Sonia Romo
2020-Aug-21
IT FDN 110 B - Foundations of Programming, Python
Assignment 07

Binary Files and Exception Handling

Introduction

This week, we learned about saving data to binary files and how to handle exceptions in code (so that the program doesn't exist and stop running for the user).

Labs

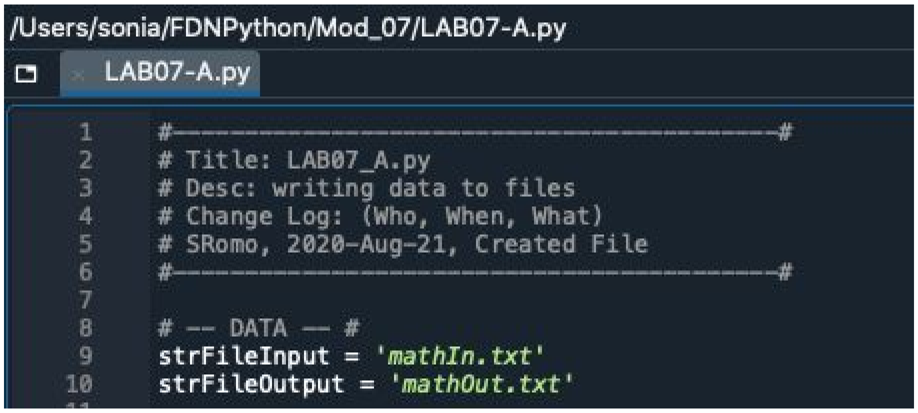
The labs this week were particularly challenging, so much so that I did the homework first (to save my sanity and my sleep) and then went back to the labs.

LAB07-A

In the `read_file` function, the code opens the text file, reads the numbers and assigns them to a variable `data`, assigns each number to a variable `numA` and `numB` and then returns the variables for use elsewhere in the program.

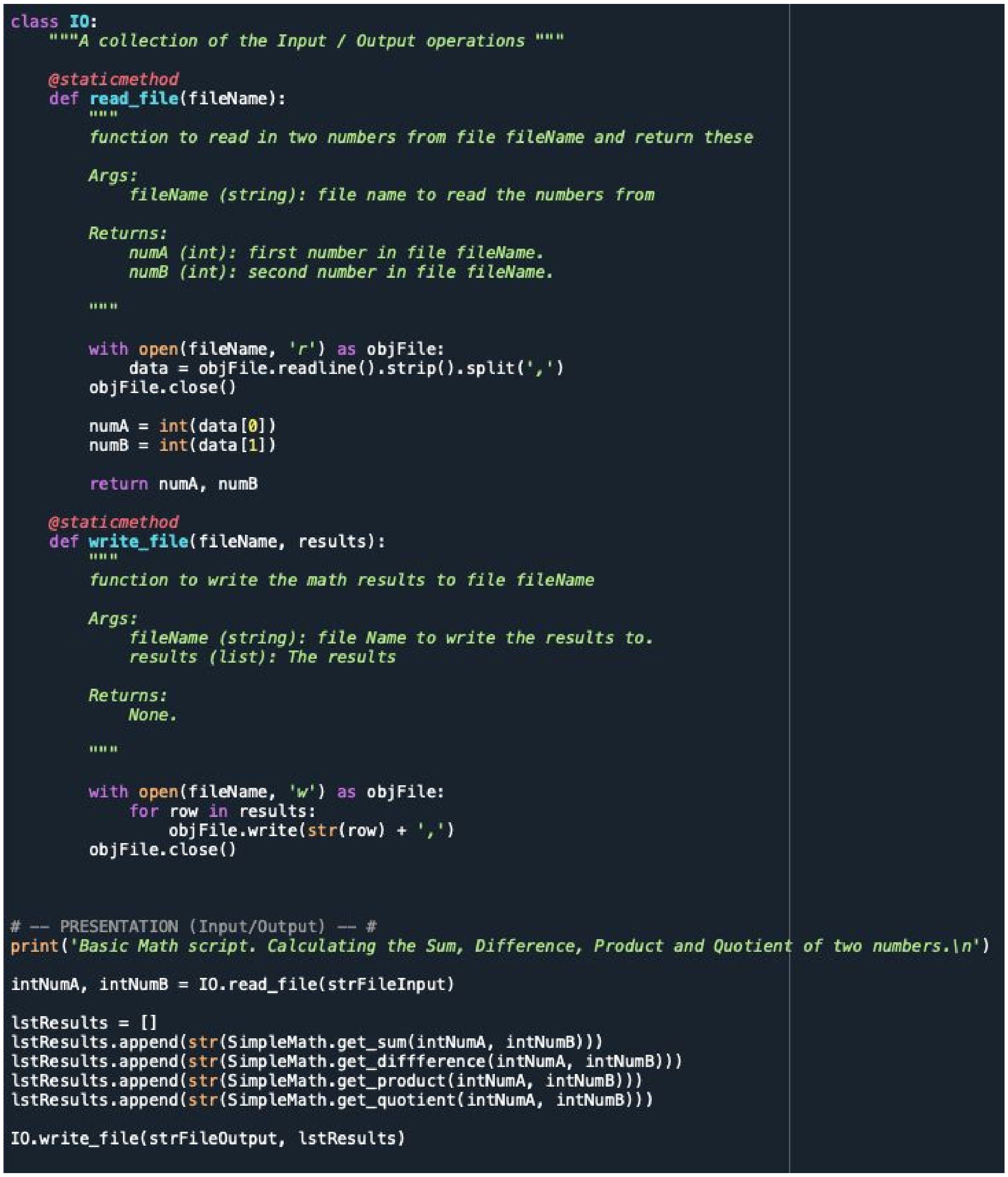
In the `write_file` function, the code runs through the rows in the table and writes each row to the file followed by a comma.

To read in multiple rows with two numbers each, I suppose you could use a loop to loop through the data.



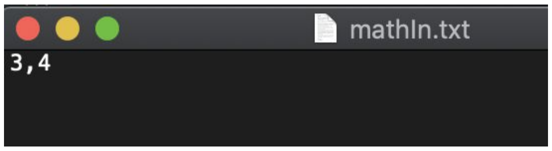
```
#-----------------------------------------------#
# Title: LAB07_A.py
# Desc: writing data to files
# Change Log: (Who, When, What)
# SRomo, 2020-Aug-21, Created File
#-----------------------------------------------#

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

Figure 01 - first part of LAB07-A code



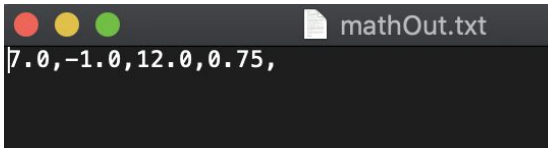
```python
class IO:
    """A collection of the Input / Output operations """

    @staticmethod
    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 objFile:
            data = objFile.readline().strip().split(',')
        objFile.close()

        numA = int(data[0])
        numB = int(data[1])

        return numA, numB

    @staticmethod
    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

        Returns:
            None.

        """

        with open(fileName, 'w') as objFile:
            for row in results:
                objFile.write(str(row) + ',')
        objFile.close()


# -- PRESENTATION (Input/Output) -- #
print('Basic Math script. Calculating the Sum, Difference, Product and Quotient of two numbers.\n')

intNumA, intNumB = IO.read_file(strFileInput)

lstResults = []
lstResults.append(str(SimpleMath.get_sum(intNumA, intNumB)))
lstResults.append(str(SimpleMath.get_diffference(intNumA, intNumB)))
lstResults.append(str(SimpleMath.get_product(intNumA, intNumB)))
lstResults.append(str(SimpleMath.get_quotient(intNumA, intNumB)))

IO.write_file(strFileOutput, lstResults)
```

Figure 02 - LAB07-A code I wrote



```
3,4
```

Figure 03 - txt input file



```
7.0,-1.0,12.0,0.75,
```

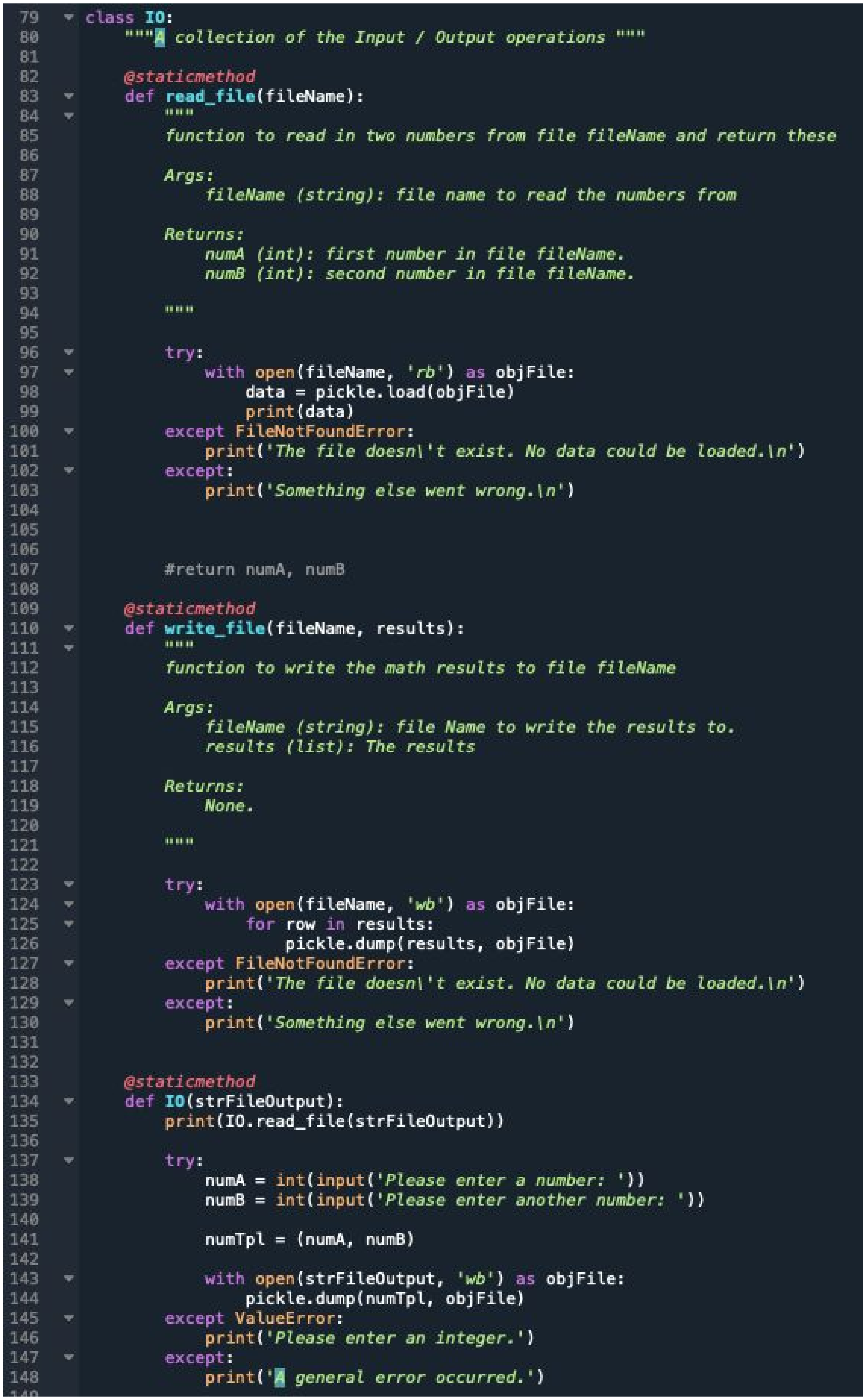
Figure 04 - txt output file

LAB07-B

The instructions on LAB07-B were quite confusing, I didn't understand them nor complete the lab. First, what is a path, aka IO path? Second, the instructions for the calc path say to "read in numbers from files" - which file or files? And "write out the results into file" - which file? How do you create `.dat` files from scratch, with data already in them? I don't remember that being explained either, so couldn't even set up the files to use.

LAB07-C

I added structured error handling to the file, although since I couldn't complete LAB07-B, the code doesn't run. However, here's the error handling I added:



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

    @staticmethod
    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.

        """

        try:
            with open(fileName, 'rb') as objFile:
                data = pickle.load(objFile)
                print(data)
        except FileNotFoundError:
            print('The file doesn\'t exist. No data could be loaded.\n')
        except:
            print('Something else went wrong.\n')



        #return numA, numB

    @staticmethod
    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

        Returns:
            None.

        """

        try:
            with open(fileName, 'wb') as objFile:
                for row in results:
                    pickle.dump(results, objFile)
        except FileNotFoundError:
            print('The file doesn\'t exist. No data could be loaded.\n')
        except:
            print('Something else went wrong.\n')


    @staticmethod
    def IO(strFileOutput):
        print(IO.read_file(strFileOutput))

        try:
            numA = int(input('Please enter a number: '))
            numB = int(input('Please enter another number: '))

            numTpl = (numA, numB)

            with open(strFileOutput, 'wb') as objFile:
                pickle.dump(numTpl, objFile)
        except ValueError:
            print('Please enter an integer.')
        except:
            print('A general error occurred.')
```

Figure 05 - structured error handling code

Homework

Research for exception handling and pickling

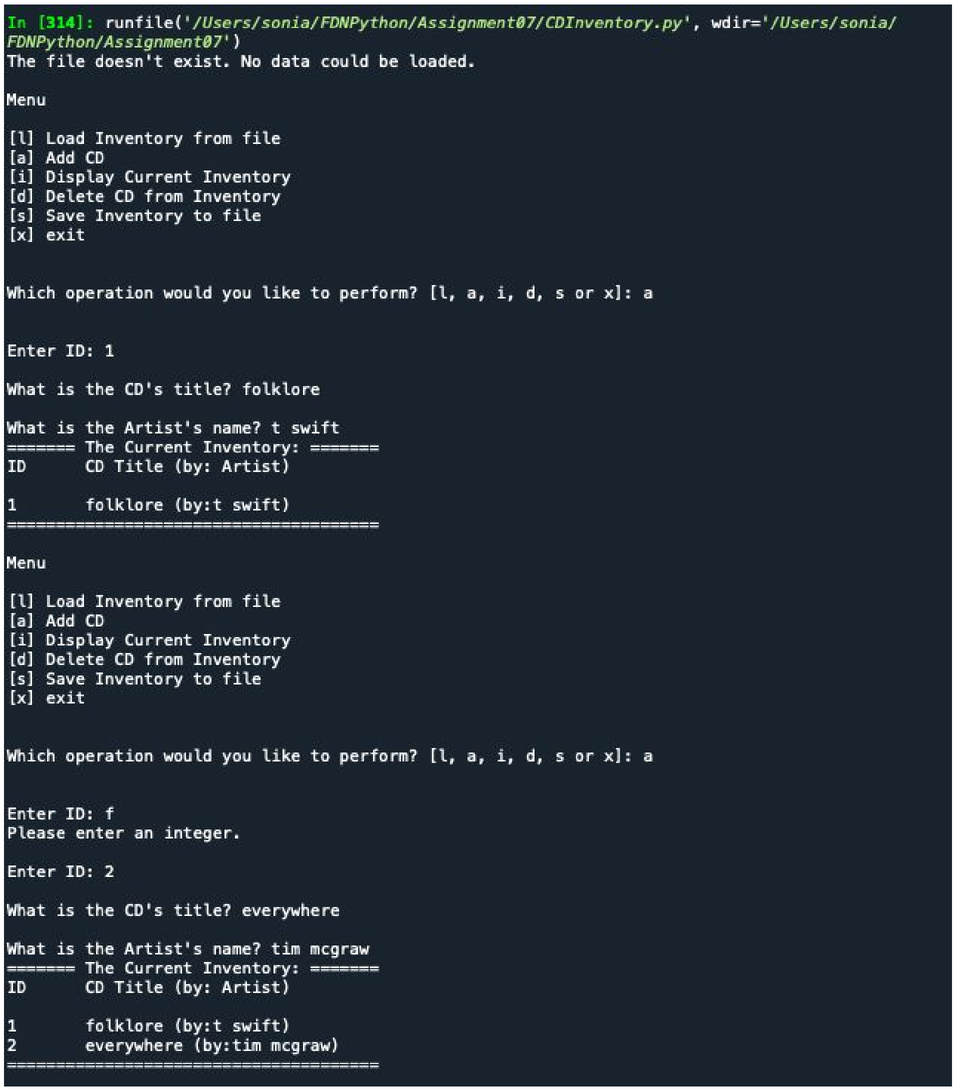
I found a few helpful articles for exception handling and pickling.

First, for exception handling:
Link 1 (retrieved 2020-Aug-25) - this article was helpful in seeing the structure of exception handling.
Link 2 (retrieved 2020-Aug-25) - this article was helpful in seeing the full list of possible built-in exceptions.

For pickling:
Link 3 (retrieved 2020-Aug-26) - this article was helpful in seeing examples of pickling.
Pages 200-202 in Python Programming for the Absolute Beginner (retrieved 2020-Aug-26) - the textbook was helpful in understanding the `.load` and `.dump` aspects.

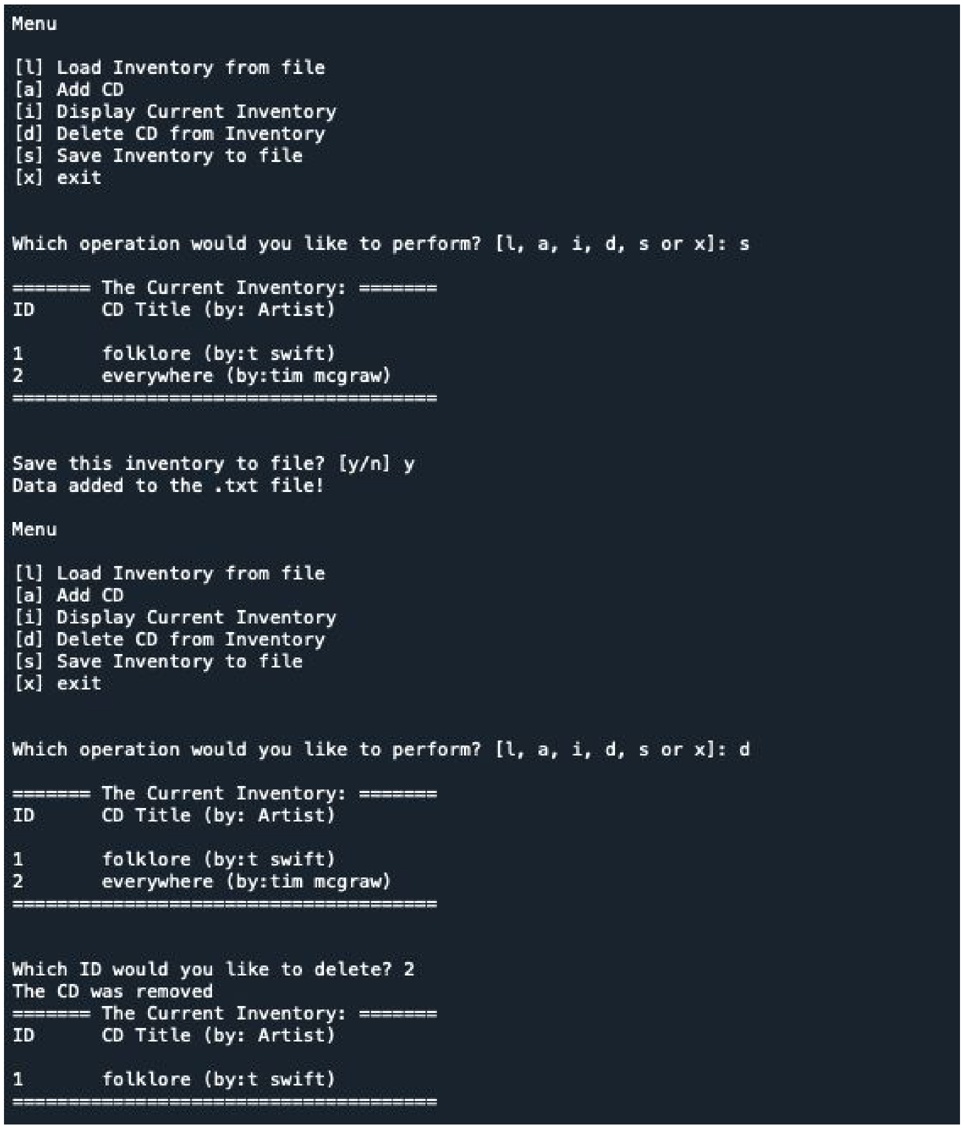
How my code works

This week, the try / except handling was fairly straight forward. Where I got caught up was on creating the `.dat` file. I was getting a file not found error (before and after implementing exception handling). This threw me off because I wasn't understanding why the file wasn't found and couldn't remember how to create the file in the code. What helped here was the tip on page 201 of Python Programming for the Absolute Beginner (retrieved 2020-Aug-26) that said that using `wb` would create the file, if it didn't exist. Once I implemented this correctly with `.load` and `.dump`, everything worked well.



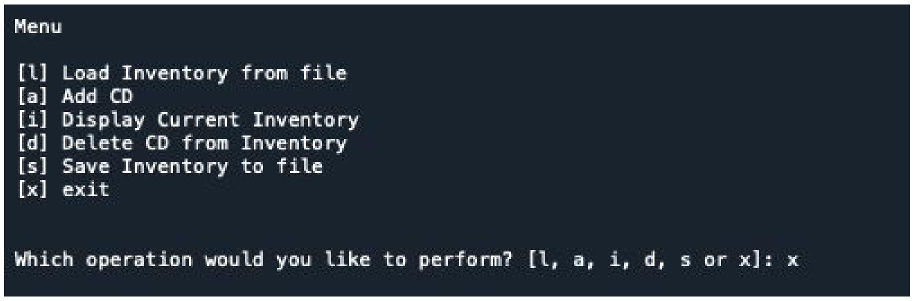
```
In [314]: runfile('/Users/sonia/FDNPython/Assignment07/CDInventory.py', wdir='/Users/sonia/
FDNPython/Assignment07')
The file doesn't exist. No data could be loaded.

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: 1

What is the CD's title? folklore

What is the Artist's name? t swift
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       folklore (by:t swift)
======================================

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: f
Please enter an integer.

Enter ID: 2

What is the CD's title? everywhere

What is the Artist's name? tim mcgraw
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       folklore (by:t swift)
2       everywhere (by:tim mcgraw)
======================================
```

Figure 06 - Spyder run screenshot #1



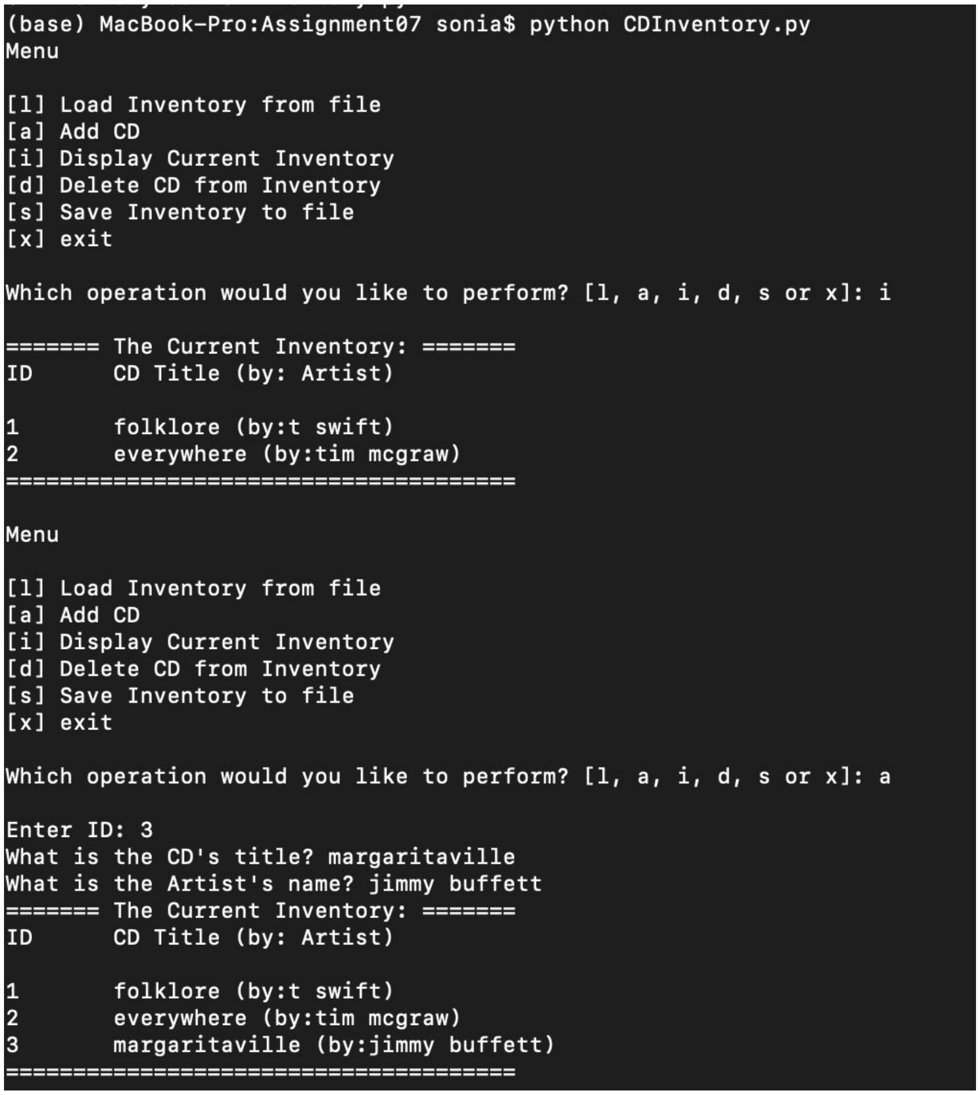
```
Menu

[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       folklore (by:t swift)
2       everywhere (by:tim mcgraw)
======================================


Save this inventory to file? [y/n] y
Data added to the .txt file!

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       folklore (by:t swift)
2       everywhere (by:tim mcgraw)
======================================


Which ID would you like to delete? 2
The CD was removed
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       folklore (by:t swift)
======================================
```

Figure 07 - Spyder run screenshot #2



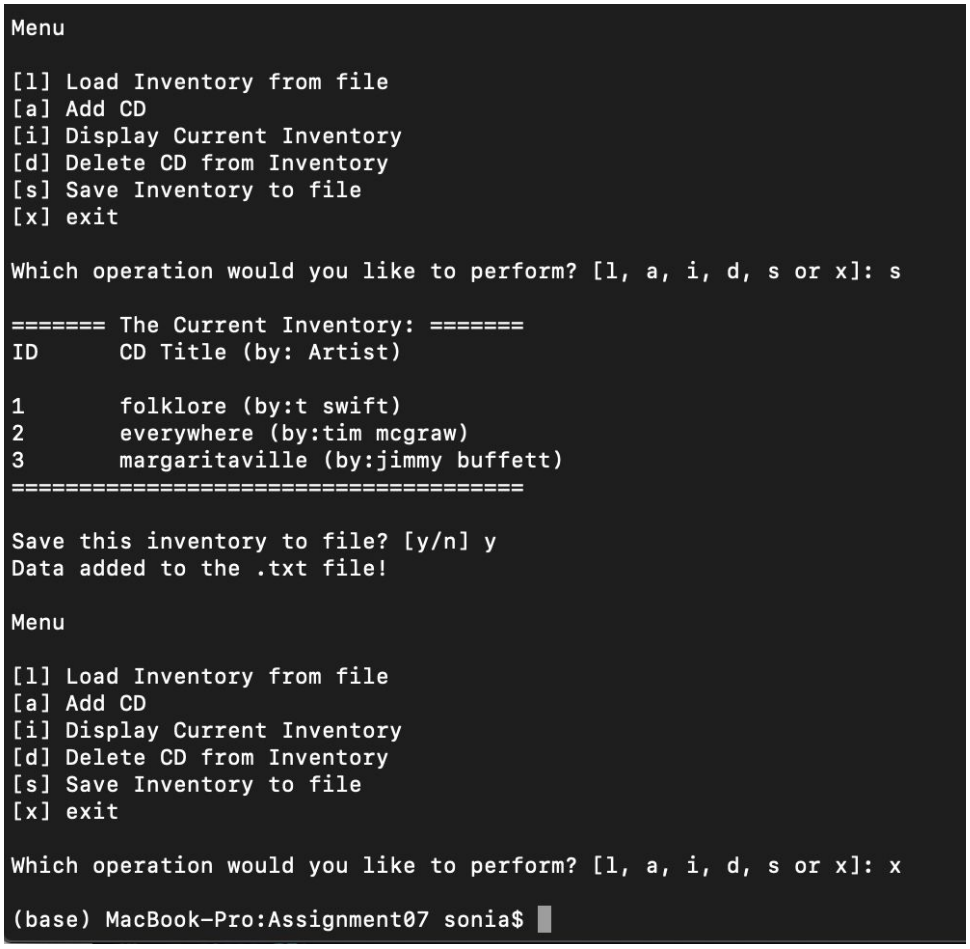
```
Menu

[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 08 - Spyder run screenshot #3



```
(base) MacBook-Pro:Assignment07 sonia$ 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]: i

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

1       folklore (by:t swift)
2       everywhere (by:tim mcgraw)
======================================

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? margaritaville
What is the Artist's name? jimmy buffett
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       folklore (by:t swift)
2       everywhere (by:tim mcgraw)
3       margaritaville (by:jimmy buffett)
======================================
```

Figure 09 - Terminal run screenshot #1



```
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       folklore (by:t swift)
2       everywhere (by:tim mcgraw)
3       margaritaville (by:jimmy buffett)
======================================

Save this inventory to file? [y/n] y
Data added to the .txt file!

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

(base) MacBook-Pro:Assignment07 sonia$ 
```

Figure 10 - Terminal run screenshot #2

Summary

This week was incredibly frustrating, but I did learn how to use binary files and structured error handling.

Appendix

Homework source code:

```
#------------------------------------------#
# Title: CDInventory.py
# Desc: Working with try / except statements and binary files.
# Change Log: (Who, When, What)
# SRomo, 2020-Aug-25, Created File
# SRomo, 2020-Aug-25, Added try / except statements
# SRomo, 2020-Aug-26, Added binary file code
# SRomo, 2020-Aug-26, Added higher level try / except statement
```

```python
#-------------------------------------------#

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:
    """Processing the in-memory data"""

    @staticmethod
    def add_cd_to_table(cdId, cdTitle, cdArtist, table):
        """Function to add the user input into the table

        Args:
            cdID: CD ID
            cdTitle: title of CD
            cdArtist: artist

        Returns:
            table
        """

        dicRow = {'ID': cdId, 'Title': cdTitle, 'Artist': cdArtist}
        table.append(dicRow)

        return table


    @staticmethod
    def remove_cd(table, userInput):
        """Function to remove a CD from the in-memory table

        Args:
            table (list of dict): 2D data structure (list of dicts) that holds the data
during runtime
            userInput: user inputted ID to delete

        Returns:
            None.

        """
        intRowNr = -1
        blnCDRemoved = False
        for row in table:
            intRowNr += 1
            if row['ID'] == userInput:
                del table[intRowNr]
                blnCDRemoved = True
                break
        if blnCDRemoved:
            print('The CD was removed')
        else:
            print('Could not find this CD!')


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 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 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

        try:
            with open(file_name, 'rb') as objFile:
                data = pickle.load(objFile)

                for row in data:
                    table.append(row)

                return table

        except FileNotFoundError:
            print('The file doesn\'t exist. No data could be loaded.\n')
        except:
            print('Something else went wrong.\n')



    @staticmethod
    def write_file(file_name, table):
        """Function to write data from the table to a file

        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
        """

        try:
            with open(file_name, 'wb') as objFile:
                pickle.dump(table, objFile)
            print('Data added to the .txt file!\n')
        except FileNotFoundError:
            print('The file doesn\'t exist. No data could be loaded.\n')
        except:
            print('Something else went wrong.\n')



# -- 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 string 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('======================================\n')

    @staticmethod
    def add_cd():
        """Allows user to add a CD

        Args:
            strID: user input for CD ID
            strTitle: CD title
            strArtist: artist name

        Returns:
            intID, strTitle, strArtist

        """

        try:
            intID = int(input('Enter ID: ').strip())
        except ValueError:
            print('Please enter an integer.')
```

```python
            intID = int(input('Enter ID: ').strip())

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

        return intID, strTitle, strArtist



# 1. When program starts, read in the currently saved Inventory
FileProcessor.read_file(strFileName, lstTbl)

# 2. start main loop

while True:
    try:
        # 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 from file
        if strChoice == 'l':
            print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
            strYesNo = input('Do you want to continue? [y/n] ')
            if strYesNo.lower() == 'y':
                print('reloading...')
                FileProcessor.read_file(strFileName, lstTbl)
                IO.show_inventory(lstTbl)
            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
            intID, strTitle, strArtist = IO.add_cd()

            # 3.3.2 Add item to the table
            DataProcessor.add_cd_to_table(intID, strTitle, strArtist, lstTbl)
            IO.show_inventory(lstTbl)
            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.
        # 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
            intIDDel = int(input('Which ID would you like to delete? ').strip())
            # 3.5.2 search thru table and delete CD
            DataProcessor.remove_cd(lstTbl, intIDDel)
            IO.show_inventory(lstTbl)
            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
                FileProcessor.write_file(strFileName, lstTbl)
            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')

    except KeyboardInterrupt:
        strYesNo = input('Are you sure you want to quit? [y/n] ').strip().lower()
        if strYesNo == 'y':
            break
        else:
            IO.print_menu()
```