Martin Stallworth

26 August 2020

Foundations of Programming, Python

Assignment 07

# Working with Files, Binary Files, and Structured Error Handling

## Introduction

During this week's assignment we pulled a lot from some things we have learned about in previous labs and expanded upon them. We have previously worked with files but this week we learned different approaches we can take to writing, appending, and reading a file. In addition, we were introduced to binary files and moved on from the simple text files we have been using. We also got a refresher on using program arguments to create distinct operational paths. Lastly, we got more information about structured error handling so that our programs can learn how to handle various errors that may occur.

## Getting the Background

We began this module by continuing to work with text files. We were shown different approaches to some of the functions we already knew when dealing with text files. Personally, I had only used one approach when opening a file, but I found the "with" approach to be a lot more convenient because you do not have to worry about closing the file when you a finished adding code. The aspect of this part of the module that I found most confusing was when the focus was on reading the data because that function has been what I have struggled the most with but the Saravji's Hut "Module 7 Part 1" video helped me grasp it decently well. After some practice with this we moved on to binary files which was something new for me. The textbook did a good job explaining the difference between binary and text files as well as what pickling a file does (Dawson 199-202). I got some additional information on this to help me with the labs and assignment from the 'Geeks for Geeks' website. To my understanding binary files can store more complex data while pickling preserves and stores this data into the file.

While working with binary files we also got a refresher on how to use program arguments to create distinct operational paths in your program. I can see how this gives you further manipulation of your program because they do not require user interaction. This was straightforward once I realized how to utilize this in the Spyder IDE by passing in the command line arguments. Lastly, we got a deeper view into structured error handling. This seems like a great tool to keep your program running smoothly when inevitable errors or mistakes arise. They can make it a lot easier on the user to understand where they went wrong when interacting with the program. I found Real Python's introduction to python exceptions to be very helpful to get a deeper look into handling errors. I mostly used the try and except clause when it came to my work this week.

## Working with Labs and Assignment

I had a long struggle with the labs this week that definitely induced near hand-banging moments but with a little help I was able to put it together. For some reason I had the most difficulty with Lab A as I struggled to write and read the file so that the two variables would accept the two values that were in the file. Initially, I did not know how to write the two numbers into the file since I was used to using a user input. I eventually came up with a method that I had used in previous labs which was the ".format" method. I also had a lot of difficulty with what to pass in as an argument that would get the write function to work for the inputs and outputs. Once I

figured out my method for writing I had difficulty reading the file so that the data would be read as two strings instead of one string. After going through everything with Doug I just pretty much scrapped everything because I figured there must have been an easier way. Once I sort of simplified things in my mind and looked back at previous assignments I was able to see how I could split the data. An example is provided below.

```
1.          with open(fileName, 'r') as fileObj:
2.              results = fileObj.readlines()
3.              for line in results:
4.                  line = line.strip().split(',')
5.                  return line
```

*Listing 1: Ex of reading line in a file*

Once I was able to split the data everything else and all of the equations ran smoothly. While I was working on I used what we discussed during the lecture as a reference but had some confusion on how to get the program argument portion to work but after getting some input I realized that I had not been passing in the command line arguments when I was running the program. This lab also gave me a chance to work with a binary file so I have provided an example of that below.

```
1.          with open(fileName, 'wb') as fileObj:
2.              pickle.dump(results, fileObj)
```

*Listing 2: writing a binary file*

The third lab was pretty straightforward and I played around with some of this different errors that I could come up with. As usual when I struggle a lot with the labs the actual assignment typically is not as difficult for me to figure out. Since we just had to add some structured error handling and shift from binary to text files I did not file the assignment to be too complex but I could always be mistaken. I used the “try” and “except” method around some of interactive portions of the code and checked to see if they showed up when I intentionally performed these errors. Since the data is loaded at the beginning of the code I added error handling to make sure that if a file had not already been created that it would be so that the user could move on to the rest of the program.

```
1.    try:
2.      FileProcessor.read_file(strFileName, lstTbl)
3.  except FileNotFoundError:
4.      print('\nThis file does not yet exist!\n')
5.      print('Creating new file for you...\n')
6.      FileProcessor.write_file(strFileName, lstTbl)
7.      print('File has been created.\n')
```

*Listing 3: Ex of structured error handling*

Once I was satisfied with how many of these error messages I built in to the program I tested it in Spyder and the terminal and posted my final product to GitHub.

```
In [466]: runfile('/Users/marti_000/_FDNProgramming/Assignment07/CDInventory.py', wdir='/Users/marti_00
_FDNProgramming/Assignment07')

This file does not yet exist!

Creating new file for you...

File has been created.

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 1: Structured error handling working in Spyder*

```
marti_000 — python _FDNProgramming/Assignment07/CDInventory.py — 80×24
Last login: Wed Aug 26 17:38:01 on ttys000
(base) Martins-Air:~ marti_000$ python _FDNProgramming/Assignment07/CDInventory.
py

This file does not yet exist!

Creating new file for you...

File has been created.

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 2: Structured error handling working in terminal*

## Summary

In this assignment, we continued working with text files and found new ways to write, append, and read data. I found an approach that I think I will stick with that is different from what I was using before. We also moved into using binary files to hold complex data instead of manipulating data into simple text files. We worked with using program arguments to create different pathways for our functions and we also worked more with structured error handling to make our programs more user-friendly. There were a lot of aspects from this week that I feel that I will continue to use in the future.

## Appendix


Using planetb's webpage (external reference). Retrieved 2020-Aug-26
Using Saravji's Hut YouTube Page (external reference). Retrieved 2020-Aug-26
Using Real Python's Webpage (external reference).Retrieved 2020-Aug-26
Using Geeks for Geeks webpage (external reference). Retrieved 2020-Aug-26
Dawson, Michael. Python Programming for the Absolute Beginner: 3rd Edition. Kindle ed., Cengage Learning, 2010

Link for GitHub: https://github.com/mstallworth513/Assignment07

## Listing for LAB07_A.py

```
#-----------------------------------------#
# Title: LAB07_A.py
# Desc: simple demonstrator for classes
# Change Log: MStallworth, 2020-Aug-26, added code to read and write numbers to a file)
# DBiesinger, 2030-Jan-01, Created File. MStallworth, 2020-Aug-26, Appened File.
#-----------------------------------------#

# -- 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


        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
```

```
70.
71.
72.         Args:
73.             val1: the number to divide
74.             val2: the number to divide by
75.
76.
77.         Returns:
78.             A float corresponding to the quotient of val1 and val2
79.         """
80.         return float(val1 / val2)
81.
82.
83. class IO:
84.     """A collection of the Input / Output operations """
85.
86.     def read_file(fileName):
87.         """
88.         function to read in two numbers from file fileName and return these
89.
90.         Args:
91.             fileName (string): file name to read the numbers from
92.
93.         Returns:
94.             numA (int): first number in file fileName.
95.             numB (int): second number in file fileName.
96.
97.         """
98.
99.         with open(fileName, 'r') as fileObj:
100.                     results = fileObj.readlines()
101.                     for line in results:
102.                         line = line.strip().split(',')
103.                         return line
104.
105.                 # TODone add code to read numbers from file
106.             def write_file(fileName, results):
107.                 """
108.                 function to write the math results to file fileName
109.
110.                 Args:
111.                     fileName (string): file Name to write the results to.
112.                     results (list): The results
113.
114.                 Returns:
115.                     None.
116.
117.                 """
118.
119.                 results = results
120.                 with open(fileName, 'w') as fileObj:
121.                     fileObj.write(str(results))
122.
123.                 # TODone add code to save results to file
124.
125.         # -- PRESENTATION (Input/Output) -- #
126.         print('Basic Math script. Calculating the Sum, Difference, Product and Quotient of two numbers
    .')
127.         strData = '100,2'
128.         IO.write_file(strFileInput, strData)
129.         intNumA, intNumB = IO.read_file(strFileInput)
130.         intNumA = float(intNumA)
131.         intNumB = float(intNumB)
132.         lstResults = []
133.         lstResults.append(SimpleMath.get_sum(intNumA, intNumB))
134.         lstResults.append(SimpleMath.get_diffference(intNumA, intNumB))
135.         lstResults.append(SimpleMath.get_product(intNumA, intNumB))
136.         lstResults.append(SimpleMath.get_quotient(intNumA, intNumB))
```

```
137.       IO.write_file(strFileOutput, lstResults)
```

## Listing for LAB07_B.py

```
3.  #-------------------------------------------#
4.  # Title: LAB07_B.py
5.  # Desc: Example of using binary file and program argument
6.  # Change Log: MStallworth, 2020-Aug-26, changed to binary file and used program argument)
7.  # MStallworth, 2020-Aug-26, Created File
8.  #-------------------------------------------#
9.  import sys
10. import pickle
11.
12. # -- DATA -- #
13. strFileInput = 'numbers.dat'
14. strFileOutput = 'results.dat'
15.
16.
17. # -- PROCESSING -- #
18. class SimpleMath:
19.     """A collection of simple math processing functions """
20.
21.     @staticmethod
22.     def get_sum(val1 = 0.0, val2 = 0.0):
23.         """Function for adding two values
24.
25.
26.         Args:
27.             val1: the first number to add
28.             val2: the second number to add
29.
30.
31.         Returns:
32.             A float corresponding to the sum of val1 and val2
33.         """
34.         return float(val1 + val2)
35.
36.     @staticmethod
37.     def get_diffference(val1 = 0.0, val2 = 0.0):
38.         """Function for subtracting two values
39.
40.
41.         Args:
42.             val1: the number to subtract from
43.             val2: the number to subtract
44.
45.
46.         Returns:
47.             A float corresponding to the difference of val1 and val2
48.         """
49.         return float(val1 - val2)
50.
51.     @staticmethod
52.     def get_product(val1 = 0.0, val2 = 0.0):
53.         """Function for multiplying two values
54.
55.
56.         Args:
57.             val1: the first number to multiply
58.             val2: the second number to multiply
59.
60.
61.         Returns:
62.             A float corresponding to the product of val1 and val2
63.         """
64.         return float(val1 * val2)
65.
```

```
66.     @staticmethod
67.     def get_quotient(val1 = 0.0, val2 = 0.0):
68.         """Function for dividing two values
69.
70.
71.         Args:
72.             val1: the number to divide
73.             val2: the number to divide by
74.
75.
76.         Returns:
77.             A float corresponding to the quotient of val1 and val2
78.         """
79.         return float(val1 / val2)
80.
81.
82. class IO:
83.     """A collection of the Input / Output operations """
84.
85.     def read_file(fileName):
86.         """
87.         function to read in two numbers from file fileName and return these
88.
89.         Args:
90.             fileName (string): file name to read the numbers from
91.
92.         Returns:
93.             numA (int): first number in file fileName.
94.             numB (int): second number in file fileName.
95.
96.         """
97.
98.         with open(fileName, 'rb') as fileObj:
99.             results = pickle.load(fileObj)
100.                return results
101.
102.            # TODone add code to read numbers from file
103.        def write_file(fileName, results):
104.            """
105.            function to write the math results to file fileName
106.
107.            Args:
108.                fileName (string): file Name to write the results to.
109.                results (list): The results
110.
111.            Returns:
112.                None.
113.
114.            """
115.
116.            with open(fileName, 'wb') as fileObj:
117.                pickle.dump(results, fileObj)
118.
119.            # TODone add code to save results to file
120.        @staticmethod
121.        def IO(fileName):
122.            """
123.            Parameters
124.            function to display current content of the results file and ask user for input two num
    bers and save them to file.
125.            fileName (string): file Name to write the results to.
126.
127.            Returns
128.            -------
129.            None.
130.
131.            """
132.            print(IO.read_file(strFileInput))
```

```
                print('Enter two values to replace current values.')
                try:
                    num1 = int(input('Please enter your first number:'))
                except ValueError:
                    print('\nInvalid input must enter an integer')
                    return
                num2 = int(input('Please enter your second number:'))
                if num2 == 0:
                    raise Exception('Cannot divide by zero')
                    return
                localTpl = (num1, num2)
                with open(strFileInput, 'wb') as fileObj:
                    pickle.dump(localTpl, fileObj)

            @staticmethod
            def calc(strFileInput, strFileOutput):
                """


                Parameters
                ----------
                fileName : TYPE
                    DESCRIPTION.

                Returns
                -------
                None.

                """
                print('Basic Math script. Calculating the Sum, Difference, Product and Quotient of two numbers.')
                intNumA, intNumB = IO.read_file(strFileInput)
                intNumA = float(intNumA)
                intNumB = float(intNumB)
                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)


        # -- PRESENTATION (Input/Output) -- #
        strArg = str(sys.argv[1])
        if strArg == 'IO':
            IO.IO(strFileInput)
        elif strArg == 'calc':
            IO.calc(strFileInput, strFileOutput)
```

## Listing for CDInventory,py

```
#-----------------------------------------#
# Title: CDInventory.py
# Desc: Working with classes and functions.
# Change Log: MStallworth, 2020-Aug-19, Moved code from main menu into the functions, added docstrings to added functions, created blank file, added IO's to menu to connect to corresponding function. MStallworth, 2020-Aug-26, added structured error handling and modified permanent data store to use binary data.
# DBiesinger, 2030-Jan-01, Created File. MStallworth, 2020-Aug-19, Appened File. MStallworth, 2020-Aug-26, Appened File.
#-----------------------------------------#

import pickle
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
    """Processing the data"""

    @staticmethod
    def add_data(strID, strTitle, strArtist, lstTbl):
        """Function to add data in memory using a list of dictionaries
        Adds the ID, CD Title, and Artist Name


        Args:
            strID (string): # for identifying each entry
            strTitle (string): Name of CD
            strArtist (string): Name of artist
            lstTbl (list of dic): 2D data structure (list of dicts) that holds the data during runtim
e


        Returns:
            None.
        """


        try:
            intID = int(strID)
        except ValueError:
            print('\nYour ID was not an integer!\n')
            print('This entry was not added!\n')
        dicRow = {'ID': intID, 'Title':strTitle, 'Artist':strArtist}
        lstTbl.append(dicRow)

    @staticmethod
    def del_data(intIDDel, lstTbl):
        """Function to delete data in memory
        deletes ID, CD Title, and Artist Name using the ID #


        Args:
            intIDDel (integer): integer for ID that represents entry to be deleted
            lstTbl (list of dic): 2D data structure (list of dicts) that holds the data during runtim
e


        Return:
            None.
        """
        intRowNr = -1
        blnCDRemoved = False
        for row in lstTbl:
            intRowNr += 1
            if row['ID'] == intIDDel:
                print('WARNING: If you continue ID', intIDDel,'will be deleted from the memory!')
                strYESNO = input('type \'yes\' to continue and delete row from the memory. Otherwise
the deletion will be cancelled.' )
                if strYESNO.lower() == 'yes':
                    print('\ndeleting...')
                    del lstTbl[intRowNr]
                    blnCDRemoved = True
                else:
                    input('cancelling...Entry not deleted. Press [ENTER] to continue to the menu.')
                break
        if blnCDRemoved:
            print('The CD was removed')
```

```
            print()
        else:
            print('\nCould not find this CD!')
            print()


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 runtim
    e

        Returns:
            None.
        """

            table.clear()  # this clears existing data and allows to load data from file
            with open(file_name, 'rb') as objFile:
                try:
                    data = pickle.load(objFile)
                    table.extend(data)
                except EOFError:
                    print('This is an empty file. Add and save data.')

        @staticmethod
        def write_file(file_name, lstTbl):
            """Function to write in memory data into a file
            Saves the data into identified file name from a 2D table


            Args:
                file_name (string): name of the file used to save the data
                lstTbl: 2D data structure used to save the in memory data


            Returns:
                None.
            """
            # TODone Add code here
            with open(file_name, 'wb') as objFile:
                pickle.dump(lstTbl, objFile)



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

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

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

            Args:
                None.

            Returns:
                None.
            """
```

```
144.
145.             print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory
    ')
146.             print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')
147.
148.         @staticmethod
149.         def menu_choice():
150.             """Gets user input for menu selection
151.
152.             Args:
153.                 None.
154.
155.             Returns:
156.                 choice (string): a lower case sting of the users input out of the choices l, a, i,
    d, s or x
157.
158.             """
159.             choice = ' '
160.             while choice not in ['l', 'a', 'i', 'd', 's', 'x']:
161.                 choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: '
    ).lower().strip()
162.             print()  # Add extra space for layout
163.             return choice
164.
165.         @staticmethod
166.         def show_inventory(table):
167.             """Displays current inventory table
168.
169.
170.             Args:
171.                 table (list of dict): 2D data structure (list of dicts) that holds the data during
    runtime.
172.
173.             Returns:
174.                 None.
175.
176.             """
177.             print('======= The Current Inventory: =======')
178.             print('ID\tCD Title (by: Artist)\n')
179.             for row in table:
180.                 print('{}\t{} (by:{})'.format(*row.values()))
181.             print('======================================')
182.
183.         @staticmethod
184.         def add_entry():
185.             """Adds user input into the in memory inventory
186.             using .add_data() from Data Processing Class
187.
188.
189.             Args:
190.                 None.
191.
192.
193.             Returns:
194.                 strID (string): ID for data entry
195.                 strTitle (string): Input for CD Title
196.                 strArtist (string): Input for Artist Name
197.             """
198.
199.             strID = input('Enter ID: ').strip()
200.             strTitle = input('What is the CD\'s title? ').strip()
201.             strArtist = input('What is the Artist\'s name? ').strip()
202.             return strID, strTitle, strArtist
203.
204.
205.         # TODone add I/O functions as needed
206.     # 1. When program starts, read in the currently saved Inventory
207.
```

```
208.        try:
209.            FileProcessor.read_file(strFileName, lstTbl)
210.        except FileNotFoundError:
211.            print('\nThis file does not yet exist!\n')
212.            print('Creating new file for you...\n')
213.            FileProcessor.write_file(strFileName, lstTbl)
214.            print('File has been created.\n')
215.
216.
217.        # 2. start main loop
218.        while True:
219.            # 2.1 Display Menu to user and get choice
220.            IO.print_menu()
221.            strChoice = IO.menu_choice()
222.
223.            # 3. Process menu selection
224.            # 3.1 process exit first
225.            if strChoice == 'x':
226.                print('\nGoodbye.')
227.                break
228.            # 3.2 process load inventory
229.            if strChoice == 'l':
230.
231.                print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-
    loaded from file.')
232.                strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will
     be canceled')
233.                if strYesNo.lower() == 'yes':
234.                    print('reloading...')
235.                    FileProcessor.read_file(strFileName, lstTbl)
236.                    IO.show_inventory(lstTbl)
237.                else:
238.                    input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the
    menu.')
239.                    IO.show_inventory(lstTbl)
240.                    continue  # start loop back at top.
241.
242.            # 3.3 process add a CD
243.            elif strChoice == 'a':
244.                # 3.3.1 Ask user for new ID, CD Title and Artist
245.                # TODone move IO code into function
246.                ID, Title, Artist = IO.add_entry()
247.
248.                # 3.3.2 Add item to the table
249.                DataProcessor.add_data(ID, Title, Artist, lstTbl)
250.                # TODone move processing code into function
251.
252.                IO.show_inventory(lstTbl)
253.                continue  # start loop back at top.
254.            # 3.4 process display current inventory
255.            elif strChoice == 'i':
256.                IO.show_inventory(lstTbl)
257.                continue  # start loop back at top.
258.            # 3.5 process delete a CD
259.            elif strChoice == 'd':
260.                # 3.5.1 get Userinput for which CD to delete
261.                # 3.5.1.1 display Inventory to user
262.                IO.show_inventory(lstTbl)
263.                # 3.5.1.2 ask user which ID to remove
264.                try:
265.                    intIDDel = int(input('Which ID would you like to delete? ').strip())
266.                except ValueError:
267.                    print('\nYour ID was not an integer!\n')
268.                    print('Enter a correct ID value to delete a CD\n')
269.                    # 3.5.2 search thru table and delete CD
270.                DataProcessor.del_data(intIDDel, lstTbl)
271.                IO.show_inventory(lstTbl)
272.                continue  # start loop back at top.
```

```
273.
274.                 # TODone move processing code into function
275.
276.             # 3.6 process save inventory to file
277.             elif strChoice == 's':
278.                 # 3.6.1 Display current inventory and ask user for confirmation to save
279.                 IO.show_inventory(lstTbl)
280.                 strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
281.                 # 3.6.2 Process choice
282.                 if strYesNo == 'y':
283.                     # 3.6.2.1 save data
284.                     # TODone move processing code into function
285.                     FileProcessor.write_file(strFileName, lstTbl)
286.                 else:
287.                     input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
288.                 continue  # start loop back at top.
289.             # 3.7 catch-
    all should not be possible, as user choice gets vetted in IO, but to be save:
290.             else:
291.                 print('General Error')
```