# Project overview: Contact Book project in Python

|  |  |
| --- | --- |
| **Project Name:** | **Contact Book project in Python** |
| **Abstract:** | This is a simple app that can be used to handle contacts related to addresses or phones. |
| **Language/Technologies Used:** | Python, [Tkinter,](https://docs.python.org/3/library/tkinter.html) [Sqlite3](https://docs.python.org/3/library/sqlite3.html) |
| **IDE** | VSCode(Recommended) |
| **Database** | **SQLite3** |
| **Python version (Recommended):** | 3.x |
| **Type/Category:** | All Python learners |

# Features

* Adding contacts
* Updating contacts
* Persisting the contacts with the sqllite3 database
* Delete a contact

# Complete code for Contact Book project in Python

We will just need a single ***main.py*** file and the code for the same is written down here, the code is well commented so as to assist you with better learning. The default username and password are “root” and “root”, you can always change that later in the code.

# Source Code

import sqlite3

from tkinter import \*

from tkinter import ttk, messagebox

class ContactManager:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("ContactBook Management")

self.root.attributes('-fullscreen', True)

title = Label(self.root, text="ContactBook Management", font=("Comic Sans MS", 20), bd=8, bg='black', fg='white')

title.pack(side=TOP, fill=X)

self.firstname = StringVar()

self.lastname = StringVar()

self.mobile = StringVar()

self.addr = StringVar()

self.pin = StringVar()

Detail\_F = Frame(self.root, bd=4, relief=RIDGE, bg='white')

Detail\_F.place(x=10, y=150, width=390, height=260)

lbl\_name = Label(Detail\_F, text="First Name", font=("Comic Sans MS", 12))

lbl\_name.grid(row=1, column=0, pady=10, padx=20, sticky="w")

txt\_name = Entry(Detail\_F, font=("Comic Sans MS", 10), bd=3, textvariable=self.firstname)

txt\_name.grid(row=1, column=1, pady=10, sticky="w")

lbl\_mob = Label(Detail\_F, text="Last Name", font=("Comic Sans MS", 12))

lbl\_mob.grid(row=2, column=0, pady=10, padx=20, sticky="w")

txt\_mob = Entry(Detail\_F, font=("Comic Sans MS", 10), bd=3, textvariable=self.lastname)

txt\_mob.grid(row=2, column=1, pady=10, sticky="w")

lbl\_aa = Label(Detail\_F, text="Mobile No.", font=("Comic Sans MS", 12))

lbl\_aa.grid(row=3, column=0, pady=10, padx=20, sticky="w")

txt\_aa = Entry(Detail\_F, font=("Comic Sans MS", 10), bd=3, textvariable=self.mobile)

txt\_aa.grid(row=3, column=1, pady=10, sticky="w")

lbl\_add = Label(Detail\_F, text="Address", font=("Comic Sans MS", 12))

lbl\_add.grid(row=4, column=0, pady=10, padx=20, sticky="w")

txt\_add = Entry(Detail\_F, font=("Comic Sans MS", 10), bd=3, textvariable=self.addr)

txt\_add.grid(row=4, column=1, pady=10, sticky="w")

lbl\_pin = Label(Detail\_F, text="PinCode", font=("Comic Sans MS", 12))

lbl\_pin.grid(row=5, column=0, pady=10, padx=20, sticky="w")

txt\_pin = Entry(Detail\_F, font=("Comic Sans MS", 10), bd=3, textvariable=self.pin)

txt\_pin.grid(row=5, column=1, pady=10, sticky="w")

recordFrame = Frame(self.root, bd=5, relief=RIDGE)

recordFrame.place(x=450, y=160, width=550, height=260)

yscroll = Scrollbar(recordFrame, orient=VERTICAL)

self.contact\_table = ttk.Treeview(recordFrame, columns=("firstname", "lastname", "mobile", "address", "pin"), yscrollcommand=yscroll.set)

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

yscroll.config(command=self.contact\_table.yview)

self.contact\_table.heading("firstname", text="First Name")

self.contact\_table.heading("lastname", text="Last Name")

self.contact\_table.heading("mobile", text="Mobile No.")

self.contact\_table.heading("address", text="Address")

self.contact\_table.heading("pin", text="PinCode")

self.contact\_table['show'] = 'headings'

self.contact\_table.column("firstname", width=100)

self.contact\_table.column("lastname", width=100)

self.contact\_table.column("mobile", width=100)

self.contact\_table.column("address", width=100)

self.contact\_table.column("pin", width=110)

self.contact\_table.pack(fill=BOTH, expand=1)

self.fetch\_data()

self.contact\_table.bind("<ButtonRelease-1>", self.get\_cursor)

btnFrame = Frame(self.root, bd=5, relief=RIDGE)

btnFrame.place(x=250, y=450, width=600, height=60)

btn1 = Button(btnFrame, text='Add record', font='arial 12 bold', bg='black', fg='white', width=9, command=self.addrecord)

btn1.grid(row=0, column=0, padx=10, pady=10)

btn2 = Button(btnFrame, text='Update', font='arial 12 bold', bg='black', fg='white', width=9, command=self.update)

btn2.grid(row=0, column=1, padx=8, pady=10)

btn3 = Button(btnFrame, text='Delete', font='arial 12 bold', bg='black', fg='white', width=9, command=self.delete)

btn3.grid(row=0, column=2, padx=8, pady=10)

btn4 = Button(btnFrame, text='Reset', font='arial 12 bold', bg='black', fg='white', width=9, command=self.reset)

btn4.grid(row=0, column=3, padx=8, pady=10)

btn5 = Button(btnFrame, text='Exit', font='arial 12 bold', bg='black', fg='white', width=9, command=self.exit\_program)

btn5.grid(row=0, column=4, padx=8, pady=10)

self.search\_frame = Frame(self.root, bd=4, relief=RIDGE)

self.search\_frame.place(x=450, y=90, width=390, height=50)

self.search\_label = Label(self.search\_frame, text="Search:", font=("Comic Sans MS", 12))

self.search\_label.grid(row=0, column=0, padx=20, pady=10, sticky="w")

self.search\_entry = Entry(self.search\_frame, font=("Comic Sans MS", 10), bd=3)

self.search\_entry.grid(row=0, column=1, pady=10, sticky="w")

self.search\_button = Button(self.search\_frame, text="Search", font='arial 12 bold', bg='black', fg='white', width=9, command=self.search\_record)

self.search\_button.grid(row=0, column=2, padx=8, pady=10)

def addrecord(self):

if self.firstname.get() == '' or self.lastname.get() == '' or self.mobile.get() == '' or self.addr.get() == '' or self.pin.get() == '':

messagebox.showerror('Error', 'Please enter details')

else:

con = sqlite3.connect('contactbook.db')

cur = con.cursor()

cur.execute("SELECT \* FROM contact")

rows = cur.fetchall()

for row in rows:

if row[2] == self.mobile.get():

messagebox.showerror('Error', 'Duplicates not allowed')

return

cur.execute("INSERT INTO contact (firstname, lastname, mobile, addr, pin) VALUES (?, ?, ?, ?, ?)",

(self.firstname.get(), self.lastname.get(), self.mobile.get(), self.addr.get(), self.pin.get()))

con.commit()

con.close()

messagebox.showinfo('Success', 'Record added successfully.')

self.fetch\_data()

self.reset()

def fetch\_data(self):

con = sqlite3.connect('contactbook.db')

cur = con.cursor()

cur.execute("SELECT \* FROM contact")

rows = cur.fetchall()

if len(rows) != 0:

self.contact\_table.delete(\*self.contact\_table.get\_children())

for row in rows:

self.contact\_table.insert('', END, values=row)

con.commit()

con.close()

def update(self):

if self.mobile.get() == '':

messagebox.showerror('Error', 'Please select a record to update.')

else:

con = sqlite3.connect('contactbook.db')

cur = con.cursor()

cur.execute("UPDATE contact SET firstname=?, lastname=?, addr=?, pin=? WHERE mobile=?",

(self.firstname.get(), self.lastname.get(), self.addr.get(), self.pin.get(), self.mobile.get()))

con.commit()

con.close()

messagebox.showinfo('Success', 'Record updated successfully.')

self.fetch\_data()

self.reset()

def delete(self):

if self.mobile.get() == '':

messagebox.showerror('Error', 'Please select a record to delete.')

else:

con = sqlite3.connect('contactbook.db')

cur = con.cursor()

cur.execute("DELETE FROM contact WHERE mobile=?", (self.mobile.get(),))

con.commit()

con.close()

messagebox.showinfo('Success', 'Record deleted successfully.')

self.fetch\_data()

self.reset()

def reset(self):

self.firstname.set('')

self.lastname.set('')

self.mobile.set('')

self.addr.set('')

self.pin.set('')

def get\_cursor(self, ev):

cursor\_row = self.contact\_table.focus()

content = self.contact\_table.item(cursor\_row)

row = content['values']

self.firstname.set(row[0])

self.lastname.set(row[1])

self.mobile.set(row[2])

self.addr.set(row[3])

self.pin.set(row[4])

def search\_record(self):

con = sqlite3.connect('contactbook.db')

cur = con.cursor()

cur.execute("SELECT \* FROM contact WHERE firstname LIKE ? OR lastname LIKE ?",

('%' + self.search\_entry.get() + '%', '%' + self.search\_entry.get() + '%'))

rows = cur.fetchall()

if len(rows) != 0:

self.contact\_table.delete(\*self.contact\_table.get\_children())

for row in rows:

self.contact\_table.insert('', END, values=row)

else:

messagebox.showinfo('Info', 'No records found.')

con.commit()

con.close()

def exit\_program(self):

self.root.destroy()

class Login:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Contact Book Management System")

self.root.geometry("300x150")

self.username = StringVar()

self.password = StringVar()

Label(self.root, text="Username:").grid(row=0, column=0, padx=10, pady=10)

Entry(self.root, textvariable=self.username).grid(row=0, column=1, padx=10, pady=10)

Label(self.root, text="Password:").grid(row=1, column=0, padx=10, pady=10)

Entry(self.root, textvariable=self.password, show="\*").grid(row=1, column=1, padx=10, pady=10)

Button(self.root, text="Login", command=self.login).grid(row=2, column=1, padx=10, pady=10)

def login(self):

if self.username.get() == "root" and self.password.get() == "root":

self.root.destroy()

nroot = Tk()

ContactManager(nroot)

else:

messagebox.showerror("Error", "Invalid username or password")

con = sqlite3.connect('contactbook.db')

cur = con.cursor()

cur.execute('CREATE TABLE IF NOT EXISTS contact (firstname TEXT, lastname TEXT, mobile TEXT PRIMARY KEY, addr TEXT, pin TEXT)')

cur.close()

con.close()

root = Tk()

obj = Login(root)

root.mainloop()

**Output for Contact Book project in Python:**

**Image output:**

**Login Screen:**



