

Bitam Massoudi

11/25/2022

Foundations of Programming: Python

Assignment 07

[bitamass/Assignment07 \(github.com\)](https://github.com/bitamass/Assignment07)

Binary Files and Structured Error Handling

Introduction

In this assignment we learned about structured error handling and binary files. In computer programming, exception handling is the process of responding to exceptions (conditions that require special processing during the execution of a program). Generally, the exception breaks the normal flow of the execution and executes a pre-registered exception handler.

Binary files are usually thought of as being a sequence of bytes, which means the binary digits (bits) are grouped in eights. Binary files typically contain bytes that are intended to be interpreted as something other than text characters.

Binary data storage

For efficiency purposes, one can use pickle to load and save data in the binary format. Instead of reading and writing row by row, one can use pickle.dump to save everything to .dat (Fig1a), then pickle.load everything out from this .dat file (fig1b).

```
with open('CDInventory.dat', 'wb') as file:  
    pickle.dump(table, file, protocol=pickle.HIGHEST_PROTOCOL)
```

(Fig 1a) pickle.dump

```
with open('CDInventory.dat', 'rb') as file:  
    table = pickle.load(file)  
    print(table)
```

(Fig 1b) pickle.load

Structured Error Handling-User Interaction

Invalid choice: In menu choice function, if the user chooses something other than [l,a,i,d,s,x], or nothing at all, an exception is raised to show the user “This is an invalid choice”. This while-loop will run until the user chooses one of the following: [l,a,i,d,s,x]. (Fig 2a, b) illustrate the script and the output.

```
@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 = ' '
    #Error Handling if user doesnt enter anything in [l, a, i, d, s, x]

    while True:
        try:
            choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
            if (choice == 'a') or (choice == 'l') or (choice == 'i') or (choice == 'd') or (choice == 's') or (choice == 'x')
                break
            else: raise Exception
        except Exception:
            print('This is an invalid choice! Please try again!!')
    print() # Add extra space for layout
    return choice
```

Fig 2a Script of user interaction for menu choice

```
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]: q
This is an invalid choice! Please try again!!
Which operation would you like to perform? [l, a, i, d, s or x]:
This is an invalid choice! Please try again!!
Which operation would you like to perform? [l, a, i, d, s or x]:
```

Fig 2b output illustrating error handling for menu choice (wrong choice or nothing entered)

The delete function, also has exception raised. If the user chooses to delete an ID that does not exist (ID is not in data inventory or the user selected a non-integer), they will receive notification to enter another value (Fig 3 a, b)

```

def delete_file (table):
    """Function to delete data from user input from dic.
    Args:
        intIDDel-ID number to be deleted
    Returns:
        None
    """
    # TODO: improve error handling if the user tries deleting a non-numeric ID
    intIDDel = '' # set intIDDel to empty
    while type(intIDDel) != int: # don't let the user go anywhere until they select a numeric ID
        try:
            intIDDel = int(input('Which ID would you like to delete? ').strip())
        except ValueError:
            print('\nSorry, that didn't work. Please select an ID from your current inventory.\n')
            intIDDel = int(input('Which ID would you like to delete? ').strip())

    intRowNr = -1
    blnCDRemoved = False
    for row in table:
        intRowNr += 1
        if int(row['ID']) == intIDDel:
            del table[intRowNr]
            blnCDRemoved = True
            break
    if blnCDRemoved:
        print('The CD # ',intIDDel,' was removed')
    else:
        #print('Could not find this CD!')
        #Error handling against non integers
    try:
        if blnCDRemoved == True:
            print ('The CD Number', intIDDel, 'was removed')
        if blnCDRemoved == False:
            raise Exception
    except Exception:
        print('\nPlease select an ID from the inventory.\n')

    return table

```

Fig 3a Script for user interaction for delete function

```

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

1   American Girl (by:Tom Petty)
2   Stay with me (by:The Faces)
3   1969 (by:The stooges)
=====
Which ID would you like to delete? 6

Please select an ID from the inventory.

Menu

[1] 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? [1, a, i, d, s or x]: d

[{'ID': 1, 'Title': 'American Girl', 'Artist': 'Tom Petty'}, {'ID': 2, 'Title': 'Stay with
me', 'Artist': 'The Faces'}, {'ID': 3, 'Title': '1969', 'Artist': 'The stooges'}]

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

1   American Girl (by:Tom Petty)
2   Stay with me (by:The Faces)
3   1969 (by:The stooges)
=====
Which ID would you like to delete? 2.5

Sorry, that didn't work. Please select an ID from your current inventory.

Which ID would you like to delete? d

Sorry, that didn't work. Please select an ID from your current inventory.

Which ID would you like to delete? Abba

Sorry, that didn't work. Please select an ID from your current inventory.

Which ID would you like to delete? |

```

Fig 3b Output error handling for delete function (deleting an ID non-existent in CD Inventory, deleting a floating int or a non-integer ID)

In case the user inputs a non-integer value for adding a CD ID, the structured error handling “ValueError” will remind the user to choose an integer. (Fig 4 a, b)

```
@staticmethod
def add_file():
    """user input, and return values for, CD ID, Title, and Artist
    try - except method to ensure users inputs an integer ID.
    Args:
        None.
    Returns:
        strID (string): user selected ID value
        strTitle (string): user input for CD name
        strArtist (string): user input for Artist
    """
    strID = ''
    while True:
        strID = input('Enter ID: ').strip()
        #Error Handling if user enter non-numeric value
        try:
            intID = int(strID)
            break
        except ValueError:
            print('Please choose an integer! Try again!')
    strTitle = input('What is the CD\'s title? ').strip()
    strArtist = input('What is the Artist\'s name? ').strip()
    return intID, strTitle, strArtist
```

Fig 4a script to add CD input

```
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: abba
Please choose an integer! Try again!
Enter ID: 2.5
Please choose an integer! Try again!
Enter ID: |
```

Fig 4b output for structured error handling (string to int)

If there is no file to save to, structured error handling (File access) “FileNotFoundError” will occur and the user is reminded that the file was not found. I removed the .dat file and ran the script. Fig 5

```
try:
    FileProcessor.read_file(datFileName, lstTbl)
except FileNotFoundError:
    print ('File not found!\n')
    break
```

Fig 5a User interaction-File access structured error handling

```
In [1]: runfile('C:/_FDProgramming/Assignment07/CD inventory.py',
wdir='C:/_FDProgramming/Assignment07')
File not found!
```

Fig 5b output-File not found message

Running the script

Adding to the CD inventory: I have added CD names and artists and the output is illustrated in Fig 6.

```
What is the CD's title? American Girl
What is the Artist's name? Tom Petty
1  American Girl (by:Tom Petty)
=====
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? 1969
What is the Artist's name? The Stooges
1  American Girl (by:Tom Petty)
2  1969 (by:The Stooges)
=====
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? Stay With Me
What is the Artist's name? The Faces
1  American Girl (by:Tom Petty)
2  1969 (by:The Stooges)
3  Stay With Me (by:The Faces)
```

Fig 6. Add function output

Next, I saved the list of the CDs that I added above to the file. Fig 7a illustrates the save function output. The file is saved on CDInventory.dat Fig 7b (created by Python pickling script as .dat format)

```

Which operation would you like to perform? [l, a, i, d, s or x]: s

[{'ID': 1, 'Title': 'American Girl', 'Artist': 'Tom Petty'},
{'ID': 2, 'Title': '1969', 'Artist': 'The Stooges'}, {'ID': 3, 'Title': 'Stay With Me', 'Artist': 'The Faces'}]

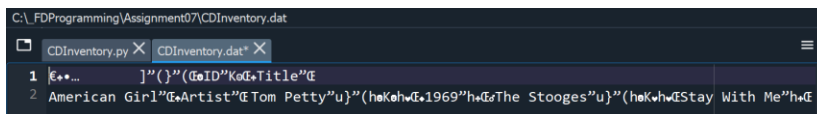
===== The Current Inventory: =====
ID  CD Title (by: Artist)
1   American Girl (by:Tom Petty)
2   1969 (by:The Stooges)
3   Stay With Me (by:The Faces)
=====
Save this inventory to file? [y/n] y
Save this inventory to file? [y/n] y
Menu

[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]:

```

Fig 7a Save function output from Python program



The screenshot shows a Windows file explorer window. The address bar displays the path 'C:_FDProgramming\Assignment07\CDInventory.dat'. The file list shows a single file named 'CDInventory.dat' with a size of 0 bytes.

Fig7b file saved in .dat format

Display function works as I can see the list of the 3 CDs added to the file Fig 8.

```

Which operation would you like to perform? [l, a, i, d, s or x]: i

[{'ID': 1, 'Title': 'American Girl', 'Artist': 'Tom Petty'},
{'ID': 2, 'Title': '1969', 'Artist': 'The Stooges'}, {'ID': 3, 'Title': 'Stay With Me', 'Artist': 'The Faces'}]

===== The Current Inventory: =====
ID  CD Title (by: Artist)
1   American Girl (by:Tom Petty)
2   1969 (by:The Stooges)
3   Stay With Me (by:The Faces)
=====
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]:

```

Fig 8. Display function output

Delete function was tested. I was able to delete ID #1 and the remaining CD list is loaded and displayed in Fig 9a, b.

```

Which operation would you like to perform? [l, a, i, d, s or
x]: d

[{'ID': 1, 'Title': 'American Girl', 'Artist': 'Tom Petty'},
{'ID': 2, 'Title': '1969', 'Artist': 'The Stooges'}, {'ID':
3, 'Title': 'Stay With Me', 'Artist': 'The Faces'}]

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

1   American Girl (by:Tom Petty)
2   1969 (by:The Stooges)
3   Stay With Me (by:The Faces)
=====
Which ID would you like to delete? 1
The CD # 1 was removed
The CD Number 1 was removed
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

[{'ID': 2, 'Title': '1969', 'Artist': 'The Stooges'}, {'ID':
3, 'Title': 'Stay With Me', 'Artist': 'The Faces'}]

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

2   1969 (by:The Stooges)
3   Stay With Me (by:The Faces)
=====

```

Fig 9a. Delete Function output from program-deleting CD#1

```

C:\_FDProgramming\Assignment07\CDInventory.dat
CDInventory.py* X CDInventory.dat X
1  """({ID}"K{Title}"1969"Artist"
2  The Stooges"u}"(hKhw
3  Stay With Me"u}"(hKhwWhole Lotta Love"hw

```

Fig 9b. CDInventory.dat file showing the updated list after deletion of CD#1

Conclusion:

In this assignment, I learned how to run structured error handling. This is important because structured error handling can potentially reduce errors that could be raised due to user interactions. I also learned about binary data storage. By using the binary data storage, the data can be read and saved as whole (dump and load) rather than line by line.