**Program source Code:**

print("Kanderi Guruteja") #Student Name  
print("Open source Programming-Python") #Course name  
print("Lab-07") #Lab Number  
print("Working with tkinter") # Name of teh lAB  
print("A20526883") #Student ID

**OUTPUT:**

contactlist = [  
 ['Guruteja,Kanderi', '8722794598'],  
 ['Jackson, Janet', '313-1352'],  
 ['James, Lebron', '457-6223'],  
 ['Madhu , Shalini', '1235567190'],  
 ['Rakesh,Ramraja', '1235567890'],  
 ['Sai, Kumar', '9848612123'],  
 ['Thompson, Bobby', '365-2622'],  
 ['yugesh', '9876543210'],  
 ['yugesh123', '1235567123'],  
 ['guru', '1234567890'],  
 ['sai', '1234567890'],  
 ['rakesh', '1234567890'],  
 ['preethi', '1234567890'],  
 ['kumar', '1234567890'],  
 ['divya', '1234567890'],  
 ['lakshmi', '1234567890'],  
 ['priya', '1234567890'],  
 ['manjula', '9848678982'],  
]

import sqlite3  
from pathlib import Path  
  
# Function to create a table  
def create\_table():  
 try:  
 db\_file = Path('contacts.db')  
 conn = sqlite3.connect(db\_file)  
 cursor = conn.cursor()  
  
 # Create the table with appropriate column names and data types  
 cursor.execute('''CREATE TABLE IF NOT EXISTS contacts  
 (id INTEGER PRIMARY KEY AUTOINCREMENT,  
 name TEXT,  
 phone TEXT)''')  
  
 conn.commit()  
 conn.close()  
 except Exception as e:  
 print("Error creating table:", e)  
  
# Function to update a table  
def update\_contact(id, name, phone):  
 try:  
 db\_file = Path('contacts.db')  
 conn = sqlite3.connect(db\_file)  
 cursor = conn.cursor()  
  
 # Execute the update query  
 cursor.execute('UPDATE contacts SET name=?, phone=? WHERE id=?', (name, phone, id))  
  
 conn.commit()  
 conn.close()  
 except Exception as e:  
 print("Error updating contact:", e)  
  
# Function to delete from a table  
def delete\_contact(id):  
 try:  
 db\_file = Path('contacts.db')  
 conn = sqlite3.connect(db\_file)  
 cursor = conn.cursor()  
  
 # Execute the delete query  
 cursor.execute('DELETE FROM contacts WHERE id=?', (id,))  
  
 conn.commit()  
 conn.close()  
 except Exception as e:  
 print("Error deleting contact:", e)  
  
# Function to insert into a table  
def insert\_contact(name, phone):  
 try:  
 db\_file = Path('contacts.db')  
 conn = sqlite3.connect(db\_file)  
 cursor = conn.cursor()  
  
 # Execute the insert query  
 cursor.execute('INSERT INTO contacts (name, phone) VALUES (?, ?)', (name, phone))  
  
 conn.commit()  
 conn.close()  
 except Exception as e:  
 print("Error inserting contact:", e)  
  
# Function to read (load in) record(s) from a table  
def read\_contacts():  
 try:  
 db\_file = Path('contacts.db')  
 conn = sqlite3.connect(db\_file)  
 cursor = conn.cursor()  
  
 # Execute the select query to retrieve all records  
 cursor.execute('SELECT \* FROM contacts')  
  
 contacts = cursor.fetchall() # Fetch all records  
  
 conn.close()  
 return contacts  
 except Exception as e:  
 print("Error reading contacts:", e)  
  
# Call the create\_table() function to ensure the table is created  
create\_table()

from LAB\_07\_PYTHON.contacts import contactlist  
  
print("Kanderi Guruteja") # Student Name  
print("Open source Programming-Python") # Course name  
print("Lab-07") # Lab Number  
print("Working with tkinter") # Name of the Lab  
print("A20526883") # Student ID  
  
# Import required modules  
from tkinter import \*  
from tkinter import messagebox  
import pickle  
import os  
import re  
import time  
from datetime import date  
from myDatabasefile import \*  
def read\_contacts\_from\_file():  
 # Check if the contacts.py file exists  
 if os.path.exists('contacts.py'):  
 # contacts.py file for loading the contacts list  
 with open('contacts.py', 'r') as f:  
 exec(f.read(), globals())  
 return contactlist  
 else:  
 return []  
  
contactlist1 = read\_contacts\_from\_file()  
  
def selection(): # Function to get the selected index from the listbox  
 selected\_indices1 = select.curselection()  
 if selected\_indices1:  
 return int(selected\_indices1[0])  
 else:  
 return -1  
  
def addContact(): # Function to add a new contact  
 nameofcontact = nameVariable1.get()  
 phoneofcontact = phone\_Variables1.get()  
  
 if nameofcontact and phoneofcontact:  
 for contact in contactlist1:  
 if contact[0] == nameofcontact:  
 messagebox.showerror("Error", "Contact with the same name already exists.")  
 return  
  
 if not re.match("^[a-zA-Z0-9, -]+$", nameofcontact):  
 messagebox.showerror("Error", "Invalid characters in the name. Please use only letters, numbers, hyphen, comma, or space.")  
 elif not re.match("^[0-9-]+$", phoneofcontact) or len(phoneofcontact) < 10:  
 messagebox.showerror("Error", "Invalid phone number. Please enter at least 10 digits and use only numbers and hyphen.")  
 else:  
 contactlist1.append([nameofcontact, phoneofcontact])  
 setList()  
 saveContacts()  
 messagebox.showinfo("Success", "Contact added successfully.")  
 else:  
 messagebox.showerror("Error", "Please enter a name and phone number.")  
  
def updateContact(): # Function to update a contact  
 selected\_index = selection()  
 if selected\_index >= 0:  
 name = nameVariable1.get()  
 phone = phone\_Variables1.get()  
 if name and phone:  
 if not re.match("^[a-zA-Z0-9,-]+$", name):  
 messagebox.showerror("Error", "Invalid characters in the name. Please use only letters, numbers, hyphen, or comma.")  
 else:  
 contactlist1[selected\_index] = [name, phone]  
 setList()  
 saveContacts()  
 messagebox.showinfo("Success", "Contact updated successfully.")  
 else:  
 messagebox.showerror("Error", "Please select a contact and load it before updating.")  
 else:  
 messagebox.showerror("Error", "No contact selected for update.")  
  
def deleteContact(): # Function to delete a contact  
 selected\_index = selection()  
 if selected\_index >= 0:  
 del contactlist1[selected\_index]  
 setList()  
 saveContacts()  
 messagebox.showinfo("Success", "Contact deleted successfully.")  
 else:  
 messagebox.showerror("Error", "No contact selected.")  
  
def loadContact(): # Function to load a contact  
 selected\_index = selection()  
 if selected\_index >= 0:  
 name, phone = contactlist1[selected\_index]  
 nameVariable1.set(name)  
 phone\_Variables1.set(phone)  
  
def buildFrame(): # Function to build the GUI frame  
 global nameVariable1, phone\_Variables1, select  
 root = Tk()  
 root.title("My Contact List")  
  
 frame1 = Frame(root)  
 frame1.grid(row=0, column=0)  
 Label(frame1, text="Name:").grid(row=0, column=0, sticky=W)  
 nameVariable1 = StringVar()  
 name = Entry(frame1, textvariable=nameVariable1)  
 name.grid(row=0, column=1, sticky=W)  
 Label(frame1, text="Phone:").grid(row=1, column=0, sticky=W)  
 phone\_Variables1 = StringVar()  
 phone = Entry(frame1, textvariable=phone\_Variables1)  
 phone.grid(row=1, column=1, sticky=W)  
  
 frame2 = Frame(root)  
 frame2.grid(row=1, column=0)  
 btn1 = Button(frame2, text=" Add ", command=addContact)  
 btn1.grid(row=0, column=0, padx=5)  
 btn2 = Button(frame2, text="Update", command=updateContact)  
 btn2.grid(row=0, column=1, padx=5)  
 btn3 = Button(frame2, text="Delete", command=deleteContact)  
 btn3.grid(row=0, column=2, padx=5)  
 btn4 = Button(frame2, text=" Load ", command=loadContact)  
 btn4.grid(row=0, column=3, padx=5)  
 btn5 = Button(frame2, text="Save", command=saveContacts)  
 btn5.grid(row=0, column=4, padx=5)  
  
 frame3 = Frame(root)  
 frame3.grid(row=2, column=0, pady=10)  
 scroll = Scrollbar(frame3, orient=VERTICAL)  
 select = Listbox(frame3, yscrollcommand=scroll.set, height=10)  
 scroll.config(command=select.yview)  
 scroll.pack(side=RIGHT, fill=Y)  
 select.pack(side=LEFT, fill=BOTH)  
  
 btn6 = Button(root, text="Exit", command=exitProgram)  
 btn6.grid(row=3, column=0, pady=10)  
  
 return root  
  
def setList(): # Function to update the listbox with the contact names  
 select.delete(0, END)  
 for contact in contactlist1:  
 select.insert(END, contact[0])  
  
def saveContacts():  
 with open('contacts.py', 'w') as f:  
 f.write("contactlist = [\n")  
 for contact in contactlist1:  
 f.write(f" ['{contact[0]}', '{contact[1]}'],\n")  
 f.write("]\n")  
 current\_time = time.strftime("%H:%M:%S")  
 current\_date = date.today().strftime("%Y-%m-%d")  
 messagebox.showinfo("Save", f"Contacts saved successfully.\nDate: {current\_date}\nTime: {current\_time}")  
  
def exitProgram():  
 if messagebox.askokcancel(title="Exit", message="Are you sure you want to exit?"):  
 os.\_exit(1)  
  
root = buildFrame()  
setList()  
root.protocol("WM\_DELETE\_WINDOW", exitProgram)  
root.mainloop()