Python Contact Book Project Details

Introduction:

Contact Book is a GUI based project using Tkinter and a message module used for storing information about some person like name and contact number. In this project we have some functionality like add, edit, delete, view, and reset contacts.

We can add new contacts by clicking on the add button and filling in the information of the new contact. We can update or edit existing contacts by selecting and clicking on the edit button. Similarly, we can delete and view the contact. Like this, we can store data.

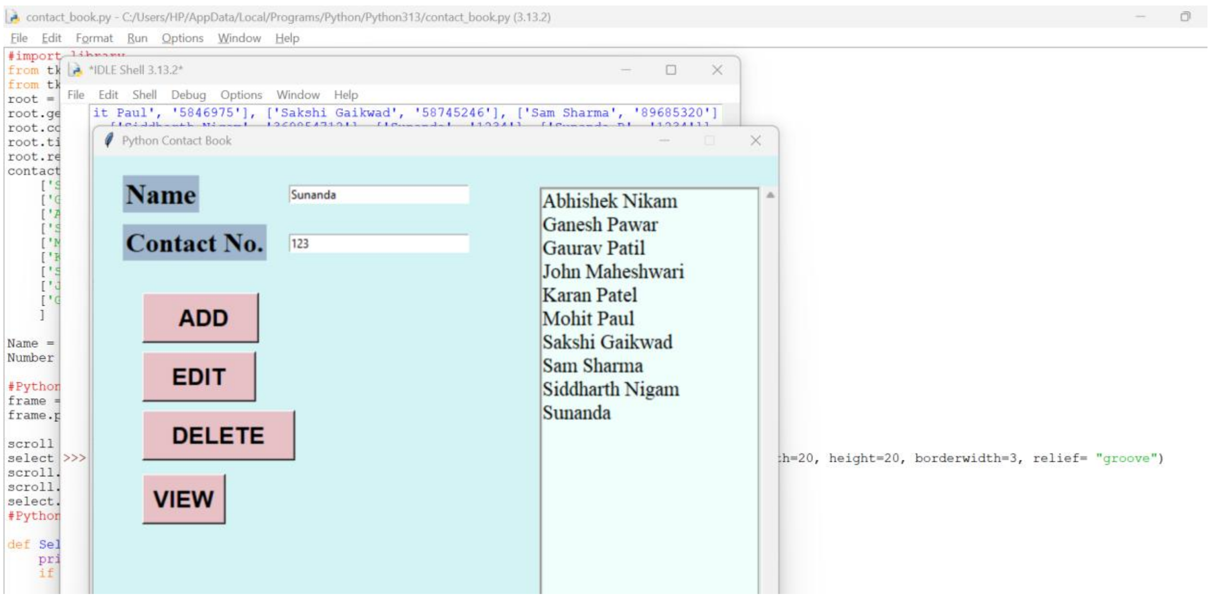
Source Code:

```
#import library

from tkinter import *

from tkinter import messagebox#PythonGeeks address book - Initialize window

root = Tk()

root.geometry('700x550')

root.config(bg = '#d3f3f5')

root.title('Python Contact Book')

root.resizable(0,0)

contactlist = [

    ['Siddharth Nigam','369854712'],

    ['Gaurav Patil', '521155222'],

    ['Abhishek Nikam', '78945614'],
    ['Sakshi Gaikwad', '58745246'],
    ['Mohit Paul', '5846975'],
    ['Karan Patel', '5647892'],
    ['Sam Sharma', '89685320'],
    ['John Maheshwari', '98564785'],
    ['Ganesh Pawar','85967412']
    ]

Name = StringVar()
Number = StringVar()

#Python - create frame
frame = Frame(root)
frame.pack(side = RIGHT)

scroll = Scrollbar(frame, orient=VERTICAL)
select = Listbox(frame, yscrollcommand=scroll.set, font=('Times new roman'
,16), bg="#f0fffc", width=20, height=20, borderwidth=3, relief= "groove")
scroll.config (command=select.yview)
scroll.pack(side=RIGHT, fill=Y)
select.pack(side=LEFT,  fill=BOTH, expand=1)
#Python - function to get select value

def Selected():
    print("hello",len(select.curselection()))
    if len(select.curselection())==0:
        messagebox.showerror("Error", "Please Select the Name")
    else:
        return int(select.curselection()[0])


def AddContact():
    if Name.get()!="" and Number.get()!="":
        contactlist.append([Name.get() ,Number.get()])
        print(contactlist)
        Select_set()
        EntryReset()
        messagebox.showinfo("Confirmation", "Successfully Add New
Contact")


    else:
        messagebox.showerror("Error","Please fill the information")
#Python -fun to add new contact
def AddContact():
    if Name.get()!="" and Number.get()!="":
        contactlist.append([Name.get() ,Number.get()])
        print(contactlist)
        Select_set()
        EntryReset()
        messagebox.showinfo("Confirmation", "Successfully Add New
Contact")
```

```
    else:
        messagebox.showerror("Error","Please fill the information")


def UpdateDetail():
    if Name.get() and Number.get():
        contactlist[Selected()] = [Name.get(), Number.get()]


        messagebox.showinfo("Confirmation", "Successfully Update Contact")
        EntryReset()
        Select_set()



    elif not(Name.get()) and not(Number.get()) and
not(len(select.curselection())==0):
        messagebox.showerror("Error", "Please fill the information")


    else:
        if len(select.curselection())==0:
            messagebox.showerror("Error", "Please Select the Name and \n press
Load button")
        else:
            message1 = """To Load the all information of \n
                          selected row press Load button\n.
                          """
            messagebox.showerror("Error", message1)

def Delete_Entry():

    if len(select.curselection())!=0:

        result=messagebox.askyesno('Confirmation','You Want to Delete
Contact\n Which you selected')

        if result==True:

            del contactlist[Selected()]

            Select_set()

    else:

        messagebox.showerror("Error", 'Please select the Contact')


def VIEW():

    NAME, PHONE = contactlist[Selected()]

    Name.set(NAME)

    Number.set(PHONE)

# exit game window

def EXIT():

    root.destroy()


def Select_set() :

    contactlist.sort()

    select.delete(0,END)

    for name,phone in contactlist :

        select.insert (END, name)

Select_set()
```

```
#Python - define buttons labels and entry widget

Label(root, text = 'Name', font=("Times new roman",22,"bold"), bg =
'SlateGray3').place(x= 30, y=20)

Entry(root, textvariable = Name, width=30).place(x= 200, y=30)

Label(root, text = 'Contact No.', font=("Times new roman",20,"bold"),bg =
'SlateGray3').place(x= 30, y=70)

Entry(root, textvariable = Number, width=30).place(x= 200, y=80)



Button(root,text=" ADD", font='Helvetica 18 bold',bg='#e8c1c7', command =
AddContact, padx=20). place(x= 50, y=140)

Button(root,text="EDIT", font='Helvetica 18 bold',bg='#e8c1c7',command =
UpdateDetail, padx=20).place(x= 50, y=200)

Button(root,text="DELETE", font='Helvetica 18
bold',bg='#e8c1c7',command = Delete_Entry, padx=20).place(x= 50, y=260)

Button(root,text="VIEW", font='Helvetica 18 bold',bg='#e8c1c7', command
= VIEW).place(x= 50, y=325)

Button(root,text="RESET", font='Helvetica 18 bold',bg='#e8c1c7', command
= EntryReset).place(x= 50, y=390)

Button(root,text="EXIT", font='Helvetica 24 bold',bg='tomato', command =
EXIT).place(x= 250, y=470)



root.mainloop()
```

OUTPUT:


contact_book.py - C:/Users/HP/AppData/Local/Programs/Python/Python313/contact_book.py (3.13.2)
File Edit Format Run Options Window Help
IDLE Shell 3.13.2
File Edit Shell Debug Options Window Help
Python Contact Book
Name
Contact No.
ADD
EDIT
DELETE
VIEW

```
it Paul', '5846975'], ['Sakshi Gaikwad', '58745246'], ['Sam Sharma', '89685320']
>>>
```

```
h=20, height=20, borderwidth=3, relief= "groove")
```

Name: Sunanda
Contact No.: 123

```
Abhishek Nikam
Ganesh Pawar
Gaurav Patil
John Maheshwari
Karan Patel
Mohit Paul
Sakshi Gaikwad
Sam Sharma
Siddharth Nigam
Sunanda
```

Applications of Contact book project:

A Python contact book can be surprisingly versatile! Here are some practical places and scenarios where it can be used:

1. **Personal Use**: Keep track of friends, family, and service contacts—especially if you want a lightweight, offline alternative to cloud-based apps.
2. **Small Businesses**: Store customer or client contact details without needing a full CRM system. It's great for freelancers or local shops.
3. **Educational Projects**: Perfect for students learning Python, file handling, databases, or GUI development. It's a classic beginner-to-intermediate project.
4. **Command-Line Tools**: Developers and sysadmins can use it to manage team or vendor contacts directly from the terminal.
5. **Internal Tools**: Companies can use it as a quick internal directory for teams, especially when integrated with a database or web interface.
6. **Embedded in Larger Apps**: It can be a module inside a bigger app—like a school management system or event planner.

Conclusion:

The Python contact book project is a practical and educational tool that combines essential programming concepts like data structures, file handling, and user interaction. It helps learners understand how to build real-world applications by implementing features such as adding, updating, deleting, and searching contacts. With options for data persistence and interface design, it lays the groundwork for more advanced software development.