[1].strip('\''))
115             return numA, numB, lstInput
116
117
118
119     def write_file(fileName, results):
120         """
121         function to write the math results to file fileName
122
123         Args:
124             fileName (string): file Name to write the results to.
125             results (list): The results
126
127         Returns:
128             None.
```

*Figure 5: Lab 07_B Script Part 2.*

```
            None.

        """

        with open(fileName, 'wb') as fileObj:
            strResults = str(results)
            strResults = strResults[1:-1]
            pickle.dump(strResults,fileObj)

# -- PRESENTATION (Input/Output) -- #

if argument == 'IO':
    IO.read_file(strFileOutput)
    intNumA, intNumB, lstInput = IO.read_file(strFileInput)
    print(lstInput)
    num1 = float(input('Enter 1st Number: '))
    num2 = float(input('Enter 2nd Number: '))
    inputs = [num1,num2]
    IO.write_file(strFileInput, inputs)

elif argument == 'calc':
    IO.read_file(strFileInput)
    intNumA, intNumB, lstInput = IO.read_file(strFileInput)
    lstResults = []
    lstResults.append(SimpleMath.get_sum(intNumA, intNumB))
    lstResults.append(SimpleMath.get_diffference(intNumA, intNumB))
    lstResults.append(SimpleMath.get_product(intNumA, intNumB))
    lstResults.append(SimpleMath.get_quotient(intNumA, intNumB))
    print(lstResults)
    IO.write_file(strFileOutput, lstResults)
```

*Figure 6: Lab 07_B Script Part 3.*

3. The script was tested,
    a. The output is as follows:

```
(base) C:\_FDProgramming\Lab_07>python Lab07_B.py IO
['564.0', ' 89.0']
Enter 1st Number: 65
Enter 2nd Number: 156

(base) C:\_FDProgramming\Lab_07>python Lab07_B.py calc
[221.0, -91.0, 10140.0, 0.4166666666666667]
```

*Figure 7: Lab 07_B Output.*

    b. The script works by creating a binary data file with some values if one is not present. The script is then updated to use binary files by implementing pickling. Furthermore, the first argument is called and the script splits based on which option is selected. This would be useful for if the user does not need to perform all of the operations in either the first or second option and would therefore save time and RAM.

## LAB 07-C

1. A copy of LAB07-B was made.
2. The script was modified per the instructions and is as follows:

```python
#-----------------------------------------#
# Title: LAB07_B.py
# Desc: Demonstrates reading and writing to a file
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# aaleshin, 2022-Aug-21, Modified File to Read and Write Text Files
# aaleshin, 2022-Aug-21, Modified File to use Arguments and Binary Files

#-----------------------------------------#

import sys, pickle

# -- DATA -- #
strFileInput = 'numbers.dat'
strFileOutput = 'results.dat'
argument = sys.argv[1]

file = open(strFileInput, 'ab')
file.close()

file = open(strFileOutput, 'ab')
file.close()

defaultInp = [1,2]

with open(strFileInput, 'ab') as fileObj:
    pickle.dump(defaultInp, fileObj)

# -- PROCESSING -- #
class SimpleMath:
    """A collection of simple math processing functions """

    @staticmethod
    def get_sum(val1 = 0.0, val2 = 0.0):
        """Function for adding two values


        Args:
            val1: the first number to add
            val2: the second number to add


        Returns:
            A float corresponding to the sum of val1 and val2
        """
        return float(val1 + val2)

    @staticmethod
    def get_diffference(val1 = 0.0, val2 = 0.0):
        """Function for subtracting two values


        Args:
            val1: the number to subtract from
            val2: the number to subtract


        Returns:
            A float corresponding to the difference of val1 and val2
        """
        return float(val1 - val2)

    @staticmethod
    def get_product(val1 = 0.0, val2 = 0.0):
        """Function for multiplying two values
```

*Figure 8: Lab 07_C Script Part 1.*

```
65          """Function for multiplying two values
66
67
68          Args:
69              val1: the first number to multiply
70              val2: the second number to multiply
71
72
73          Returns:
74              A float corresponding to the product of val1 and val2
75          """
76          return float(val1 * val2)
77
78      @staticmethod
79      def get_quotient(val1 = 0.0, val2 = 0.0):
80          """Function for dividing two values
81
82
83          Args:
84              val1: the number to divide
85              val2: the number to divide by
86
87
88          Returns:
89              A float corresponding to the quotient of val1 and val2
90          """
91          return float(val1 / val2)
92
93
94  class IO:
95      """A collection of the Input / Output operations """
96
97      def read_file(fileName):
98          """
99          function to read in two numbers from file fileName and return these
100
101         Args:
102             fileName (string): file name to read the numbers from
103
104         Returns:
105             numA (int): first number in file fileName.
106             numB (int): second number in file fileName.
107
108         """
109
110         with open(fileName, 'rb') as fileObj:
111             data = pickle.load(fileObj)
112             lstInput = data.split(',')
113             numA = float(lstInput[0].strip('\''))
114             numB = float(lstInput[1].strip('\''))
115             return numA, numB, lstInput
116
117     def write_file(fileName, results):
118         """
119         function to write the math results to file fileName
120
121         Args:
122             fileName (string): file Name to write the results to.
123             results (list): The results
124
125         Returns:
126             None.
127
128         """
```

*Figure 9: Lab 07_C Script Part 2.*

```
        """

        with open(fileName, 'wb') as fileObj:
            strResults = str(results)
            strResults = strResults[1:-1]
            pickle.dump(strResults,fileObj)

# -- PRESENTATION (Input/Output) -- #

if argument == 'IO':
    try:
        IO.read_file(strFileOutput)
    except FileNotFoundError as e:
        print('Binary file does not exist!')
        print('Build in error info:')
        print(type(e), e, e.__doc__, sep = '\n')
    except Exception as e:
        print('There was a general error!')
        print('Build in error info:')
        print(type(e), e, e.__doc__, sep = '\n')

    intNumA, intNumB, lstInput = IO.read_file(strFileInput)
    print(lstInput)
    try:
        num1 = float(input('Enter 1st Number: '))
    except ValueError as e:
        print('That is not an integer!')
        print('Build in error info:')
        print(type(e), e, e.__doc__, sep = '\n')
    except Exception as e:
        print('There was a general error!')
        print('Build in error info:')
        print(type(e), e, e.__doc__, sep = '\n')

    try:
        num2 = float(input('Enter 2nd Number: '))
        if num2 == 0:
            raise ZeroDivisionError
    except ValueError as e:
        print('That is not an integer!')
        print('Build in error info:')
        print(type(e), e, e.__doc__, sep = '\n')
    except Exception as e:
        print('There was a general error!')
        print('Build in error info:')
        print(type(e), e, e.__doc__, sep = '\n')

    inputs = [num1,num2]

    try:
        IO.write_file(strFileInput, inputs)
    except Exception as e:
        print('There was a general error!')
        print('Build in error info:')
        print(type(e), e, e.__doc__, sep = '\n')

elif argument == 'calc':
    try:
        IO.read_file(strFileInput)
    except FileNotFoundError as e:
        print('Binary file does not exist!')
        print('Build in error info:')
        print(type(e), e, e.__doc__, sep = '\n')
    except Exception as e:
```

*Figure 10: Lab 07_C Script Part 3.*

```
191         except Exception as e:
192             print('There was a general error!')
193             print('Build in error info:')
194             print(type(e), e, e.__doc__, sep = '\n')
195 
196         intNumA, intNumB, lstInput = IO.read_file(strFileInput)
197         lstResults = []
198         lstResults.append(SimpleMath.get_sum(intNumA, intNumB))
199         lstResults.append(SimpleMath.get_diffference(intNumA, intNumB))
200         lstResults.append(SimpleMath.get_product(intNumA, intNumB))
201         lstResults.append(SimpleMath.get_quotient(intNumA, intNumB))
202         print(lstResults)
203         try:
204             IO.write_file(strFileOutput, lstResults)
205         except Exception as e:
206             print('There was a general error!')
207             print('Build in error info:')
208             print(type(e), e, e.__doc__, sep = '\n')
```

*Figure 11: Lab 07_C Script Part 4.*

3. The script was tested,
    a. The output is as follows:

```
(base) C:\_FDProgramming\Lab_07>python Lab07_C.py calc
[12.0, -2.0, 35.0, 0.7142857142857143]

(base) C:\_FDProgramming\Lab_07>python Lab07_C.py IO
['5.0', ' 7.0']
Enter 1st Number: a
That is not an integer!
Build in error info:
<class 'ValueError'>
could not convert string to float: 'a'
Inappropriate argument value (of correct type).
Enter 2nd Number: b
That is not an integer!
Build in error info:
<class 'ValueError'>
could not convert string to float: 'b'
Inappropriate argument value (of correct type).
Traceback (most recent call last):
  File "C:\_FDProgramming\Lab_07\Lab07_C.py", line 175, in <module>
    inputs = [num1,num2]
NameError: name 'num1' is not defined

(base) C:\_FDProgramming\Lab_07>python Lab07_C.py IO
['5.0', ' 7.0']
Enter 1st Number: 23
Enter 2nd Number: 68

(base) C:\_FDProgramming\Lab_07>python Lab07_C.py calc
[91.0, -45.0, 1564.0, 0.3382352941176471]

(base) C:\_FDProgramming\Lab_07>python Lab07_C.py IO
['23.0', ' 68.0']
Enter 1st Number: 2
Enter 2nd Number: 5

(base) C:\_FDProgramming\Lab_07>
```

*Figure 12: Lab 07_C Output.*

    b. The script is essentially the same as that of Lab07-B with the exception that error handling was added for non-numeric inputs, division by zero and reading and writing the files. Each of the structured error handling instances was accomplished by using the try and except keywords.

# Assignment 07

1. Folder called Assignment07 was created.
2. The code was modified and is as follows:

```python
#------------------------------------------#
# Title: Assignment07.py
# Desc: Working with classes and functions.
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# aaleshin, 2022-Aug-13, Modified File to move code into functions under classes
# aaleshin, 2022-Aug-14, Modified File to include docstrings
# aaleshin, 2022-Aug-21, Modified File to use Binary Files and Structured Error Handling
#------------------------------------------#

import pickle

# -- DATA -- #
strChoice = '' # User input
lstTbl = []  # list of lists to hold data
dicRow = {}  # list of data row
strFileName = 'CDInventory.dat'  # data storage file
objFile = None  # file object

# -- PROCESSING -- #
class DataProcessor:
    # TODone add functions for processing here
    """Manipulating the data in memory"""

    @staticmethod
    def delete_CD(table):
        """Deletes CD from in memory table

        Args:
            table (list of dict): In memory 2D data structure (list of dicts) that holds the data during runtime

        Returns:
            None.
        """
        intRowNr = -1
        blnCDRemoved = False
        for row in table:
            intRowNr += 1
            if row['ID'] == intIDDel:
                del table[intRowNr]
                blnCDRemoved = True
                break
        if blnCDRemoved:
            print('The CD was removed')
        else:
            print('Could not find this CD!')

    @staticmethod
    def add_CD(table, ID, Title, Artist):
        """Adds CD information to in memory table

        Args:
            table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
            ID (String): String containing CD ID.
            Title (String): String containing CD Title.
            Artist (String): String containing CD Artist.

        Returns:
            None.
        """
        intID = int(ID)
        dicRow = {'ID': intID, 'Title': Title, 'Artist': Artist}
        table.append(dicRow)
        IO.show_inventory(table)

```

*Figure 13: Assignment 07 Script Part 1.*

```python
class FileProcessor:
    """Processing the data to and from text file"""

    @staticmethod
    def read_file(file_name, table):
        """Function to manage data ingestion from file to a list of dictionaries

        Reads the data from binary file identified by file_name into a 2D table
        (list of dicts) table one line in the file represents one dictionary row in table.

        Args:
            file_name (string): name of binary file used to read the data from
            table (list of dict): 2D data structure (list of dicts) that holds the data during runtime

        Returns:
            None.
        """
        table.clear()  # this clears existing data and allows to load data from file
        objFile = open(file_name, 'rb')
        lst = []
        while True:
            try:
                lst.append(pickle.load(objFile))
            except (EOFError):
                break
        #data = pickle.load(objFile)
        for line in lst:
            line = line.strip().split(',')
            dicRow = {'ID': int(line[0]), 'Title': line[1], 'Artist': line[2]}
            table.append(dicRow)
        objFile.close()

    @staticmethod
    def write_file(file_name, table):
        # TODone Add code here
        """Function to manage writing data from table to file

        Writes data from 2D table (list of dicts) to binary file one row at a time.

        Args:
            file_name (string): name of file used to read the data from
            table (list of dict): 2D data structure (list of dicts) that holds the data during runtime

            Returns:
                None.
        """
        objFile = open(file_name, 'wb')
        for row in table:
            lstValues = list(row.values())
            lstValues[0] = str(lstValues[0])
            pickle.dump(','.join(lstValues) + '\n', objFile)
        objFile.close()

```

*Figure 14: Assignment 07 Script Part 2.*

```python
# -- PRESENTATION (Input/Output) -- #

class IO:
    """Handling Input / Output"""

    @staticmethod
    def print_menu():
        """Displays a menu of choices to the user

        Args:
            None.

        Returns:
            None.
        """

        print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
        print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')

    @staticmethod
    def menu_choice():
        """Gets user input for menu selection

        Args:
            None.

        Returns:
            choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x

        """
        choice = ' '
        while choice not in ['l', 'a', 'i', 'd', 's', 'x']:
            choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
        print()  # Add extra space for layout
        return choice

    @staticmethod
    def show_inventory(table):
        """Displays current inventory table


        Args:
            table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.

        Returns:
            None.

        """
        print('======= The Current Inventory: =======')
        print('ID\tCD Title (by: Artist)\n')
        for row in table:
            print('{}\t{} (by:{})'.format(*row.values()))
        print('======================================')

    # TODone add I/O functions as needed


    @staticmethod
    def get_CD():
        """Gets user input for CD ID, Title and Artist.

        Args:
            None.

        Returns:
            Returns the CD ID as a string.
            Returns the CD Title as a string.
            Returns the CD Artist as as string.
        """
        ID = input('Enter ID: ').strip()
```

*Figure 15: Assignment 07 Script Part 3.*

```
            ID = input('Enter ID: ').strip()
            Title = input('What is the CD\'s title? ').strip()
            Artist = input('What is the Artist\'s name? ').strip()

            return ID, Title, Artist

    # 1. When program starts, read in the currently saved Inventory
    try:
        FileProcessor.read_file(strFileName, lstTbl)
    except FileNotFoundError as e:
        print('Binary file does not exist!')
        print('Build in error info:')
        print(type(e), e, e.__doc__, sep = '\n')
    except EOFError as e:
        print('The file is empty!')
        print('Build in error info:')
        print(type(e), e, e.__doc__, sep = '\n')


    # 2. start main loop
    while True:
        # 2.1 Display Menu to user and get choice
        IO.print_menu()
        strChoice = IO.menu_choice()

        # 3. Process menu selection
        # 3.1 process exit first
        if strChoice == 'x':
            break
        # 3.2 process load inventory
        if strChoice == 'l':
            print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
            strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled')
            if strYesNo.lower() == 'yes':
                print('reloading...')
                try:
                    FileProcessor.read_file(strFileName, lstTbl)
                    IO.show_inventory(lstTbl)
                except FileNotFoundError as e:
                    print('Binary file does not exist!')
                    print('Build in error info:')
                    print(type(e), e, e.__doc__, sep = '\n')
                except EOFError as e:
                    print('The file is empty!')
                    print('Build in error info:')
                    print(type(e), e, e.__doc__, sep = '\n')
            else:
                input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
                IO.show_inventory(lstTbl)
            continue  # start loop back at top.
        # 3.3 process add a CD
        elif strChoice == 'a':
            # 3.3.1 Ask user for new ID, CD Title and Artist
            strID, strTitle, strArtist = IO.get_CD()
            # 3.3.2 Add item to the table
            # ToDone move processing code into function
            try:
                DataProcessor.add_CD(lstTbl, strID, strTitle, strArtist)
            except ValueError as e:
                print('ID is not an integer!')
                print('Build in error info:')
                print(type(e), e, e.__doc__, sep = '\n')
            continue  # start loop back at top.
        # 3.4 process display current inventory
        elif strChoice == 'i':
            IO.show_inventory(lstTbl)
            continue  # start loop back at top.
```

*Figure 16: Assignment 07 Script Part 4.*

```
            continue  # start loop back at top.
        # 3.5 process delete a CD
        elif strChoice == 'd':
            # 3.5.1 get Userinput for which CD to delete
            # 3.5.1.1 display Inventory to user
            IO.show_inventory(lstTbl)
            # 3.5.1.2 ask user which ID to remove
            try:
                intIDDel = int(input('Which ID would you like to delete? ').strip())
                # 3.5.2 search thru table and delete CD
                # TODone move processing code into function
                DataProcessor.delete_CD(lstTbl)
                IO.show_inventory(lstTbl)
            except ValueError as e:
                print('ID is not an integer!')
                print('Build in error info:')
                print(type(e), e, e.__doc__, sep = '\n')
            continue  # start loop back at top.
        # 3.6 process save inventory to file
        elif strChoice == 's':
            # 3.6.1 Display current inventory and ask user for confirmation to save
            IO.show_inventory(lstTbl)
            strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
            # 3.6.2 Process choice
            if strYesNo == 'y':
                # 3.6.2.1 save data
                # TODone move processing code into function
                try:
                    FileProcessor.write_file(strFileName, lstTbl)
                except Exception as e:
                    print('There was a general error!')
                    print('Build in error info:')
                    print(type(e), e, e.__doc__, sep = '\n')
            else:
                input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
            continue  # start loop back at top.
        # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
        else:
            print('General Error')
```

*Figure 17: Assignment 07 Script Part 5.*

3. The script was tested in both Spyder and Anaconda Prompt.
    a. The output in Spyder:

```
In [1]: runfile('C:/_FDProgramming/Assignment07/CDInventory.py', wdir='C:/_FDProgramming/Assignment07')
Binary file does not exist!
Build in error info:
<class 'FileNotFoundError'>
[Errno 2] No such file or directory: 'CDInventory.dat'
File not found.
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]: l

WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.
type 'yes' to continue and reload from file. otherwise reload will be canceledyes
reloading...
Binary file does not exist!
Build in error info:
<class 'FileNotFoundError'>
[Errno 2] No such file or directory: 'CDInventory.dat'
File not found.
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]: a
```

*Figure 18: Assignment 07 Spyder Output Part 1.*

```
Enter ID: 1
What is the CD's title? Ten
What is the Artist's name? Pearl Jam
======= The Current Inventory: =======
ID  CD Title (by: Artist)

1   Ten (by:Pearl Jam)
=======================================
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]: i

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

1   Ten (by:Pearl Jam)
=======================================
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]: d

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

1   Ten (by:Pearl Jam)
=======================================
```

*Figure 19: Assignment 07 Spyder Output Part 2.*

```
====================================
Which ID would you like to delete? a
ID is not an integer!
Build in error info:
<class 'ValueError'>
invalid literal for int() with base 10: 'a'
Inappropriate argument value (of correct type).
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   Ten (by:Pearl Jam)
======================================
Save this inventory to file? [y/n] y
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]: l

WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.
type 'yes' to continue and reload from file. otherwise reload will be canceledyes
reloading...
======= The Current Inventory: =======
ID  CD Title (by: Artist)
```

*Figure 20: Assignment 07 Spyder Output Part 3.*

```
1   Ten (by:Pearl Jam)
======================================
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]: a

Enter ID: b
What is the CD's title? Nevermind
What is the Artist's name? Nirvana
ID is not an integer!
Build in error info:
<class 'ValueError'>
invalid literal for int() with base 10: 'b'
Inappropriate argument value (of correct type).
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]: a

Enter ID: 2
What is the CD's title? Nervermind
What is the Artist's name? Nirvana
======= The Current Inventory: =======
ID  CD Title (by: Artist)

1   Ten (by:Pearl Jam)
2   Nervermind (by:Nirvana)
```

*Figure 21: Assignment 07 Spyder Output Part 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   Ten (by:Pearl Jam)
2   Nervermind (by:Nirvana)
======================================
Save this inventory to file? [y/n] y
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]: l

WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.
type 'yes' to continue and reload from file. otherwise reload will be canceledyes
reloading...
======= The Current Inventory: =======
ID  CD Title (by: Artist)

1   Ten (by:Pearl Jam)
2   Nervermind (by:Nirvana)
======================================
Menu
```

*Figure 22: Assignment 07 Spyder Output Part 5.*

```
======================================
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]: x
```

*Figure 23:: Assignment 07 Spyder Output Part 6.*

b. The output in Anaconda Prompt:

```
(base) C:\_FDProgramming>cd Assignment07

(base) C:\_FDProgramming\Assignment07>python CDInventory.py
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]: l

WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.
type 'yes' to continue and reload from file. otherwise reload will be canceledyes
reloading...
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Ten (by:Pearl Jam)
2       Nervermind (by:Nirvana)
======================================
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]: a

Enter ID: 3
What is the CD's title? Dirt
What is the Artist's name? Alice in Chains
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Ten (by:Pearl Jam)
2       Nervermind (by:Nirvana)
3       Dirt (by:Alice in Chains)
======================================
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       Ten (by:Pearl Jam)
2       Nervermind (by:Nirvana)
3       Dirt (by:Alice in Chains)
======================================
Save this inventory to file? [y/n] y
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]: d
```

*Figure 24: Assignment 07 Anaconda Prompt Output Part 1.*

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

1       Ten (by:Pearl Jam)
2       Nervermind (by:Nirvana)
3       Dirt (by:Alice in Chains)
=======================================
Which ID would you like to delete? yujmt
ID is not an integer!
Build in error info:
<class 'ValueError'>
invalid literal for int() with base 10: 'yujmt'
Inappropriate argument value (of correct type).
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]: d

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

1       Ten (by:Pearl Jam)
2       Nervermind (by:Nirvana)
3       Dirt (by:Alice in Chains)
=======================================
Which ID would you like to delete? 2
The CD was removed
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Ten (by:Pearl Jam)
3       Dirt (by:Alice in Chains)
=======================================
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       Ten (by:Pearl Jam)
3       Dirt (by:Alice in Chains)
=======================================
Save this inventory to file? [y/n] y
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]: l

WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.
type 'yes' to continue and reload from file. otherwise reload will be canceledyes
reloading...
======= The Current Inventory: =======
ID      CD Title (by: Artist)
```

*Figure 25: Assignment 07 Anaconda Prompt Output Part 2.*

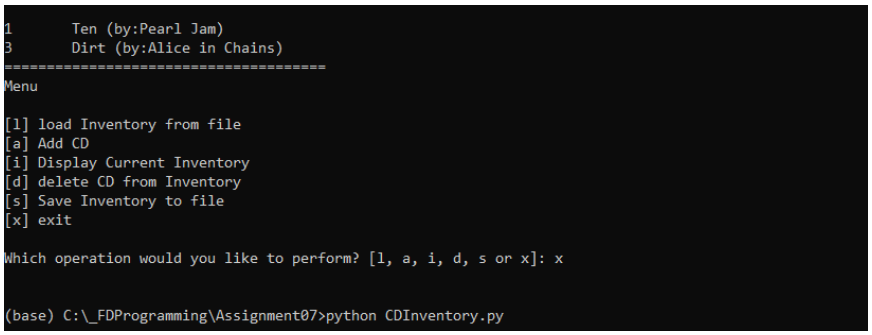


*Figure 26: Assignment 07 Anaconda Prompt Output Part 3.*

4. All options of the script were tested. The binary file after the operations shown above is:

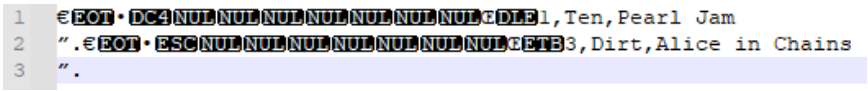


*Figure 27: Assignment 07 Binary File.*

In order to perform the assignment, I relied a lot on the lessons learned from the labs. I first modified the code to use binary files by using the pickling module. I then checked the original code for any signs of where an error could occur and then mitigated them by using the try: except: method. I made sure that only proper types of strings fit the ID when the user inputs it. I also modified the read and write methods. This particular assignment required a lot of testing, but all options appear to work fine.

This Knowledge Document and the script for the assignment have been uploaded to the following GitHub Repository:

https://github.com/aaleshinUW/Assignment_07

## Summary

The document covers the Points of Interest, the Laboratories, and the Assignment from Module 07. By completing Module 07, I learned about working with Binary Files, Text Files and Structured Error Handling.