Richard Hayes Crowley

02/18/2021

CSC\_157\_Lab\_015

**SOURCE CODE:**

**MyDatabase.py**

*import* sqlite3

def createConnection(*dbFile*):

conn = None

*try*:

conn = sqlite3.connect(dbFile)

*except* Error *as* e:

print(e)

*return* conn

def createTable(*conn*):

cur = conn.cursor()

cur.execute('''CREATE TABLE IF NOT EXISTS contacts

(id integer PRIMARY KEY, name text NOT NULL, phone text NOT NULL)''')

*# setList function*

def readTable(*conn*):

cur = conn.cursor()

*# table is an iterable of rows*

table = cur.execute('SELECT \* FROM contacts')

*return* [row *for* row *in* table]

*# load function*

def selectFromTable(*conn*, *contactId*):

cur = conn.cursor()

params = (contactId)

sql = '''SELECT FROM contacts WHERE id = ? '''

*return* cur.execute(sql, params)

*# add function*

def insertIntoTable(*conn*, *contact*: list):

params = (contact[0], contact[1])

sql = '''INSERT INTO contacts(name,phone)

VALUES (?,?)'''

cur = conn.cursor()

cur.execute(sql, params)

conn.commit()

print(f"{params} added to DB")

*# update function*

def updateTable(*conn*, *contact*: list):

params = (contact[0], contact[1], contact[2])

sql = f''' UPDATE contacts

SET name = ? ,

phone = ?

WHERE id = ?'''

cur = conn.cursor()

cur.execute(sql, params)

conn.commit()

print(f"{params} updated in DB")

*# delete function*

def deleteFromTable(*conn*, *contactId*: int):

*# remember the tuple gotcha, gotta have that comma!*

*# PS second argument in cur.execute must be tuple type*

params = (contactId,)

sql = ''' DELETE FROM contacts

WHERE id = ?'''

cur = conn.cursor()

cur.execute(sql, params)

conn.commit()

print(f"Entry with ID: {contactId} removed from DB")

**tkContacts.py**

*from* tkinter *import* \*

*from* tkinter *import* messagebox

*import* os

*import* sqlite3

*import* myDatabase *as* db

class EmptyValueException(Exception):

*pass*

def main():

global conn

*# create a database connection*

conn = db.createConnection("contacts.db")

*# create table*

*if* conn is not None:

*# create contacts table*

db.createTable(conn)

*else*:

print("Error! cannot create the database connection.")

*with* conn:

def selection():

*# print(int(select.curselection()[0]))*

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

def addContact():

*try*:

*if* not nameVar.get().strip() or not phoneVar.get().strip():

*raise* EmptyValueException

db.insertIntoTable(conn, [nameVar.get(), phoneVar.get()])

setList()

*except* EmptyValueException:

messagebox.showerror(

"Error", "Please enter a value for both name and phone number.")

def updateContact():

*try*:

contactId = contactlist[selection()][0]

db.updateTable(

conn, [nameVar.get(), phoneVar.get(), contactId])

setList()

*except* IndexError:

messagebox.showerror(

"Error", "Please select a contact to update.")

def deleteContact():

*try*:

contactId = contactlist[selection()][0]

db.deleteFromTable(conn, contactId)

setList()

*except* IndexError:

messagebox.showerror(

"Error", "Please select a contact to delete.")

def loadContact():

*try*:

contactId, name, phone = contactlist[selection()]

nameVar.set(name)

phoneVar.set(phone)

*except* IndexError:

messagebox.showerror(

"Error", "Please select a contact to load.")

def exitMenu():

*if* (messagebox.askokcancel(*title*="My Contact List", *message*="Do you want to exit, OK or Cancel") == 1):

os.\_exit(1)

def buildFrame():

global nameVar, phoneVar, select

root = Tk()

root.title("My Contact List")

frame1 = Frame(root)

frame1.pack()

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

nameVar = StringVar()

name = Entry(frame1, *textvariable*=nameVar)

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

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

phoneVar = StringVar()

phone = Entry(frame1, *textvariable*=phoneVar)

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

frame1 = Frame(root) *# add a row of buttons*

frame1.pack()

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)

btn1.pack(*side*=LEFT)

btn2.pack(*side*=LEFT)

btn3.pack(*side*=LEFT)

btn4.pack(*side*=LEFT)

frame1 = Frame(root) *# allow for selection of names*

frame1.pack()

scroll = Scrollbar(frame1, *orient*=VERTICAL)

select = Listbox(frame1, *yscrollcommand*=scroll.set, *height*=7)

scroll.config(*command*=select.yview)

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

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

frame1 = Frame(root) *# allow for selection of names*

frame1.pack()

btn6 = Button(frame1, *text*=" Exit ", *command*=exitMenu)

btn6.pack(*side*=BOTTOM)

*return* root

def setList():

global contactlist

contactlist = db.readTable(conn)

*# sort the list by name, which is second element in tuple...*

*# lambda signifies an anonymous function, so we're calling an anon function that returns the second element in the tuple to the sort function*

contactlist.sort(*key*=lambda *x*: x[1])

*# 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* id, name, phone *in* contactlist:

select.insert(END, name)

root = buildFrame()

setList()

root.mainloop()

*if* \_\_name\_\_ == '\_\_main\_\_':

main()

**OUTPUT:**

**LIST OF SCREENSHOTS:**

**1. Your interface at runtime**

**2. Initial snapshot of ENTIRE listbox of contacts at runtime**

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

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

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

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated with medium confidence