GUI Programming (Graphical User Interface or Windows Programming)

What is GUI Application?

A graphical user interface (GUI) is a digital interface in which a user interacts with graphical components such as icons, buttons, and menus. In a GUI, the visuals displayed in the user interface convey information relevant to the user, as well as actions that they can take.

GUI applications are developed using different libraries

1. Tkinter
2. Wxpython
3. PyQT
4. PyGUI
5. Kivy

Tkinter is default module which comes with python software.
Tkinter provides set of classes and each class represents one gui component or widget.
Button, Label, Checkbox, RadioButton,…

Creating Window
Window is root widget for developing any GUI Application.
Tk() function is used for creating window object.

Example:
```
import tkinter

w=tkinter.Tk()
```

Window behavior and styles
title() : It is a method of window object, which is used to read title of window and change title of window

Example:
```
import tkinter

w=tkinter.Tk()
```

w.title("Login")

Size and Location
In Tk, a window's position and size on the screen are known as its *geometry*. A full geometry specification looks like this: *width*x*height*±*x*±*y*.

Example:
```
import tkinter

w=tkinter.Tk()
w.title("Login")
w.geometry("200x300+10+100")
```

The *x* (horizontal position) is specified with a leading plus or minus, so +25 means the left edge of the window should be 25 pixels from the left edge of the screen, while -50 means the right edge of the window should be 50 pixels from the right edge of the screen. Similarly, a *y* (vertical) position of +10 means the top edge of the window should be ten pixels below the top of the screen, while -100 means the bottom edge of the window should be 100 pixels above the bottom of the screen.

Resizing Behavior
By default, toplevel windows, including the root window, can be resized by users. However, sometimes you may want to prevent users from resizing the window. You can do this via the resizable method. Its first parameter controls whether users can change the width, and the second if they can change the height.

window.resizable(False,False)

Example
```
import tkinter

w=tkinter.Tk()
w.title("Login")
w.geometry("300x300+10+100")
w.resizable(width=False,height=False)
```

Intercepting the Close Button
Most windows have a close button in their title bar. By default, Tk will destroy the window if users click on that button. You can, however, provide

a callback that will be run instead. A common use is to prompt the user to save an open file if modifications have been made.

window.protocol("WM_DELETE_WINDOW", *callback*)

Example:

```
import tkinter
def main():
    w=tkinter.Tk()
    w.title("Login")
    w.geometry("500x300+150+50")
    w.resizable(False,False)
    def fun1():
        print("Message")
        w.destroy()
    w.protocol("WM_DELETE_WINDOW",fun1)
main()
```

Transparency

Windows can be made partially transparent by specifying an alpha channel, ranging from 0.0 (fully transparent) to 1.0 (fully opaque).

***window*.attributes("-alpha", 0.5)**

Example

```
import tkinter

def main():
    w=tkinter.Tk()
    w.title("Login")
    w.geometry("500x300+150+50")
    w.resizable(False,False)
    w.attributes("-alpha", 0.5)
main()
```

Full Screen

You can make a window expand to take up the full screen:

window.attributes("-fullscreen", 1)

```
import tkinter

def main():
    w=tkinter.Tk()
    w.title("Login")
    w.geometry("500x300+150+50")
    w.attributes("-fullscreen", 1)
main()
```

Iconifying and Withdrawing
On most systems, you can temporarily remove the window from the screen by iconifying it. In Tk, whether or not a window is iconified is referred to as the window's *state*.

```
import tkinter


w=tkinter.Tk()
w.title("Login")
w.geometry("500x300+150+50")
thestate = w.state()
w.state('normal')
#  w.iconify()
```

Background of window

```
import tkinter

w=tkinter.Tk()
w.title("Login")
w.geometry("500x300+150+50")
w['bg']="blue"
```

Basic Widgets of Tkinter

Frame

A frame is a widget that displays as a simple rectangle. Frames help to organize your user interface, often both visually and at the coding level. Frames often act as master widgets for a geometry manager like grid, which manages the slave widgets contained within the frame.

Frames are created using the ttk.Frame class:

```
frame = ttk.Frame(parent)
```

Padding
The padding configuration option is used to request extra space around the inside of the widget. If you're putting other widgets inside the frame, there will be a margin all the way around. You can specify the same padding for all sides, different horizontal and vertical padding, or padding for each side separately.

```
f['padding'] = 5          # 5 pixels on all sides
f['padding'] = (5,10)     # 5 on left and right, 10 on top and bottom
f['padding'] = (5,7,10,12) # left: 5, top: 7, right: 10, bottom: 12
```

Borders
You can display a border around a frame widget to visually separate it from its surroundings

```
frame['borderwidth'] = 2
frame['relief'] = 'sunken'
```

set the borderwidth configuration option (which defaults to 0, i.e., no border) and the relief option, which specifies the visual appearance of the border. This can be one of: flat (default), raised, sunken, solid, ridge, or groove.

Example:

```
import tkinter
w=tkinter.Tk()
w.geometry("300x300")
w['bg']="cyan"
frame=tkinter.Frame(w,width=200,height=150,bg="yellow")
frame['borderwidth'] = 5
frame['relief'] = 'solid'
```

frame.place(x=50,y=10)

Label
A label is a widget that displays text or images, typically that users will just view but not otherwise interact with. Labels are used to identify controls or other parts of the user interface, provide textual feedback or results, etc.

label = ttk.Label(*parent*, text='Full name:')

```
import tkinter

w=tkinter.Tk()
w.geometry("400x500")
w['bg']="cyan"
l1=tkinter.Label(w,text="UserName",font=("Arial",15),bg="yellow",fg="red")
l1.place(x=10,y=100)
l2=tkinter.Label(w,text="Password",font=("Arial",15),bg="yellow",fg="red")
l2.place(x=10,y=150)
img=tkinter.PhotoImage(file='bird.gif')
l3=tkinter.Label(w,image=img,width=300,height=200)
l3.place(x=10,y=200)
```

Entry
An **entry** widget presents users with a single-line text field where they can type in a string value.
Syntax:
Entry(root,properties)

Example:

```
import tkinter

w=tkinter.Tk()
w.geometry("400x400")
l1=tkinter.Label(w,text="UserName",font=("Arial",15))
l2=tkinter.Label(w,text="Password",font=("Arial",15))
e1=tkinter.Entry(w,width=10,font=("Arial",15))
e2=tkinter.Entry(w,width=10,font=("Arial",15),show="*")
l1.place(x=50,y=50)
l2.place(x=50,y=100)
e1.place(x=200,y=50)
```

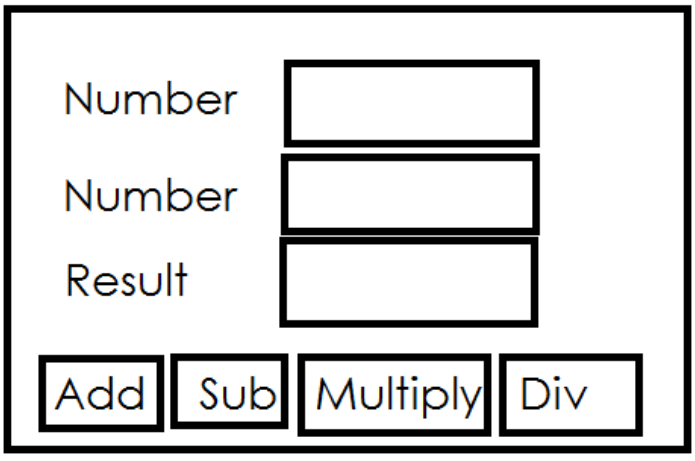
```
e2.place(x=200,y=100)
```

Button

A button, unlike a frame or label, is very much there to interact with. Users press a button to perform an action. Like labels, they can display text or images but accept additional options to change their behavior.

Syntax:Button(root,text,command)

```
import tkinter
w=tkinter.Tk()
w.geometry("300x300")
b1=tkinter.Button(w,text="Ok",width=10,height=2,bg="cyan",fg="red",font=("
arial",14))
image=tkinter.PhotoImage(file='bird.gif')
myimg=image.subsample(7,7)
b2=tkinter.Button(w,image=myimg)
b2.image=myimg
b1.place(x=50,y=50)
b2.place(x=50,y=120)
```




```
import tkinter as gui

w=gui.Tk()
w.geometry("200x200")
l1=gui.Label(w,text="Number",font=("Arial",14))
l2=gui.Label(w,text="Number",font=("Arial",14))
```

```
l3=gui.Label(w,text="Result",font=("Arial",14))

e1=gui.Entry(w,width=10,font=("Arial",14))
e2=gui.Entry(w,width=10,font=("Arial",14))
e3=gui.Entry(w,width=10,font=("Arial",14))
def add():
    n1=eval(e1.get())
    n2=eval(e2.get())
    n3=n1+n2
    e3.delete(0,gui.END)
    e3.insert(0,str(n3))

def sub():
    n1=eval(e1.get())
    n2=eval(e2.get())
    n3=n1-n2
    e3.delete(0,gui.END)
    e3.insert(0,str(n3))
def mul():
    n1=eval(e1.get())
    n2=eval(e2.get())
    n3=n1*n2
    e3.delete(0,gui.END)
    e3.insert(0,str(n3))
def div():
    n1=eval(e1.get())
    n2=eval(e2.get())
    n3=n1/n2
    e3.delete(0,gui.END)
    e3.insert(0,str(n3))
b1=gui.Button(w,text="Add",width=10,command=add)
b2=gui.Button(w,text="Sub",width=10,command=sub)
b3=gui.Button(w,text="Mul",width=10,command=mul)
b4=gui.Button(w,text="Div",width=10,command=div)

l1.grid(row=1,column=1)
l2.grid(row=2,column=1)
l3.grid(row=3,column=1)

e1.grid(row=1,column=2)
```

```
e2.grid(row=2,column=2)
e3.grid(row=3,column=2)

b1.grid(row=4,column=1)
b2.grid(row=4,column=2)
b3.grid(row=5,column=1)
b4.grid(row=5,column=2)
```

Checkbutton

A **checkbutton** widget is like a regular button that also holds a binary value of some kind (i.e., a toggle). When pressed, a checkbutton flips the toggle and then invokes its callback. Checkbutton widgets are frequently used to allow users to turn an option on or off.

```
measureSystem = StringVar()
check = ttk.Checkbutton(parent, text='Use Metric',
          command=metricChanged, variable=measureSystem,
          onvalue='metric', offvalue='imperial')
```

Radiobutton

A radiobutton widget lets you choose between one of several mutually exclusive choices. Unlike a checkbutton, they are not limited to just two options. Radiobuttons are always used together in a set, where multiple radiobutton widgets are tied to a single choice or preference. They are appropriate to use when the number of options is relatively small.

```
import tkinter as t
def main():
    w=t.Tk()
    w.geometry("300x300")
    x=t.BooleanVar()
    def fun1():
        print(x.get())
    c1=t.Checkbutton(w,text="Save
Password",onvalue=True,offvalue=False,command=fun1,variable=x)
    c1.place(x=100,y=50)
    y=t.StringVar()
```

```
    def fun2():
        w['bg']=y.get()
    r1=t.Radiobutton(w,text="Red",variable=y,value="red",command=fun2)
    r2=t.Radiobutton(w,text="Green",variable=y,value="green",command=fu
n2)
    r3=t.Radiobutton(w,text="Blue",variable=y,value="blue",command=fun2)
    r1.place(x=100,y=100)
    r2.place(x=100,y=150)
    r3.place(x=100,y=200)

main()
```

Combobox

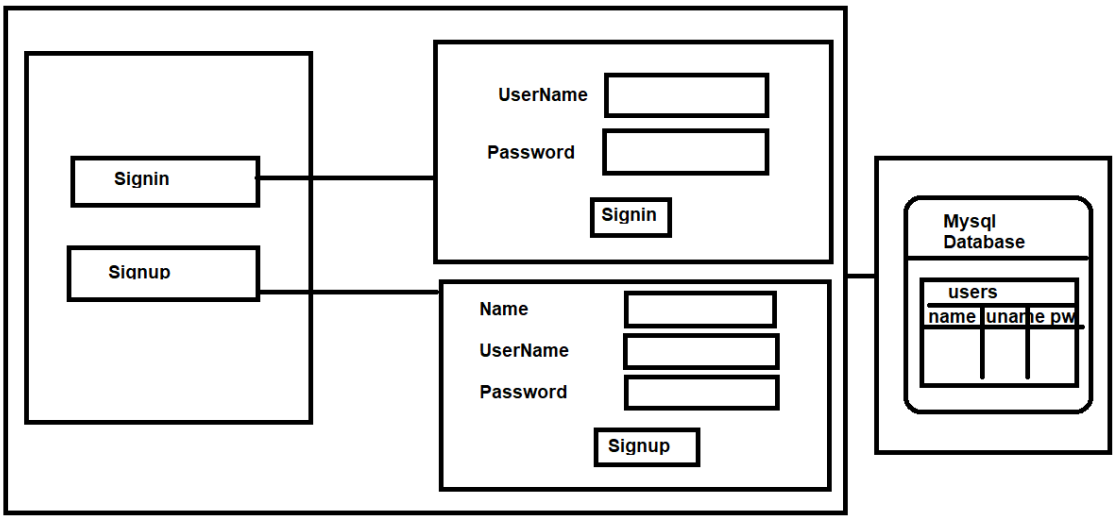
A combobox widget combines an entry with a list of choices. This lets users either choose from a set of values you've provided (e.g., typical settings), but also put in their own value (e.g., for less common cases).

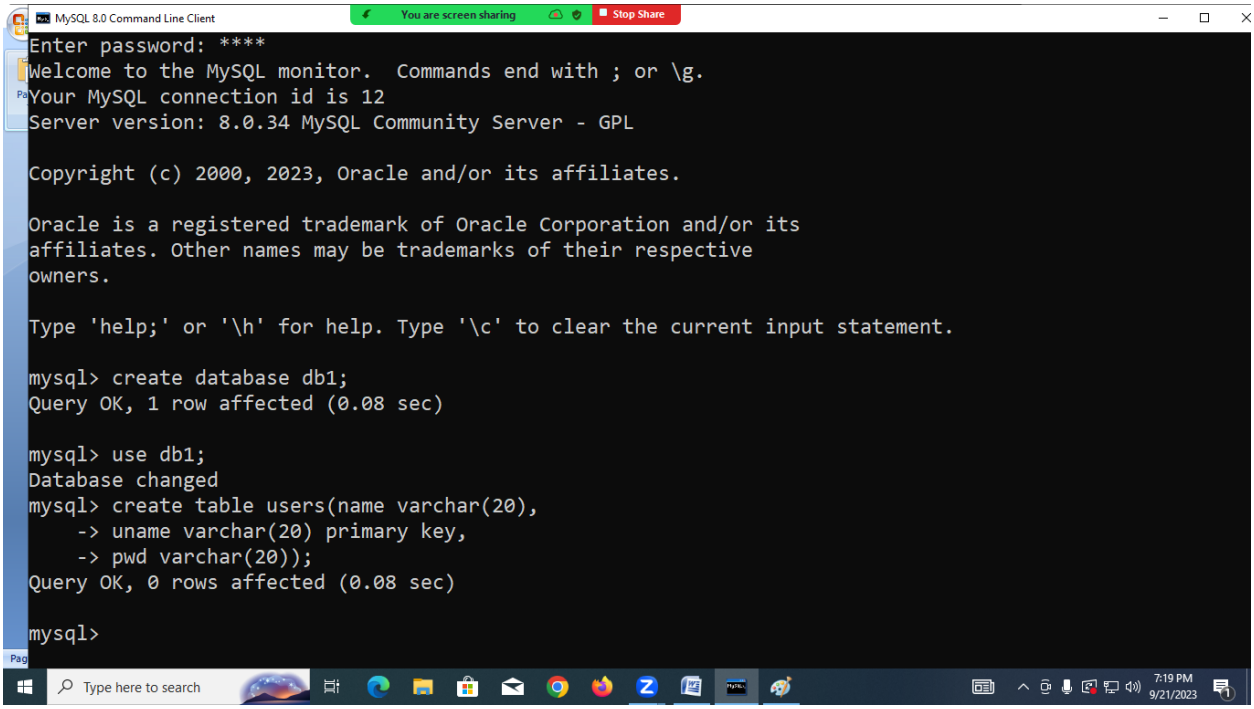
```
countryvar = StringVar()
country = ttk.Combobox(parent, textvariable=countryvar)
```

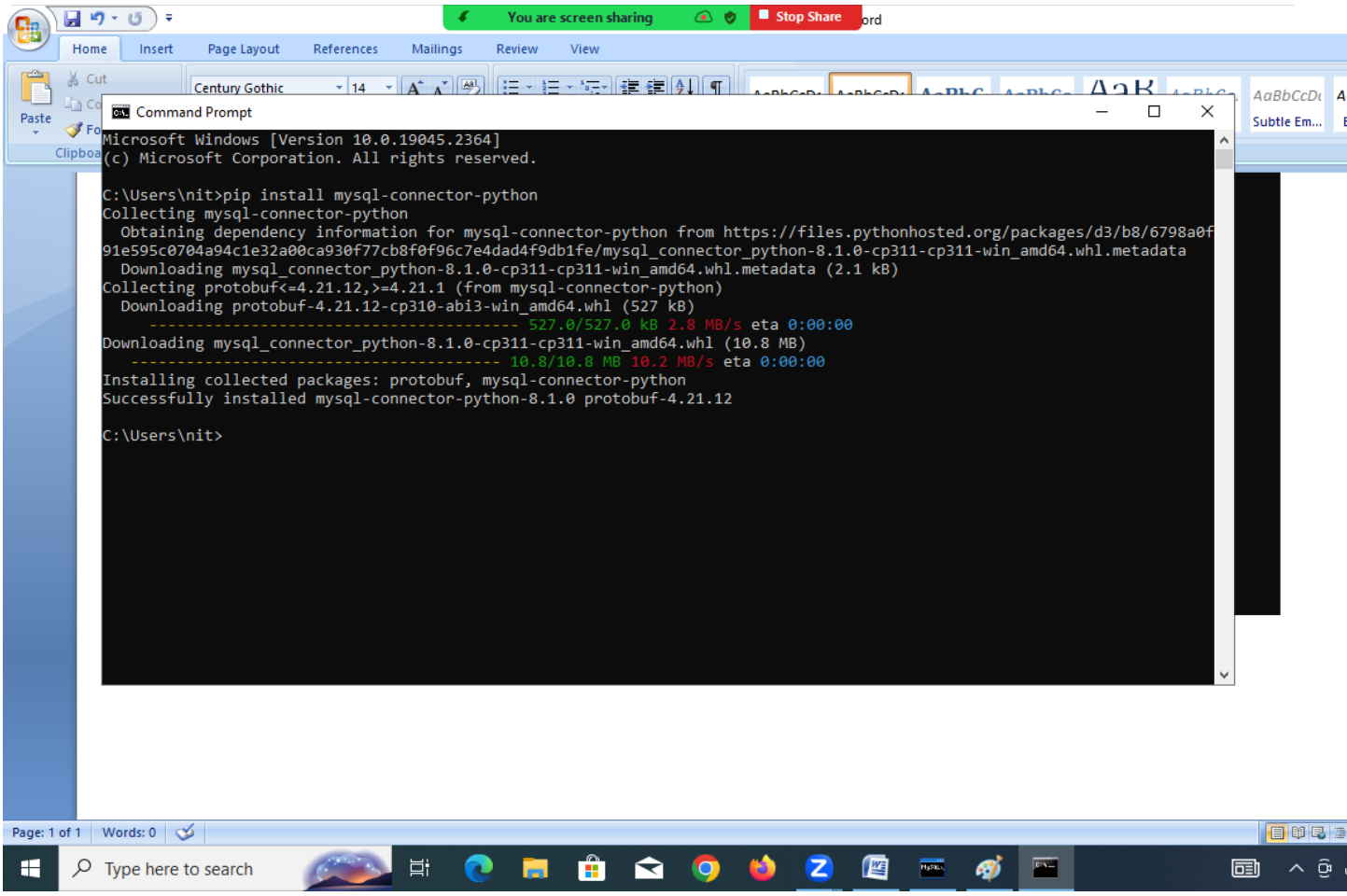
Example:

```
import tkinter as t
from tkinter import ttk
def main():
    w=t.Tk()
    w.geometry("300x300")
    x=t.StringVar()
    def fun1(event):
        print(x.get())
    c=ttk.Combobox(w,textvariable=x)
    c['values']=["Java","Python","Oracle"]
    c.bind("<<ComboboxSelected>>",fun1)
    c.place(x=50,y=50)

main()
```


Signin
Signup
UserName
Password
Signin
Name
UserName
Password
Signup
Mysql
Database
users
name uname pw


MySQL 8.0 Command Line Client
You are screen sharing
Stop Share
Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.34 MySQL Community Server - GPL
Copyright (c) 2000, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> create database db1;
Query OK, 1 row affected (0.08 sec)
mysql> use db1;
Database changed
mysql> create table users(name varchar(20),
-> uname varchar(20) primary key,
-> pwd varchar(20));
Query OK, 0 rows affected (0.08 sec)
mysql>
Type here to search
7:19 PM
9/21/2023




```
import tkinter as gui
import tkinter.messagebox
import mysql.connector
cn=mysql.connector.connect(database="db1",user="root",password="root")
w=gui.Tk()
w.geometry("300x200")
w.title("Banking System")
def signin():
    w1=gui.Tk()
    w1.geometry("300x200")
    w1.title("Signin")
    l1=gui.Label(w1,text="UserName",font=("Arial",14),fg="blue")
    l2=gui.Label(w1,text="Password",font=("Arial",14),fg="blue")
    e1=gui.Entry(w1,width=10,font=("Arial",14))
    e2=gui.Entry(w1,width=10,font=("Arial",14),show="*")
    def signin_verify():
```

```
        c=cn.cursor()
        c.execute("select * from users where uname=%s and
pwd=%s",params=(e1.get(),e2.get()))
        row=c.fetchone()
        if row==None:
            tkinter.messagebox.showinfo(title="invalid",message="Invalid
Usename or Password")
        else:
             tkinter.messagebox.showinfo(title="welcome",message="Welcome"
)
             w1.destroy()
             w.destroy()
             w3=gui.Tk()



    b1=gui.Button(w1,text="Signin",command=signin_verify)
    l1.place(x=50,y=50)
    l2.place(x=50,y=100)
    e1.place(x=150,y=50)
    e2.place(x=150,y=100)
    b1.place(x=120,y=150)


def signup():
    w2=gui.Tk()
    w2.geometry("300x200")
    w2.title("Signup")
    l1=gui.Label(w2,text="Name",font=("Arial",14),fg="Blue")
    l2=gui.Label(w2,text="UserName",font=("Arial",14),fg="Blue")
    l3=gui.Label(w2,text="Password",font=("Arial",14),fg="Blue")
    e1=gui.Entry(w2,width=10,font=("Arial",14))
    e2=gui.Entry(w2,width=10,font=("Arial",14))
    e3=gui.Entry(w2,width=10,font=("Arial",14),show="*")
    def register():
        c=cn.cursor()
        try:
            c.execute("insert into users
values(%s,%s,%s)",params=(e1.get(),e2.get(),e3.get()))
```

```
            tkinter.messagebox.showinfo(title="info",message="User
Registered...")
            e1.delete(0,gui.END)
            e2.delete(0,gui.END)
            e3.delete(0,gui.END)
            cn.commit()
            w2.destroy()
        except:
            tkinter.messagebox.showerror(title="Error",message="User Exists")
    b1=gui.Button(w2,text="Signup",command=register)
    l1.grid(row=1,column=1)
    l2.grid(row=2,column=1)
    l3.grid(row=3,column=1)
    e1.grid(row=1,column=2)
    e2.grid(row=2,column=2)
    e3.grid(row=3,column=2)
    b1.grid(row=4,column=1)

b1=gui.Button(w,text="Signin",font=("Arial",15),command=signin)
b2=gui.Button(w,text="Signup",font=("Arial",15),command=signup)
b1.place(x=100,y=50)
b2.place(x=100,y=150)
```