CSC 157

Name \_James Aniciete\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date \_\_5/9/2020\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Lab No. \_\_15\_\_\_\_\_

Purpose of the Lab Activity

**The purpose of this lab activity is to reconfigure Lab 8 to work with a SQLite database.**

Source Code

**contacts.py:**

# Modifications from original:

# James Aniciete was added but removed to follow snapshot directions

# Lebron ==> LeBron

# Janet Jackson deleted

contactlist = [

['James, LeBron', '457-6223'],

['Manfredi, Ralph', '872-2221'],

['Robbins, Tony', '329-2310'],

['Siemens, Harper', '323-4149'],

['Smith, Patti', '239-1212'],

['Thompson, Bobby', '365-2622'],

['Ziegler, Zig', '667-1101'],

]

**myDatabasefile.py:**

# Programmer: James Aniciete

# Course No.: CSC 157

# Lab No.: 15

# Date: 5/9/2020

import sqlite3

from contacts import \* # for contactlist

# open a db connection

db = sqlite3.connect("contacts.db")

# get a cursor object to execute SQLite statements in Python

c = db.cursor()

tblName = "James"

# if table already exists, uncomment this

c.execute(f"DROP TABLE {tblName}")

# function to create a "James" table and import data from contacts.py from the end of Lab 8

def createTable():

# try to create the table

try:

c.execute(f"CREATE TABLE IF NOT EXISTS {tblName}(name TEXT, phone TEXT)")

print(f"Table {tblName} has been created.")

# insert contactlist data from contacts.py into the contacts table

for row in contactlist:

c.execute(f"INSERT INTO {tblName} VALUES('{row[0]}', '{row[1]}')")

print(f"Importing data from contacts.py into the table '{tblName}'")

db.commit()

# display exception message if table already exists

except:

print(f"Table {tblName} already exists.")

# insert function

def insert(name, phone):

# put the commits before the modifications to allow a rollback

db.commit()

c.execute(f"INSERT INTO {tblName} VALUES ('{name}', '{phone}')")

print(f"A contact has been added with:\n\t Name: {name}\n\t Phone: {phone}")

# update function

def update(oName, oPhone, nName, nPhone): # o = old / n = new

db.commit()

c.execute(f"UPDATE {tblName} \

SET name = '{nName}', phone = '{nPhone}'\

WHERE name = '{oName}' AND phone = '{oPhone}'")

print(f"A contact has been updated:\n\tOriginal: {oName} and {oPhone}\n\tUpdated: {nName} and {nPhone}")

# delete function

def delete(name, phone):

db.commit()

c.execute(f"DELETE FROM {tblName}\

WHERE name = '{name}' OR phone = '{phone}'")

print(f"A contact has been deleted for:\n\t Name: {name}\n\t Phone: {phone}")

# function to select \* from table

def selectAll():

c.execute(f"SELECT \* FROM {tblName}")

return c.fetchall()

# function to rollback

def rollback():

db.rollback()

print("The last action was rolled back.")\

**tkContacts.py:**

# Programmer: James Aniciete

# Course No.: CSC 157

# Lab No.: 15

# Date: 5/9/2020

from tkinter import \*

from tkinter import messagebox # for exit button's messagebox

import os # for exiting the app

import myDatabasefile as dbf

import sqlite3

# create table

dbf.createTable()

# get contactlist

contactlist = dbf.selectAll()

# function to check for valid data entries

def validate(s): # s for string

if s.strip("") != "":

return True

else:

return False

# function to get the selection from the listbox

def selection():

return int(select.curselection()[0])

# function to add a contact

def addContact():

if validate(nameVar.get()) == True and validate(phoneVar.get()) == True:

dbf.insert(nameVar.get(), phoneVar.get())

canRoll = True

# refresh the GUI

refresh()

elif validate(nameVar.get()) == False:

print("Error: Enter a name.")

elif validate(phoneVar.get()) == False:

print("Error: Enter a phone number.")

else:

print("Error: Contact not added.\nMake sure that the Name and Phone fields are filled.")

# function to update a contact

def updateContact():

if validate(nameVar.get()) == True and validate(phoneVar.get()) == True:

dbf.update(oName, oPhone, nameVar.get(), phoneVar.get())

canRoll = True

# refresh the GUI

refresh()

elif validate(nameVar.get()) == False:

print("Error: Enter a name.")

elif validate(phoneVar.get()) == False:

print("Error: Enter a phone number.")

else:

print("Error: Contact not updated.\nMake sure a contact is selected and that the Name and Phone fields are filled.")

# function to delete a contact

def deleteContact():

try:

if messagebox.askokcancel(title = "Delete Contact",

message = f"Are you sure you want to delete {contactlist[selection()][0]}'s contact information?") == 1:

dbf.delete(nameVar.get(), phoneVar.get())

canRoll = True

refresh()

except:

print("Error: Select a contact to be deleted.")

# function to load a contact

def loadContact():

try:

# not really sure how this works

global oName, oPhone

oName = contactlist[selection()][0]

oPhone = contactlist[selection()][1]

# put name and phone selections into a tuple

name, phone = contactlist[selection()]

# use tuple to assign values to name and phone variables

nameVar.set(name)

phoneVar.set(phone)

except:

print("Error: Select a contact from the list.")

# function to rollback a change

def rollback():

global canRoll

if canRoll == True:

if (messagebox.askokcancel(title = "Rollback", message = "Would you like to undo the previous change?") == 1):

dbf.rollback()

refresh()

canRoll = False

# function to exit the program

def exitContact():

app\_title = "Contacts"

if messagebox.askokcancel(title = app\_title, message = "Do you want to exit, OK or Cancel") == 1:

# commit and close the database

dbf.db.commit()

dbf.db.close()

os.\_exit(1)

# function that places all widgets into the frame individually

def buildFrame () :

# define global variables

global nameVar, phoneVar, select

# create the main window widget

root = Tk()

# add title to the frame

root.title("My Contact List")

# create & pack a frame in the root window

frame1 = Frame(root)

frame1.pack()

# on 1st row of frame:

# create a label for name

Label(frame1, text="Name:").grid(row=0, column=0, sticky=W)

# initialize StringVar for name

nameVar = StringVar()

# assign entry button value to the name var

name = Entry(frame1, textvariable=nameVar)

# position name var in first row, second column, aligned to the west cell border

name.grid(row=0, column=1, sticky=W)

# on 2nd row of the frame:

# create a label for phone no.

Label(frame1, text="Phone:").grid(row=1, column=0, sticky=W)

# create string var for phone no.

phoneVar= StringVar()

# assign entry button value to phone var

phone= Entry(frame1, textvariable=phoneVar)

# position phone var in second row, second column, aligned to the west

phone.grid(row=1, column=1, sticky=W)

# create & pack a frame in the root window

frame1 = Frame(root)

frame1.pack()

# add a row of buttons to frame1 with respective callback functions

btn1 = Button(frame1,text=" Add ",command=addContact)

btn2 = Button(frame1,text="Update",command=updateContact)

btn3 = Button(frame1,text="Delete",command=deleteContact)

btn4 = Button(frame1,text=" Load ",command=loadContact)

btn5 = Button(frame1,text="Rollback",command=rollback)

# pack the buttons on the same row to the left

btn1.pack(side=LEFT)

btn2.pack(side=LEFT)

btn3.pack(side=LEFT)

btn4.pack(side=LEFT)

btn5.pack(side=LEFT)

# allow for selection of names from a ListBox with a scrollbar

frame1 = Frame(root)

frame1.pack()

# create a vertical bar widget

scroll = Scrollbar(frame1, orient=VERTICAL)

# whichever value from the ListBox is clicked is assigned to select

# height = # of values visible in the Listbox

select = Listbox(frame1, yscrollcommand=scroll.set, height=8)

scroll.config (command=select.yview)

scroll.pack(side=RIGHT, fill=Y)

select.pack(side=LEFT, fill=BOTH)

# create frame for Exit button at the bottom of the window

frame2 = Frame(root)

frame2.pack()

# create exit button & pack it

btn6 = Button(frame2, text = " Exit ", command = exitContact)

btn6.pack()

# return root object to allow for the frame to be built

return root

# sorts the contact list & allows for an update to the ListBox

def setList():

contactlist.sort()

# delete all elements from the select element

select.delete(0, END)

# insert each name from the list to the end of the select element

for name, phone in contactlist:

select.insert(END, name)

# refresh function - used add the end of add, update, delete functions

def refresh():

global canRoll, contactlist

canRoll = True

contactlist = dbf.selectAll()

setList()

# initialize the application

root = buildFrame()

setList()

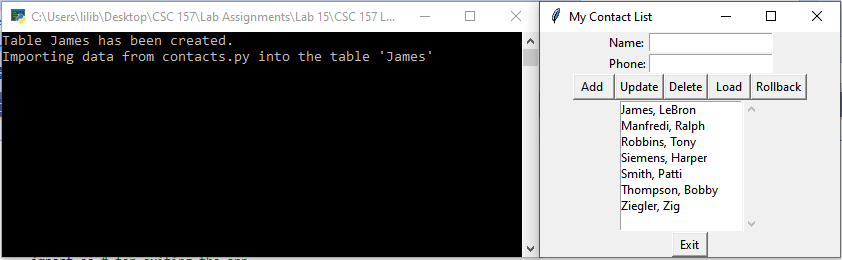
# set size of window (width x height)

root.geometry("300x225")

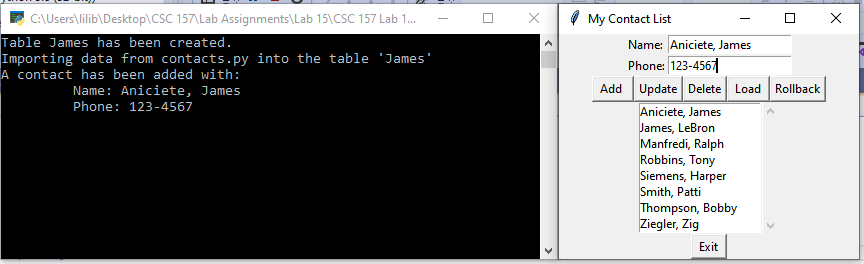
root.mainloop()

Snippet(s) of Output(s)

**Interface at runtime:**

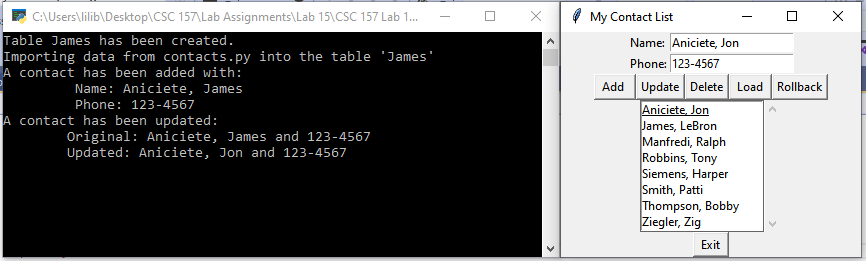


**Show your contact (name) being added via a print statement to the IDLE shell:**

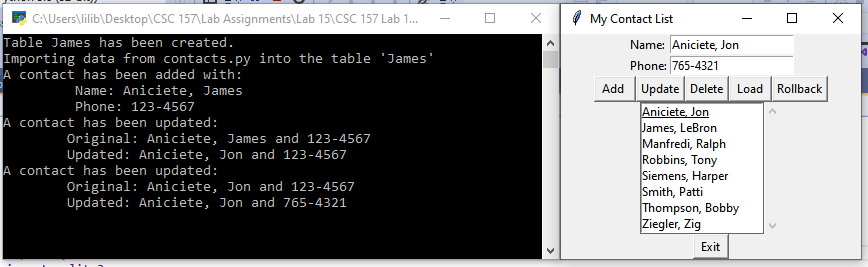


**Show your contact (name/phone number) you changed via a print statement to the IDLE shell:**

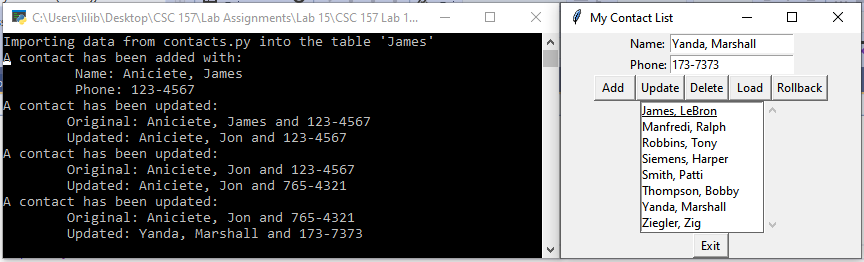
**Name changed:**



**Number changed:**

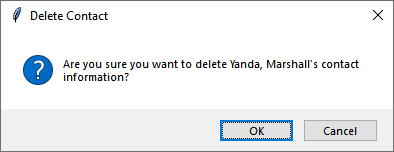


**Both Name and Number changed:**

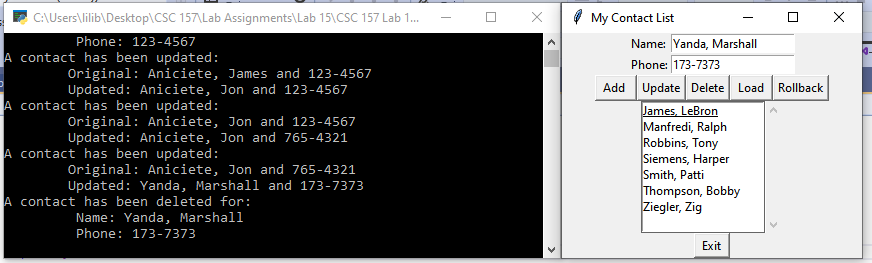


**Show the contact name being deleted via a print statement to the IDLE shell:**

**Confirmation Message Box:**

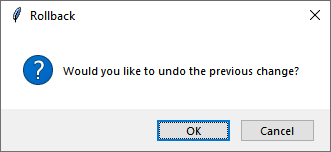


**Contact Deleted:**

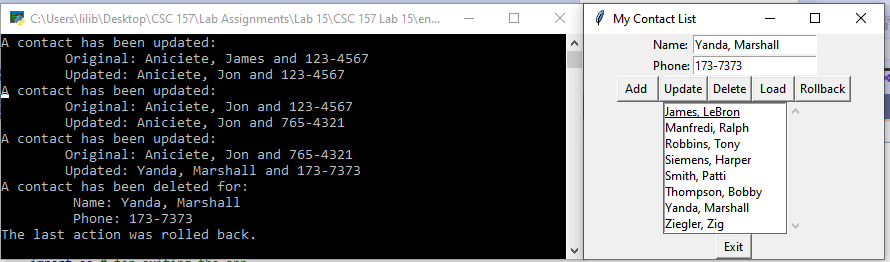


**Rollback Deleted Contact:**

**Confirmation Message Box:**



**Rollback:**



Modified Source Code

n/a

Snippet(s) of Output(s) from execution of modified Code

n/a

Excel Spreadsheet (when Calculations are involved)

n/a

Answers to Questions (Be sure to copy the questions themselves!)

**(1)** Review the program segment that is given below and answer each of these questions.

(a) What is the name of the database file that was created?

**“test.db” is the name of the database file that was created.**

(b) Which **sqlite3** method was used to connect to the named database file?

**The .connect() method was used to connect to the named database file.** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**import sqlite3**

**conn = sqlite3.connect("test.db")**

**print ("Opened database successfully")**

**(2)** Review the program segment that is given below and answer each of these questions.

(a) Considering the **CREATE TABLE** statement given in this program, how many fields in the **Sales** table were included?

**Five fields were included in the Sales table.**

(b) What is the purpose of this statement segment, as it is used below?

**IF NOT EXISTS**

**“IF NOT EXISTS” prevents the statement from creating a table if one with the same name already exists.** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**import sqlite3**

**conn = sqlite3.connect("test.db")**

**print ("database opened successfully")**

**sqlQueryStr = "DROP TABLE IF EXISTS Sales;"**

**conn.execute(sqlQueryStr)**

**print ("table successfully dropped")**

**sqlQueryStr = "CREATE TABLE IF NOT EXISTS Sales"**

**sqlQueryStr += "(ID INT PRIMARY KEY NOT NULL,"**

**sqlQueryStr += "Agent\_Name TEXT NOT NULL,"**

**sqlQueryStr += "Years\_Of\_Service INT NOT NULL,"**

**sqlQueryStr += "Address CHAR(50),"**

**sqlQueryStr += "Salary REAL);"**

**conn.execute(sqlQueryStr)**

**print ("table successfully created")**

**conn.close()**

**PROJECT Working with Tkinter ( Part 2 ) and SQLite3 database**

**(3)** Review the program segment that is given below and answer each of these questions.

(a) Considering the **INSERT** statement given in this program, how many fields from the **Sales** table were included?

**Five fields were included in the INSERT statement.**

(b) What is the purpose of this statement, as it is used below?

**conn.commit()**

**The purpose of the “conn.commit()” is to commit or save the inserted record to the Sales table to the database.** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**import sqlite3**

**conn = sqlite3.connect("test.db")**

**print ("database opened successfully")**

**sqlQueryStr = "INSERT INTO Sales"**

**sqlQueryStr += "(ID, Agent\_Name, Years\_Of\_Service,"**

**sqlQueryStr += "Address, Salary)"**

**sqlQueryStr += "VALUES (1, 'Ivy', 7, 'Gary', 3000);"**

**conn.execute(sqlQueryStr)**

**conn.commit()**

**print ("record successfully posted")**

**conn.close()**

**(4)** Review the program segment that is given below and answer each of these questions.

(a) Considering the **SELECT** statement given in this program, which field from the **Sales** table was absent in the query?

**SELECT \* FROM Sales;**

**The Years\_Of\_Service field is absent from the query.**

(b) In reference to the **Sales** table, how and why is the **for** loop used below?

**The for loop is used to print all of the listed field values for each record, using the respective index.** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**import sqlite3**

**conn = sqlite3.connect("test.db")**

**print ("database opened successfully")**

**sqlQueryStr = "SELECT ID, Agent\_Name,"**

**sqlQueryStr += "Address, Salary FROM Sales;"**

**cursor = conn.execute(sqlQueryStr)**

**for row in cursor :**

**print ("Agent ID = ", row[0])**

**print ("Agent Name = ", row[1])**

**print ("Address = ", row[2])**

**print ("Salary = ", row[3], "\n")**

**print ("operation performed successfully")**

**conn.close()**

**PROJECT Working with Tkinter ( Part 2 ) and SQLite3 database**

**(5)** Review the program segment that is given below and answer each of these questions.

(a) In the query string that follows, what is the purpose of the \* symbol? As it is used here, what is the symbol \* often

called?

**SELECT \* FROM Sales;**

**The \* symbol is used to select all of the records from the Sales table and is often called “all” when it is used in this manner.**

(b) What is the purpose of this statement, as it is used below?

**conn.commit()**

**The purpose of this statement is to commit or save the update to the Sales table.** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**import sqlite3**

**conn = sqlite3.connect("test.db")**

**print ("database opened successfully")**

**sqlQueryStr = "UPDATE Sales SET Salary = '25000'"**

**sqlQueryStr += "WHERE ID = 1;"**

**conn.execute(sqlQueryStr)**

**conn.commit()**

**print ("total rows updated =", conn.total\_changes)**

**cursor = conn.execute("SELECT \* FROM Sales;")**

**for row in cursor :**

**print ("ID = ", row[0])**

**print ("Agent Name = ", row[1])**

**print ("Years of Service = ", row[2])**

**print ("Address = ", row[3])**

**print ("Salary = ", row[4], "\n")**

**print ("operation performed successfully")**

**conn.close()**