Eliana Arias-Dotson
Created: 2020, March 7th 2020
Final Editing: March 9nd 2020
Foundations of Programming, Python
Assignment 07

Introduction

Module 07 focused on working with files, how to read it, write, append, loop and managing the content of the file as a whole or per line using a series of access modes options (r,w,a,r+,w+, a+). We also started using the concept of "pickling" , unpickling and shelving data.
The last part of this module was working more on handling Exceptions and specifying Exception's types. Although the concept of creating your own exepctions was introduced, we focused on built in python functions.

Working with Files - Opening, closing,writing, appending, looping

Previously we have learned how to create a file using the open functions and the different modes to access or handle the file to create a file object containing information that can be used and manipulated on the code (1).
A new way to access the file was introduced using the "with" statement. This method seems to be a cleaner and briefer way to access the file plus it does not require the use of a close() function, it does not automatically.
I tried to exercise this new option on the provided code for the read and save functions (Appendix 1 and Figure 1)

```
@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:
        table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
    """
    try:
        with open(file_name, 'r') as objFile:
        #objFile = open(file_name, 'r')
            table.clear()  # this clears existing data and allows to load data from file
            for line in objFile:
                data = line.strip().split(',')
                dicRow = {'ID': data[0], 'Title': data[1], 'Artist': data[2]}
                table.append(dicRow)
                # if we want to pickle the information on our dicRow:
                pickle_out=open("dict.pickle","wb")
                pickle.dump(dicRow,pickle_out)
                pickle_out.close()
            objFile.close()
    except FileNotFoundError:
        print("The file {} could not be loaded".format(file_name))
    return table
```

Figure 1. Opening the file usint "with"

(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

Pickling

Pickling is a tool used fo serializing a de-serializign a python object structure such a list, boolean, integers, floating points, tuples, dictionaries, functions built in functions, classes previously defined, at a top level. When Pickling is applied a python object is converted to a byte stream. When pickling a python object is converted into a byte stream to store it in a file/ database, maintain program state across sessions, or transport data over the network [2].

Although I tried to implement the picking I did not unpickle it for any use on the script. I used it simply as a practice pickling the information stored on the dictionary

```
with open(file_name, 'r') as objFile:
        #objFile = open(file_name, 'r')
          table.clear()  # this clears existing data and allows to load data from file
          for line in objFile:
            data = line.strip().split(',')
            dicRow = {'ID': data[0], 'Title': data[1], 'Artist': data[2]}
            table.append(dicRow)
            # if we want to pickle the information on our dicRow:
            pickle_out=open("dict.pickle","wb")
            pickle.dump(dicRow,pickle_out)
            pickle_out.close()
          objFile.close()
```

If desired to unpickled this information:

```
print('\nUnpickling Dictionary")
f=open("pickle_out", "rb")  ###
ID=pickle.load(f)
Title=pickle.load(f)
Artist=pickle.load(f)
```

The functions used to pickle (or write the pickled version of the object to file is "dump" in binary format. The function "load" is used to de-pickle returning the next pickle object.

Handling Exceptions

On this assignment the purpose of the assignment was to implement handling exceptions to manage the possibility of errors and add more clarification to the code specially when interating with user [3,4].

(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

A simple way for me to understand the exceptio handling was having present the following structure:

try:
potential risky operations
except:
what to do to handle problems if they arise
else:
do this ONLY if not exceptions are caught
finally:
do this regardless of wheter there were exceptions

where the else and finally sections are optional. Initially I used the finally statement on my functions where I opened files using the "with" statement. but after there is no need to close statements when used "with" I removed it, without affecting the code (3)

We can also use the "raise" option when a condition occurs. On the assignment I tried to implement it when the possibility of a negative value is enter as and ID by the user (def get_UserInput():)

```
ValidID =False
    while not ValidID:
        try:
            ID = int(input('Enter ID: ').strip())
            if ID <=0:
                raise ValueError('That is not a positive integer number!')
        except ValueError:
            print('Please enter ID using positive integers')
        else:
            ValidID=True
```

```
Which operation would you like to perform? [l, a, i, d, s or x]: a
Thank you for entering a valid choise, please continue:

Enter ID: -4
Please enter ID using positive integers

Enter ID:
```

Figure 2. Output approving valid selection of choices and handling exception for negative entry for ID.

(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

Going through the provided code the “management” of errors seemed to me implemented withouth the use of conditions (If, else) seem to play a similar role to the Exception handling.

Although I tried to incorporate the use of exceptions I believe the correct use is not clear to me yet.

```
@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 = ' '
    try:
      choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
      while choice not in ['l', 'a', 'i', 'd', 's', 'x']:
        raise ValueError('That is not a valid choice!')
    except ValueError as e:
      print(type(e))
      print('Please enter one of the offered options from menu')
    else:
      print("Thank you for entering a valid choise, please continue:")
      return choice
```

Figure 2a. Handling exception for valid or invalid choices in menu_choices function

```
Which operation would you like to perform? [l, a, i, d, s or x]: x
Thank you for entering a valid choise, please continue:


Which operation would you like to perform? [l, a, i, d, s or x]: p
<class 'ValueError'>
Please enter one of the offered options from menu
General Error
Menu
```

Figure 2b. Output from handling exception for valid (top) or invalid (bottom) choices in menu_choices function

(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

Exceptions for handling the file

```
try:
        with open(file_name, 'r') as objFile:
        #objFile = open(file_name, 'r')
            table.clear()  # this clears existing data and allows to load data from file
            for line in objFile:
                data = line.strip().split(',')
                dicRow = {'ID': data[0], 'Title': data[1], 'Artist': data[2]}
                table.append(dicRow)
                # if we want to pickle the information on our dicRow:
                pickle_out=open("dict.pickle","wb")
                pickle.dump(dicRow,pickle_out)
                pickle_out.close()
            objFile.close()
except FileNotFoundError:
                        print("The file {} could not be loaded, please check file location or create
one".format(file_name))
    return table
```

```
The file CDInventory.txt could not be loaded, please check file location or create
one
======= The Current Inventory: =======
ID      CD Title (by: Artist)

=======================================
Menu
```

Figure 3. Handling FileNotFoundError and display to user.

Lessons Learned:

In this lesson the code seems to be taking shape to be a cleaner, well documented and better structured to handle received information from the user and manipulation of the information. Although the concepts are starting to sink and make more sense, personally I need more practice to implement correctly all the statements and make the code as clear as possible without redundant information or unnecessary warnings. I used as a starting point the corrected assignment from Module 6 wich helped me significantly since it is building some familiarity with the code and what each function does.

Working Code - work in progress, not working as desired

(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

I am aware the code is not functioning fully as expected but I have been trying to fix my own errors following Mr Kloss suggestions and figuring them out after lecture. I know is not perfect but is making more sense now. little steps i guess.

Valuable Resources of information aside from provided links and videos:

I've listed below the links that seem to make more sense to me and I used as additional information. RealPython and python.org offered very clear documentation and clarified several of the concepts. There are a lot of videos and websites related to python scripting, as we moved along I am trying to stay away from sites that offer solutions or approaches by other programmers since it seems to confuse me more, so I'm just trying to digest slowly information from the class book, which has simple examples and codes as well as main websites and well recommended online tutorials.

GitHub

Link to the my assignment and personalized README on Github is located under

https://github.com/elidot/Assignment07
with an edited README

Appendix 1 - Full Code -

```
# Title: Assignment07.py
# Desc: Working with classes and functions.
# Change Log: Eliana Arias-Dotson 2020, Jan 27 Created File
# Modified  : Eliana Arias-Dotson 2020, Jan 28 moved processing codes into
functions
# Modified  : Eliana Arias-Dotson 2020, Jan 29 understanding flow
# Modified  : Eliana Arias-Dotson 2020, March 2 Workign script, editing
docstrings
# Modified  : Douglas Klos, 2020 March 3, Refactored functions, grading.
# Modified  : Eliana Arias-Dotson, 2020, March 7th-8th - Adding Exception
clauses, pickling and cleaning up
# Modified  : Eliana Arias-Dotson, 2020, March 9th - Cleaning up printing
statements and test for submission
#------------------------------------------#
```


(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

```
import pickle

# -- PROCESSING -- #
#Definition of Classes and functions:

class DataProcessor:
    """A set of functions to load, add and delete data from Magic
Inventory"""
    @staticmethod
    def add_cd(cd_id, cd_title, cd_artist, table):
        """Function to add new data to list.
        Args:
            cd_iD (string): ID representing the ID of the new CD
            cd_title (string): Title of the new CD
            cd_artist (string): Artist of the new CD
            table (list of dict): 2D data structure (list of dicts) that
holds the data during runtime

        Returns:
            table (list of dict): 2D data structure (list of dicts) that
holds the data during runtime
        """
        new_cd = {'ID': cd_id, 'Title': cd_title, 'Artist': cd_artist}
        table.append(new_cd)
        return table

#function to delete delete_ID
    @staticmethod
    def delete_cd(cd_id, table):
        """Function to delete entry associated to a user data entry
           to be deleted and searchs through the list of dict to delete the
           ID entry

        Args:
```

(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

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

        Returns:
            table (list of dict): 2D data structure (list of dicts) that
holds the data during runtime
        """
        # 3.5.1.2 ask user which ID to remove



        # 3.5.2 search thru table and delete CD
        intRowNr = -1
        blnCDRemoved = False
        try:
            for row in table:
                intRowNr += 1
                if row['ID'] != cd_id:
                    del table[intRowNr]
                    blnCDRemoved = False
                    #break
        except:
            print('Could not find this CD!')
        else:
            if blnCDRemoved:
               print('The CD was removed')
        return table


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

    @staticmethod
    def read_file(file_name, table):
```


(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

```
        """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:
            table (list of dict): 2D data structure (list of dicts) that
holds the data during runtime
        """
        try:
            with open(file_name, 'r') as objFile:
            #objFile = open(file_name, 'r')
                table.clear()  # this clears existing data and allows to load
data from file
                for line in objFile:
                    data = line.strip().split(',')
                    dicRow = {'ID': data[0], 'Title': data[1], 'Artist':
data[2]}
                    table.append(dicRow)
                    # if we want to pickle the information on our dicRow:
                    pickle_out=open("dict.pickle","wb")
                    pickle.dump(dicRow,pickle_out)
                    pickle_out.close()
                objFile.close()
        except FileNotFoundError:
            print("The file {} could not be loaded, please check file
location or create one".format(file_name))
        return table
```


(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

```
    @staticmethod
    def write_file(file_name, table):

        """Function to write data to file from list

        Opens the data file with option to write, loops through the new row
added to list
        and adds the new entry to file
        (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, Write to file
        """
        try:
            with open(file_name, 'w') as objFile:
                for row in table:
                    lstValues = list(row.values())
                    lstValues[0] = str(lstValues[0])
                    objFile.write(','.join(lstValues) + '\n')
        except IOError:
            print("Error! Couldnot find file or read data")



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

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


(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

```
    @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 = ' '
        try:
            choice = input('Which operation would you like to perform? [l, a,
i, d, s or x]: ').lower().strip()
            while choice not in ['l', 'a', 'i', 'd', 's', 'x']:
                raise ValueError('That is not a valid choice!')
        except ValueError as e:
```


(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

```
            print(type(e))
            print('Please enter one of the offered options from menu')
        else:
            print("Thank you for entering a valid choise, please continue:")
            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('=======================================')

    @staticmethod
    def get_UserInput():
       """Function to accept User input

       Gets input from user for ID, Title and Artist to be save in table

       Args:
           None

       Returns:
```


(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

```
            ID (string): ID representing the ID of the new CD
            Title (string): Title of the new CD
            Artist (string): Artist of the new CD
        """
        ValidID =False
        while not ValidID:
            try:
                ID = int(input('Enter ID: ').strip())
                if ID <=0:
                    raise ValueError('That is not a positive integer number!')
            except ValueError:
                print('Please enter ID using positive integers')
            else:
                ValidID=True
        Title = input('What is the CD\'s title? ').strip()
        Artist = input('What is the Artist\'s name? ').strip()
        return ID, Title, Artist


# MAIN PROCESSING
# -- DATA -- #
strChoice = '' # User input
lstTbl = []  # list of lists to hold data
dicRow = {}  # dictionary of data row
strFileName = 'CDInventory.txt'  # data storage file
objFile = None  # file object
strID =None
strTitle = None
strArtist= None

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

# 2. start main loop
while True:
```


(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

```
    # 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...')
            lstTbl = 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':
        strID, strTitle, strArtist = IO.get_UserInput()
        lstTbl = DataProcessor.add_cd(strID, 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':
```


(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

```
        # 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
        strIDDel = input('Which ID would you like to delete? ').strip()
        # 3.5.2 search thru table and delete CD
        lstTbl = DataProcessor.delete_cd(strIDDel, lstTbl)
        IO.show_inventory(lstTbl)
        continue
    # 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
        strYesNo = ' '
        strYesNo = input('Save this inventory to file? [y/n]
').strip().lower()
        try:
            while strYesNo not in ['y','n']:
                raise ValueError('That is not a valid choice!')
        except Exception as error:
            print('Please enter y for yes or n for no. No other inputs are
valid')
            print(error)
        else:
            if strYesNo == 'y':
            # 3.6.2.1 save data
                FileProcessor.write_file(strFileName, lstTbl)
                print('Data saved to file')
            if strYesNo=='n':
                input('The inventory was NOT saved to file. Press [ENTER] to
return to the menu.')
```


(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/

```
        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')
```


(1) https://www.pythonforbeginners.com/files/reading-and-writing-files-in-python
(2) https://docs.python.org/2/library/pickle.html#the-pickle-protocol
(3) https://docs.python.org/3/tutorial/errors.html
(4) https://realpython.com/python-exceptions/
(5) https://www.boldgrid.com/support/wordpress-tutorials/commit-a-file-change-on-github/