Lab 8 Keynotes:

**tkContacts.py**

Import your myDatabase.py file

Make use of a record pointer!!

Ex.

**currentID = 0**

Build a new list

Ex.

**contacts = []**

Listbox binding: sort list on last name=> contacts.sort(key = lambda x : x[1])

Include CRUD method calls where necessary for Add, Delete, Update functions

Ex.

# Function to add record which calls the insertDataDB in myDatabase

def addContact () :

phone = phoneVar.get()

name = nameVar.get()

# To perform validation if fields are empty and through error message

if (name == "" or phone == ""):

name = None

phone = None

myDatabase.insertContact(name, phone)

else:

# insert data on providing complete details

myDatabase.insertContact(name, phone)

::

**myDatabase.py** (SQLite)

\*Build **contacts.db**

\*Main import -> **import** **sqlite3**

Best practices:

-Error traps

Watch for nulls hitting DB!!

-Use of cursor

-Console messaging

-Commit entries?

-Comment often!

Make use of fetch commands for various result sets

To retrieve data after executing a SELECT statement, you can either treat the cursor as an iterator, call the cursor’s fetchone() method to retrieve a single matching row, or call fetchall() to get a list of the matching rows.

Ex.

conn = sqlite3.connect('example.db')

c = conn.cursor()

c.execute('SELECT \* FROM aTable WHERE symbol=?', t)

print(c.fetchone())

c.execute("SELECT \* FROM aTable")

rows = c.fetchall()

print('Total Row(s):', c.rowcount)

for row in rows:

    print(row)

#or fetch via use of iterator

for row in c.execute('SELECT \* FROM aTable ORDER BY something'):

print(row)

**myDatabasefile.py**

CRUD Method operations – Use cases

-CREATE -if table does not exist -> build table, insert records from contacts.py

-READ -return tuple or row result set

\*Executes & returns result set

\*Rows seen as ‘tuples’

-UPDATE – update record(s) by id

-DELETE – delete record(s) by id

-INSERT – grab tuple (name, phone)

[Queries](https://docs.python.org/3/library/sqlite3.html):

To connect to a db / create connection object

-conn = sqlite3.connect('contacts.db')

To execute any query for CRUD action

-conn.execute(sql string)

-cursor.execute(sql, data)

Make use of prepared statements (placeholders)!

def insertContact(name,phone):

conn = sqlite3.connect('contacts.db')

sql = ("INSERT INTO JAMES "

"(NAME, PHONE) "

"VALUES (?, ?)")

data = (name, phone)

cursor = conn.cursor()

# Try block to throw exception if fields are blank

try:

cursor.execute(sql, data)

conn.commit()

except Exception as e:

conn.rollback()

messagebox.show

**myDatabasefile.py**

\*Passing an id to process query into methods!!!

# Function to delete contact from database

def deleteContact(id):

conn = sqlite3.connect('contacts.db')

data = str(id)

sql = "DELETE FROM JAMES WHERE ID = ?;"

conn.execute(sql , (data))

conn.commit()

conn.close()

print("deleted contact from DB")

# Function to update a record

def updateContact(id,name,phone):

::

Extra efforts!

Rollbacks

Commits

Messaging