School of Computer Science Engineering and Technology

Course - BTech	**Type** - Core
Course Code - ECSE105L	**Course Name** - Computational Thinking and Programming
Year - 1st Year	**Semester** - Even
Date -	**Batch** -

Type- Tutorial **# No. (16.1)**

GUI Programming

Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter app:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x it is 'tkinter'.

```
import tkinter
```

There are two main methods used which the user needs to remember while creating the Python application with GUI.

1. **Tk(screenName=None, baseName=None, className='Tk', useTk=1):** To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

m=tkinter.Tk() where m is the name of the main window object

2. **mainloop():** There is a method known by the name mainloop() is used when your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

```
m.mainloop()
import tkinter
m = tkinter.Tk()
'''
widgets are added here
'''
```

```
m.mainloop()
```

tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes class.

1. **pack() method:**It organizes the widgets in blocks before placing in the parent widget.

2. **grid() method:**It organizes the widgets in grid (table-like structure) before placing in the parent widget.

3. **place() method:**It organizes the widgets by placing them on specific positions directed by the programmer.

There are several widgets which you can put in your tkinter application. Some of the major widgets are explained below:

Button

To add a button in your application, this widget is used.

The general syntax is:

```
w=Button(master, option=value)
```

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the Buttons. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- activebackground: to set the background color when button is under the cursor.
- activeforeground: to set the foreground color when button is under the cursor.
- bg: to set he normal background color.
- command: to call a function.
- font: to set the font on the button label.
- image: to set the image on the button.
- width: to set the width of the button.
- height: to set the height of the button.

Canvas

It is used to draw pictures and other complex layout like graphics, text, and widgets.

The general syntax is:

```
w = Canvas(master, option=value)
```

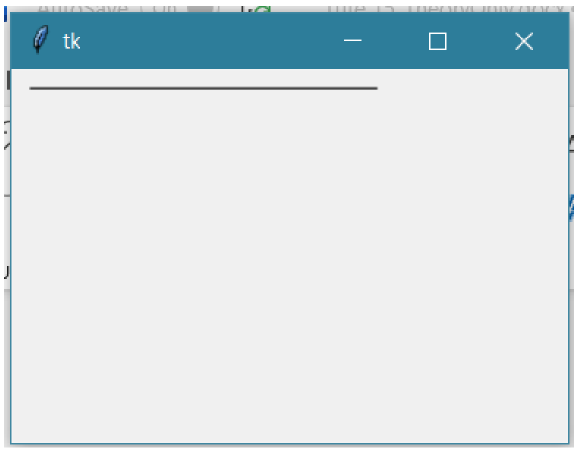
master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- bd: to set the border width in pixels.
- bg: to set the normal background color.
- cursor: to set the cursor used in the canvas.
- highlightcolor: to set the color shown in the focus highlight.
- width: to set the width of the widget.
- height: to set the height of the widget.

```
from tkinter import *
master = Tk()
w = Canvas(master, width=300, height=200)
w.pack()
canvas_height=20
canvas_width=200
y = int(canvas_height / 2)
w.create_line(10, y, canvas_width, y )
mainloop()
```

Output:



CheckButton:

To select any number of options by displaying several options to a user as toggle buttons. The general syntax is:

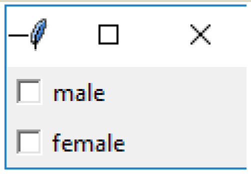
```
w = CheckButton(master, option=value)
```

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- Title: To set the title of the widget.

- activebackground: to set the background color when widget is under the cursor.
- activeforeground: to set the foreground color when widget is under the cursor.
- bg: to set he normal backgrouSteganography
- command: to call a function.
- font: to set the font on the button label.
- image: to set the image on the widget.

```
from tkinter import *
master = Tk()
var1 = IntVar()
Checkbutton(master, text='male', variable=var1).grid(row=0, sticky=W)
var2 = IntVar()
Checkbutton(master, text='female', variable=var2).grid(row=1, sticky=W)
mainloop()
```



Entry

It is used to input the single line text entry from the user.. For multi-line text input, Text widget is used.

The general syntax is:

```
w=Entry(master, option=value)
```

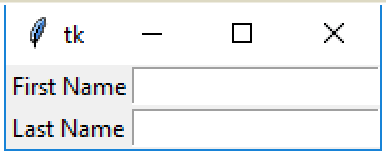
master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- bd: to set the border width in pixels.
- bg: to set the normal background color.
- cursor: to set the cursor used.
- command: to call a function.
- highlightcolor: to set the color shown in the focus highlight.
- width: to set the width of the button.
- height: to set the height of the button.

```
from tkinter import *
master = Tk()
```

```
Label(master, text='First Name').grid(row=0)
Label(master, text='Last Name').grid(row=1)
e1 = Entry(master)
e2 = Entry(master)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
mainloop()
```



Frame

It acts as a container to hold the widgets. It is used for grouping and organizing the widgets. The general syntax is:

```
w = Frame(master, option=value)
```

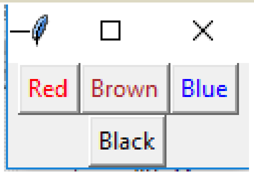
master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- highlightcolor: To set the color of the focus highlight when widget has to be focused.
- bd: to set the border width in pixels.
- bg: to set the normal background color.
- cursor: to set the cursor used.
- width: to set the width of the widget.
- height: to set the height of the widget.

```
from tkinter import *
root = Tk()
frame = Frame(root)
frame.pack()
bottomframe = Frame(root)
bottomframe.pack( side = BOTTOM )
redbutton = Button(frame, text = 'Red', fg ='red')
redbutton.pack( side = LEFT)
greenbutton = Button(frame, text = 'Brown', fg='brown')
greenbutton.pack( side = LEFT )
bluebutton = Button(frame, text ='Blue', fg ='blue')
bluebutton.pack( side = LEFT )
blackbutton = Button(bottomframe, text ='Black', fg ='black')
blackbutton.pack( side = BOTTOM)
```

```
root.mainloop()
```



Label

It refers to the display box where you can put any text or image which can be updated any time as per the code.

The general syntax is:

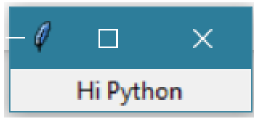
```
w=Label(master, option=value)
```

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- bg: to set the normal background color.
- bg to set the normal background color.
- command: to call a function.
- font: to set the font on the button label.
- image: to set the image on the button.
- width: to set the width of the button.
- height” to set the height of the button.

```
from tkinter import *
root = Tk()
w = Label(root, text='Hi Python')
w.pack()
root.mainloop()
```



Listbox

It offers a list to the user from which the user can accept any number of options.

The general syntax is:

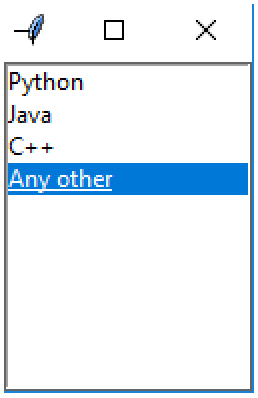
```
w = Listbox(master, option=value)
```

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- highlightcolor: To set the color of the focus highlight when widget has to be focused.
- bg: to set the normal background color.
- bd: to set the border width in pixels.
- font: to set the font on the button label.
- image: to set the image on the widget.
- width: to set the width of the widget.
- height: to set the height of the widget.

```
from tkinter import *
top = Tk()
Lb = Listbox(top)
Lb.insert(1, 'Python')
Lb.insert(2, 'Java')
Lb.insert(3, 'C++')
Lb.insert(4, 'Any other')
Lb.pack()
top.mainloop()
```



MenuButton

It is a part of top-down menu which stays on the window all the time. Every menubutton has its own functionality. The general syntax is:

```
w = MenuButton(master, option=value)
```

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- activebackground: To set the background when mouse is over the widget.
- activeforeground: To set the foreground when mouse is over the widget.
- bg: to set the normal background color.
- bd: to set the size of border around the indicator.
- cursor: To appear the cursor when the mouse over the menubutton.
- image: to set the image on the widget.
- width: to set the width of the widget.
- height: to set the height of the widget.
- highlightcolor: To set the color of the focus highlight when widget has to be focused.

Menu

It is used to create all kinds of menus used by the application.

The general syntax is:

```
w = Menu(master, option=value)
```

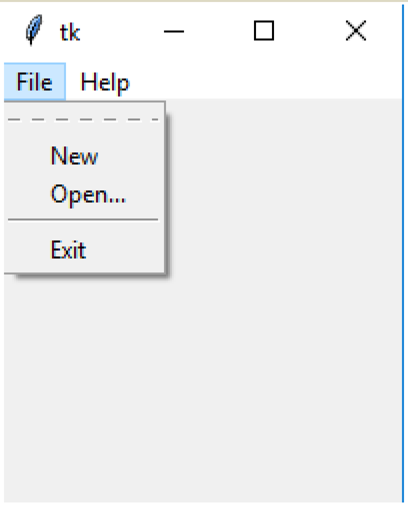
master is the parameter used to represent the parent window.

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- title: To set the title of the widget.
- activebackground: to set the background color when widget is under the cursor.
- activeforeground: to set the foreground color when widget is under the cursor.
- bg: to set he normal background color.
- command: to call a function.
- font: to set the font on the button label.
- image: to set the image on the widget.

```
from tkinter import *
root = Tk()
menu = Menu(root)
root.config(menu=menu)
filemenu = Menu(menu)
menu.add_cascade(label='File', menu=filemenu)
filemenu.add_command(label='New')
filemenu.add_command(label='Open...')
filemenu.add_separator()
filemenu.add_command(label='Exit', command=root.quit)
helpmenu = Menu(menu)
```

```
menu.add_cascade(label='Help', menu=helpmenu)
helpmenu.add_command(label='About')
mainloop()
```



Message: It refers to the multi-line and non-editable text. It works same as that of Label.

The general syntax is:

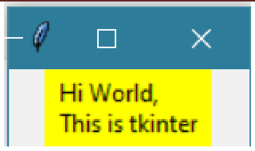
```
w = Message(master, option=value)
```

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- bd: to set the border around the indicator.
- bg: to set he normal background color.
- font: to set the font on the button label.
- image: to set the image on the widget.
- width: to set the width of the widget.
- height: to set the height of the widget.

```
from tkinter import *
main = Tk()
ourMessage ='Hi World, This is tkinter'
messageVar = Message(main, text = ourMessage)
messageVar.config(bg='Yellow')
messageVar.pack( )
main.mainloop( )
```


Hi World,
This is tkinter

RadioButton: It is used to offer multi-choice option to the user. It offers several options to the user and the user has to choose one option.

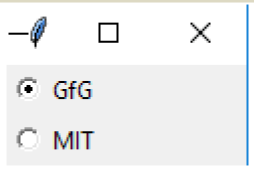
The general syntax is:

```
w = RadioButton(master, option=value)
```

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- activebackground: to set the background color when widget is under the cursor.
- activeforeground: to set the foreground color when widget is under the cursor.
- bg: to set he normal background color.
- command: to call a function.
- font: to set the font on the button label.
- image: to set the image on the widget.
- width: to set the width of the label in characters.
- height: to set the height of the label in characters.

```
from tkinter import *
root = Tk()
v = IntVar()
Radiobutton(root, text='GfG', variable=v, value=1).pack(anchor=W)
Radiobutton(root, text='MIT', variable=v, value=2).pack(anchor=W)
mainloop()
```



Scale: It is used to provide a graphical slider that allows to select any value from that scale. The general syntax is:

```
w = Scale(master, option=value)
```

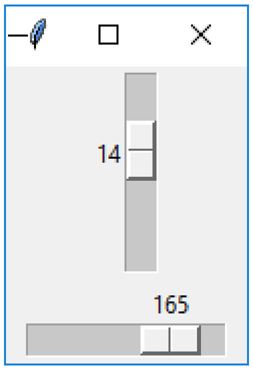
master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- cursor: To change the cursor pattern when the mouse is over the widget.

- activebackground: To set the background of the widget when mouse is over the widget.
- bg: to set he normal background color.
- orient: Set it to HORIZONTAL or VERTICAL according to the requirement.
- from_: To set the value of one end of the scale range.
- to: To set the value of the other end of the scale range.
- image: to set the image on the widget.
- width: to set the width of the widget.

```
from tkinter import *
master = Tk()
w = Scale(master, from_=0, to=42)
w.pack()
w = Scale(master, from_=0, to=200, orient=HORIZONTAL)
w.pack()
mainloop()
```



Scrollbar: It refers to the slide controller which will be used to implement listed widgets.

The general syntax is:

```
w = Scrollbar(master, option=value)
```

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

```
o      width: to set the width of the widget.
o      activebackground: To set the background when mouse is over the widget.
o      bg: to set he normal background color.
o      bd: to set the size of border around the indicator.
o      cursor: To appear the cursor when the mouse over the menubutton.
from tkinter import *
root = Tk()
```

```
scrollbar = Scrollbar(root)
scrollbar.pack( side = RIGHT, fill = Y )
mylist = Listbox(root, yscrollcommand = scrollbar.set )
for line in range(100):
   mylist.insert(END, 'This is line number' + str(line))
mylist.pack( side = LEFT, fill = BOTH )
scrollbar.config( command = mylist.yview )
mainloop()
```

Text: To edit a multi-line text and format the way it has to be displayed.

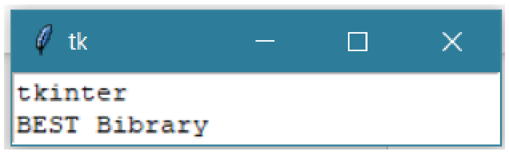
The general syntax is:

```
w  =Text(master, option=value)
```

There are number of options which are used to change the format of the text. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- highlightcolor: To set the color of the focus highlight when widget has to be focused.
- insertbackground: To set the background of the widget.
- bg: to set he normal background color.
- font: to set the font on the button label.
- image: to set the image on the widget.
- width: to set the width of the widget.
- height: to set the height of the widget.

```
from tkinter import *
root = Tk()
T = Text(root, height=2, width=30)
T.pack()
T.insert(END, 'tkinter\nBEST Bibrary\n')
mainloop()
```



```
tkinter
BEST Bibrary
```

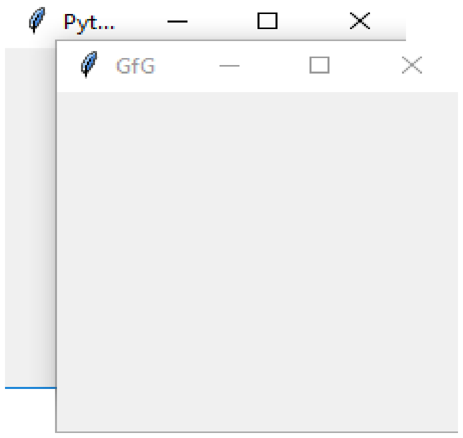
TopLevel: This widget is directly controlled by the window manager. It don't need any parent window to work on.The general syntax is:

```
w = TopLevel(master, option=value)
```

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- bg: to set he normal background color.
- bd: to set the size of border around the indicator.
- cursor: To appear the cursor when the mouse over the menubutton.
- width: to set the width of the widget.
- height: to set the height of the widget.

```
from tkinter import *
root = Tk()
root.title('GfG')
top = Toplevel()
top.title('Python')
top.mainloop()
```



SpinBox: It is an entry of ‘Entry’ widget. Here, value can be input by selecting a fixed value of numbers.The general syntax is:

```
w = SpinBox(master, option=value)
```

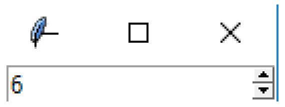
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- bg: to set he normal background color.
- bd: to set the size of border around the indicator.
- cursor: To appear the cursor when the mouse over the menubutton.
- command: To call a function.
- width: to set the width of the widget.
- activebackground: To set the background when mouse is over the widget.
- disabledbackground: To disable the background when mouse is over the widget.
- from_: To set the value of one end of the range.

- to: To set the value of the other end of the range.

```
from tkinter import *
master = Tk()
w = Spinbox(master, from_ = 0, to = 10)
w.pack()
mainloop()
```

Output:



PannedWindow

It is a container widget which is used to handle number of panes arranged in it. The general syntax is:

```
w = PannedWindow(master, option=value)
```

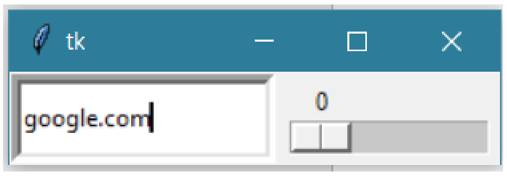
master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- bg: to set he normal background color.
- bd: to set the size of border around the indicator.
- cursor: To appear the cursor when the mouse over the menubutton.
- width: to set the width of the widget.
- height: to set the height of the widget.

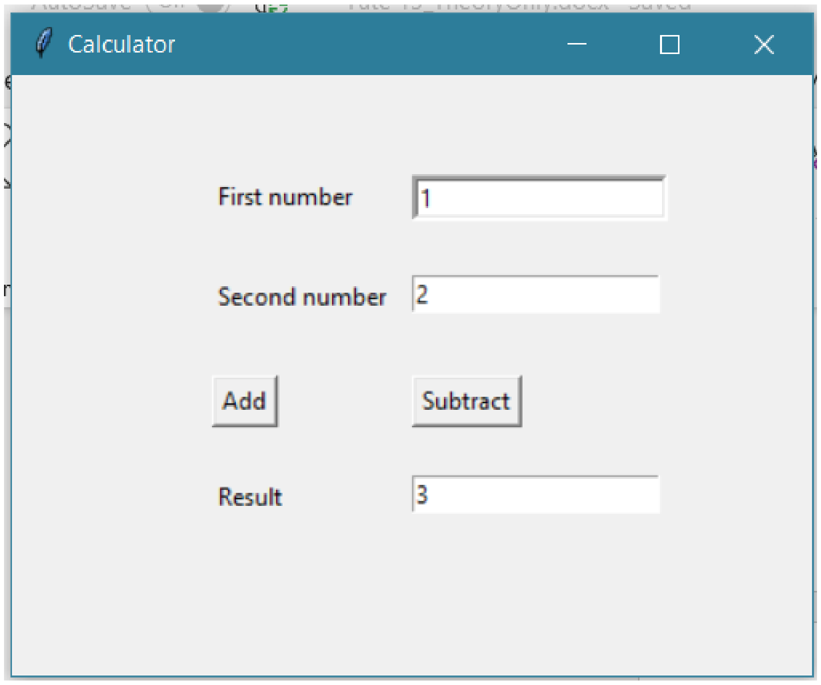
```
from tkinter import *
m1 = PanedWindow()
m1.pack(fill = BOTH, expand = 1)
left = Entry(m1, bd = 5)
m1.add(left)
m2 = PanedWindow(m1, orient = VERTICAL)
m1.add(m2)
top = Scale( m2, orient = HORIZONTAL)
m2.add(top)
mainloop()
```

Output:



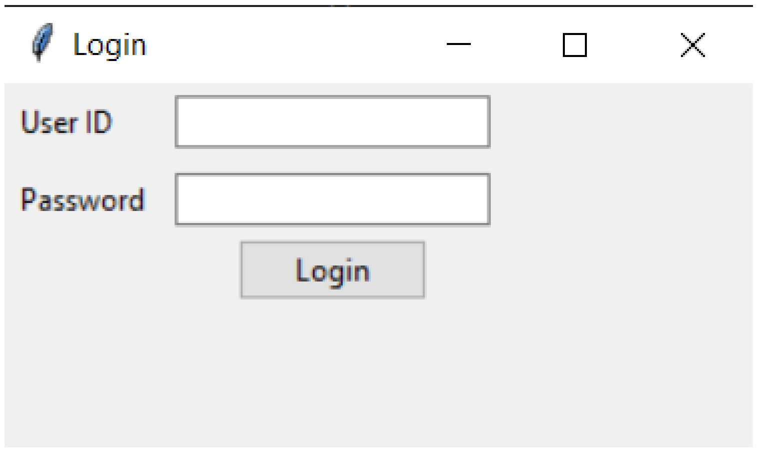
Making a calculator using tkinter library

```
from tkinter import *
class MyWindow:
    def __init__(self, win):
        self.lbl1=Label(win, text='First number')
        self.lbl2=Label(win, text='Second number')
        self.lbl3=Label(win, text='Result')
        self.t1=Entry(bd=3)
        self.t2=Entry()
        self.t3=Entry()
        self.btn1 = Button(win, text='Add')
        self.btn2=Button(win, text='Subtract')
        self.lbl1.place(x=100, y=50)
        self.t1.place(x=200, y=50)
        self.lbl2.place(x=100, y=100)
        self.t2.place(x=200, y=100)
        self.b1=Button(win, text='Add', command=self.add)
        self.b2=Button(win, text='Subtract')
        self.b2.bind('<Button-1>', self.sub)
        self.b1.place(x=100, y=150)
        self.b2.place(x=200, y=150)
        self.lbl3.place(x=100, y=200)
        self.t3.place(x=200, y=200)
    def add(self):
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
        num2=int(self.t2.get())
        result=num1+num2
        self.t3.insert(END, str(result))
    def sub(self, event):
        self.t3.delete(0, 'end')
        num1=int(self.t1.get())
        num2=int(self.t2.get())
        result=num1-num2
        self.t3.insert(END, str(result))

window=Tk()
mywin=MyWindow(window)
window.title('Calculator')
window.geometry("400x300+30+30")
window.mainloop()
```



Problems:

1. Make a login screen which will ask user to enter user id and password. Your program should check for the correct user id and password. If both are correct then it should print "Welcome". If the credentials are incorrect then it should print "Incorrect credentials".



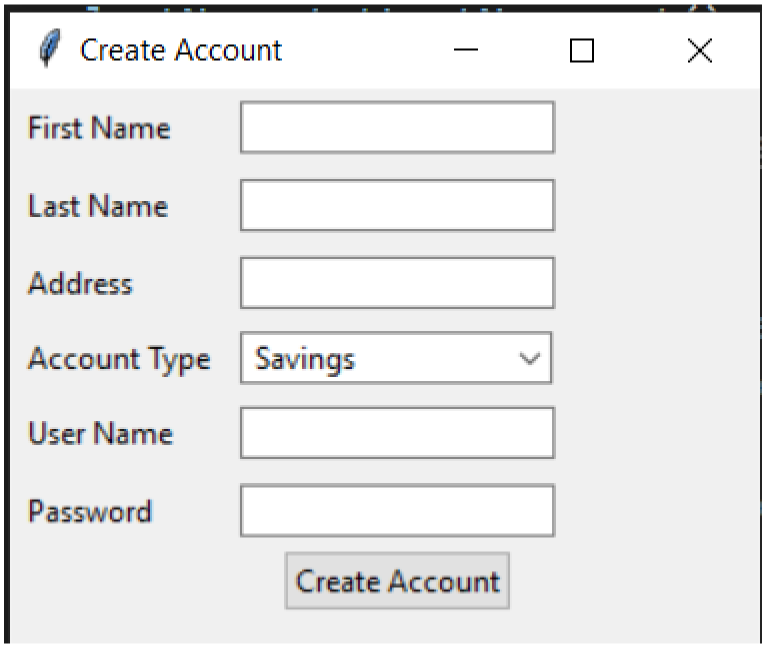
```
from cgitb import text
from tkinter import *
from tkinter import messagebox
```

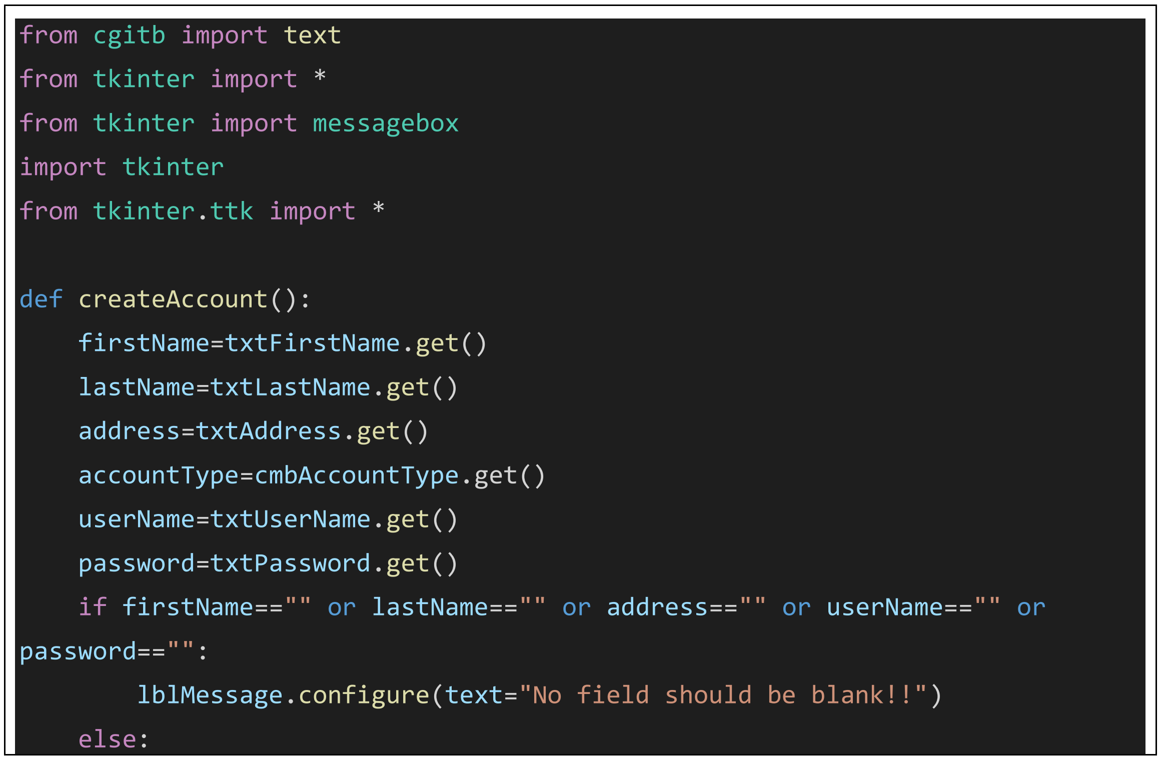
```python
import tkinter
from tkinter.ttk import *

def doLogin():
    userid=txtUserid.get()
    password=txtPassword.get()
    if userid=="" or password=="":
        lblMessage.configure(text="Userid or password can't be blank")
    else:
        if userid=="ABC" and password=="ABC":
            lblMessage.configure(text="You are welcome")
        else:
            lblMessage.configure(text="Incorrect credentials")

mainWindow=Tk()
mainWindow.geometry("300x300")
lblUserid = Label(mainWindow, text = "User ID")
lblUserid.grid(row=0,column=0,sticky=tkinter.W, padx=5, pady=5)
txtUserid=Entry(mainWindow,width=20)
txtUserid.grid(row=0,column=1,sticky=tkinter.W, padx=5, pady=5)
lblPassword=Label(mainWindow,text="Password")
lblPassword.grid(row=1,column=0,sticky=tkinter.W, padx=5, pady=5)
txtPassword=Entry(mainWindow,width=20)
txtPassword.grid(row=1,column=1,sticky=tkinter.W, padx=5, pady=5)
btnLogin=Button(mainWindow,text="Login",command=doLogin)
btnLogin.grid(row=2,column=1)
lblMessage=Label(mainWindow)
lblMessage.grid(row=3,column=1)

mainWindow.mainloop()
```

2. Create a GUI for account opening in bank. It should be able to take following information from user
    a. First name
    b. Last name
    c. Address
    d. Account Type (Current/Saving/Joint)

e. Username
f. Password

It should also have "Create Account" button. No field should be blank. Upon successful creation a message should be displayed.

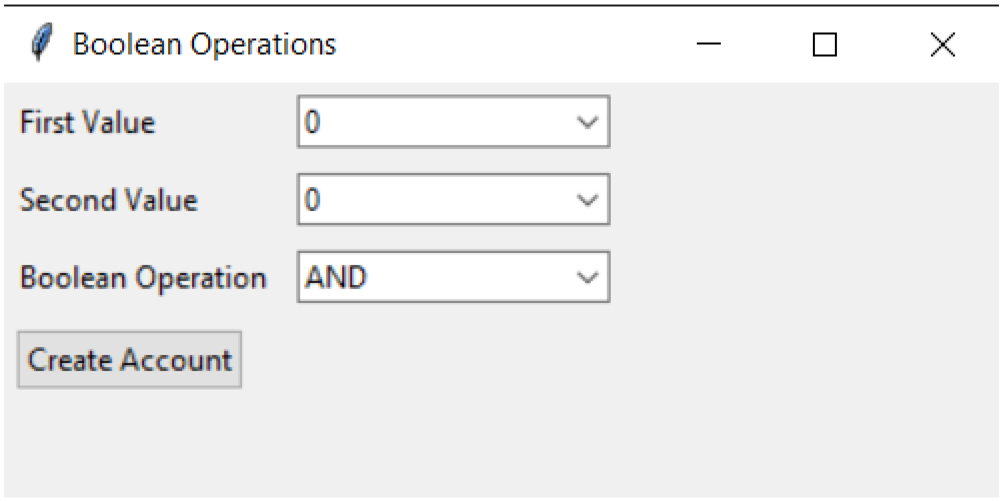




```
from cgitb import text
from tkinter import *
from tkinter import messagebox
import tkinter
from tkinter.ttk import *

def createAccount():
    firstName=txtFirstName.get()
    lastName=txtLastName.get()
    address=txtAddress.get()
    accountType=cmbAccountType.get()
    userName=txtUserName.get()
    password=txtPassword.get()
    if firstName=="" or lastName=="" or address=="" or userName=="" or 
password=="":
        lblMessage.configure(text="No field should be blank!!")
    else:
        lblMessage.configure(text="You are registered.")


mainWindow=Tk()
mainWindow.geometry("300x500")
mainWindow.title("Create Account")
lblFirstName = Label(mainWindow, text = "First Name")
lblFirstName.grid(row=0,column=0,sticky=tkinter.W, padx=5, pady=5)
txtFirstName=Entry(mainWindow,width=20)
txtFirstName.grid(row=0,column=1,sticky=tkinter.W, padx=5, pady=5)
lblLastName=Label(mainWindow,text="Last Name")
lblLastName.grid(row=1,column=0,sticky=tkinter.W, padx=5, pady=5)
txtLastName=Entry(mainWindow,width=20)
txtLastName.grid(row=1,column=1,sticky=tkinter.W, padx=5, pady=5)
lblAddress=Label(mainWindow,text="Address")
lblAddress.grid(row=2,column=0,sticky=tkinter.W, padx=5, pady=5)
txtAddress=Entry(mainWindow,width=20)
txtAddress.grid(row=2,column=1,sticky=tkinter.W, padx=5, pady=5)

lblAccount=Label(mainWindow,text="Account Type")
lblAccount.grid(row=3,column=0,sticky=tkinter.W, padx=5, pady=5)

cmbAccountType = tkinter.ttk.Combobox(mainWindow, width = 17, state =
"readonly")

cmbAccountType['values'] = (' Savings',
                          ' Current',
                          ' Joint')

cmbAccountType.grid(row=3,column = 1)
cmbAccountType.current(0)
lblUserName=Label(mainWindow,text="User Name")
lblUserName.grid(row=4,column=0,sticky=tkinter.W, padx=5, pady=5)
txtUserName=Entry(mainWindow,width=20)
txtUserName.grid(row=4,column=1,sticky=tkinter.W, padx=5, pady=5)
lblPassword=Label(mainWindow,text="Password")
```

```
lblPassword.grid(row=5,column=0,sticky=tkinter.W, padx=5, pady=5)
txtPassword=Entry(mainWindow,width=20)
txtPassword.grid(row=5,column=1,sticky=tkinter.W, padx=5, pady=5)

btnCreate=Button(mainWindow,text="Create Account",command=createAccount)
btnCreate.grid(row=6,column=1)
lblMessage=Label(mainWindow)
lblMessage.grid(row=7,column=1)

mainWindow.mainloop()
```

3. Create a GUI program using Tkinter libraries to input two boolean values (0 or 1) and select a boolean operation from set of boolean operations AND, OR, NAND, NOR, XOR. Result should be printed in a text box/label next to the boolean selection box.


Boolean Operations

First Value 0

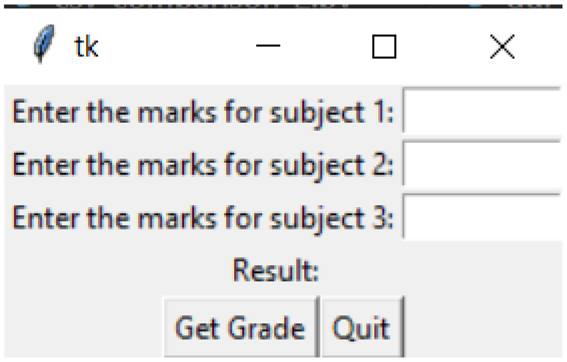
Second Value 0

Boolean Operation AND

```
from cgitb import text
from tkinter import *
from tkinter import messagebox
import tkinter
from tkinter.ttk import *

def createAccount():
    firstValue=int(cmbValue1.get())
    secondValue=int(cmbValue2.get())
    operation=cmbOperation.get()
    result=""
    if operation=="AND":
        result=firstValue & secondValue
    elif operation=="OR":
        result= firstValue | secondValue
    elif operation=="XOR":
        result=firstValue ^ secondValue
    elif operation=="NAND":
        if firstValue == 1 and secondValue == 1:
            result=0
        else:
            result=1
    elif operation=="NOR":
        if(firstValue == 0) and (secondValue == 0):
            result= 1
        elif(firstValue == 0) and (secondValue == 1):
            result= 0
        elif(firstValue == 1) and (secondValue == 0):
            result= 0
        elif(firstValue == 1) and (secondValue == 1):
            result= 0

    lblMessage.configure(text=str(result))

mainWindow=Tk()
mainWindow.geometry("400x300")
mainWindow.title("Boolean Operations")

lblValue1=Label(mainWindow,text="First Value")
lblValue1.grid(row=0,column=0,sticky=tkinter.W, padx=5, pady=5)
cmbValue1 = tkinter.ttk.Combobox(mainWindow, width = 17, state =
"readonly")
cmbValue1['values'] = ('0','1')
cmbValue1.grid(row=0,column = 1,sticky=tkinter.W, padx=5, pady=5)
cmbValue1.current(0)

lblValue2=Label(mainWindow,text="Second Value")
```

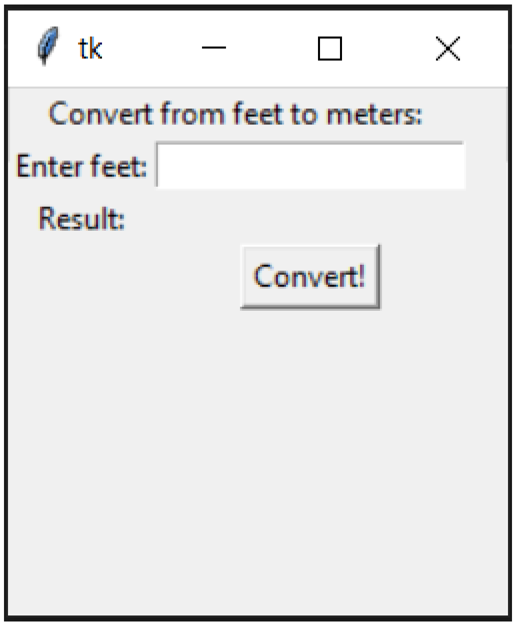
```
lblValue2.grid(row=1,column=0,sticky=tkinter.W, padx=5, pady=5)
cmbValue2 = tkinter.ttk.Combobox(mainWindow, width = 17, state =
"readonly")
cmbValue2['values'] = ('0','1')
cmbValue2.grid(row=1,column = 1,sticky=tkinter.W, padx=5, pady=5)
cmbValue2.current(0)

lblOperation=Label(mainWindow,text="Boolean Operation")
lblOperation.grid(row=2,column=0,sticky=tkinter.W, padx=5, pady=5)
cmbOperation = tkinter.ttk.Combobox(mainWindow, width = 17, state =
"readonly")
cmbOperation['values'] = ('AND', 'OR', 'NAND', 'NOR', 'XOR')
cmbOperation.grid(row=2,column = 1,sticky=tkinter.W, padx=5, pady=5)
cmbOperation.current(0)

btnGetResult=Button(mainWindow,text="Create Account",command=createAccount)
btnGetResult.grid(row=3,column=0,sticky=tkinter.W, padx=5, pady=5)
lblMessage=Label(mainWindow)
lblMessage.grid(row=3,column=1)

mainWindow.mainloop()
```

4. Create a GUI which will take marks of three subjects from user and provide grades. Numbers should be taken from 100 and output should be printed as per the following:
   a. Percentage greater than or equal to 80 then print Excellent
   b. Percentage greater than or equal to 65 but less than 80 then print Good
   c. Percentage greater than or equal to 50 but less than 65 then print Pass
   d. Percentage less than 50 then print Fail

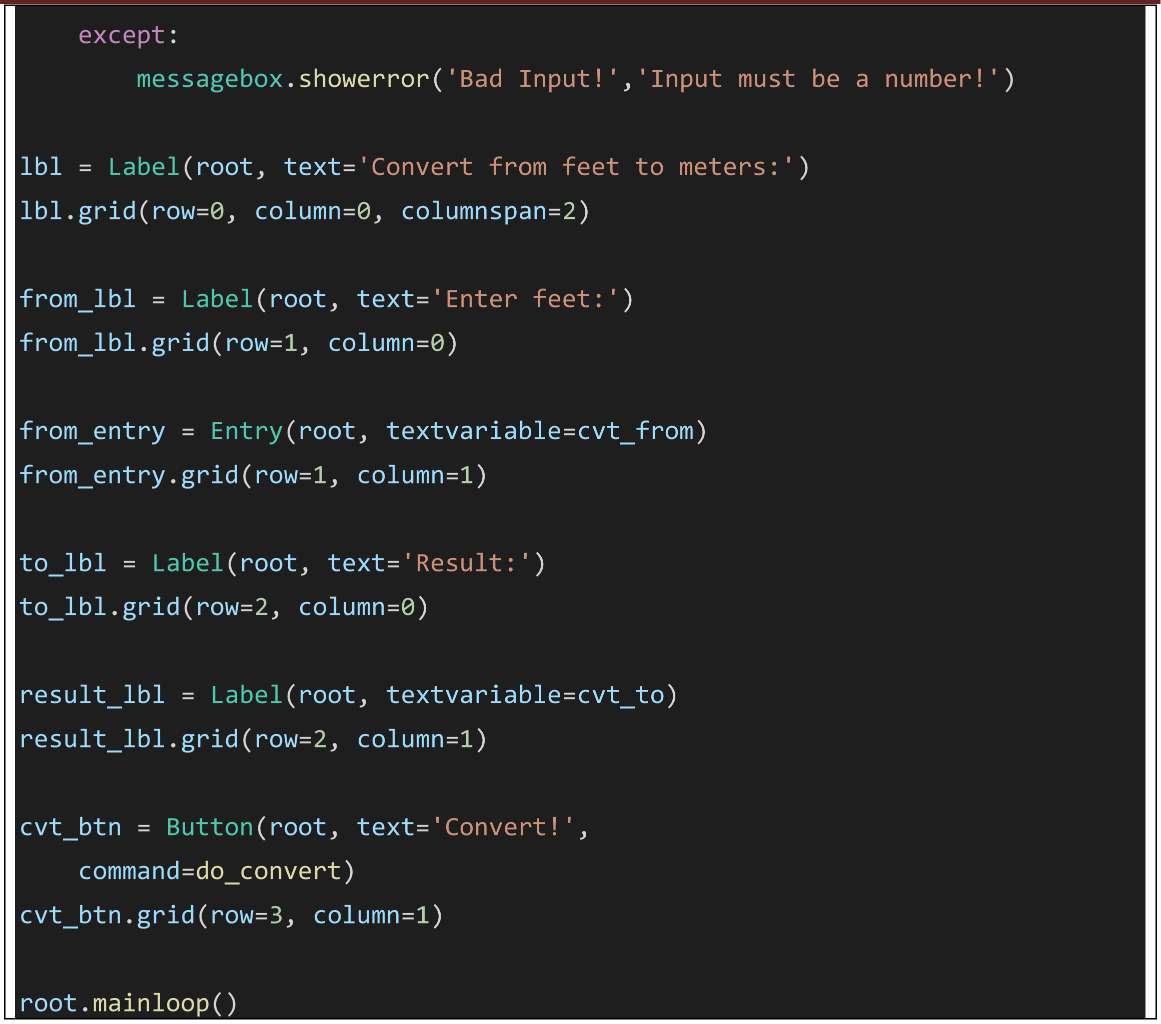

tk
Enter the marks for subject 1:
Enter the marks for subject 2:
Enter the marks for subject 3:
Result:
Get Grade
Quit

```python
from tkinter import *
from tkinter import messagebox

class ResultCalculator:
    def __init__(self):
        self._main_window = Tk()

        self._test1_frame = Frame(self._main_window)
        self._test2_frame = Frame(self._main_window)
        self._test3_frame = Frame(self._main_window)
        self._result_frame = Frame(self._main_window)
        self._button_frame = Frame(self._main_window)

        self._test1_label = Label(self._test1_frame,text='Enter the marks
for subject 1:')
        self._test1_entry = Entry(self._test1_frame, width=10)
        self._test1_label.pack(side='left')
        self._test1_entry.pack(side='left')

        self._test2_label = Label(self._test2_frame,
                                  text='Enter the marks for subject 2:')
        self._test2_entry = Entry(self._test2_frame, width=10)
        self._test2_label.pack(side='left')
        self._test2_entry.pack(side='left')

        self._test3_label = Label(self._test3_frame,
                                  text='Enter the marks for subject 3:')
        self._test3_entry = Entry(self._test3_frame, width=10)
        self._test3_label.pack(side='left')
        self._test3_entry.pack(side='left')

        self._result_label = Label(self._result_frame, text='Result:')
        self._result_disp = StringVar()  # To update avg_label
        self._avg_label = Label(self._result_frame,
textvariable=self._result_disp)
        self._result_label.pack(side='left')
        self._avg_label.pack(side='left')

        self._calc_button = Button(self._button_frame, text='Get Grade',
                                   command=self._calculate_result)
        self._quit_button = Button(self._button_frame, text='Quit',
                                   command=self._main_window.destroy)
        self._calc_button.pack(side='left')
        self._quit_button.pack(side='left')

        self._test1_frame.pack()
        self._test2_frame.pack()
        self._test3_frame.pack()
        self._result_frame.pack()
        self._button_frame.pack()

        mainloop()

    def _calculate_result(self):
        test1 = float(self._test1_entry.get())
        test2 = float(self._test2_entry.get())
        test3 = float(self._test3_entry.get())

        if test1>100 or test2>100 or test3>300:
            messagebox.showerror("Marks can not be more than 100")

        sum = test1 + test2 + test3
        percentage=(sum/300)*100

        if percentage>=80:
            self._result_disp.set(f"Percentage is: {percentage}.
Excellent!..")
        elif percentage>=65 and percentage<80:
            self._result_disp.set(f"Percentage is: {percentage}. Good!..")
        elif percentage>=50 and percentage<65:
            self._result_disp.set(f"Percentage is: {percentage}. Pass!..")
        elif percentage<50:
            self._result_disp.set(f"Percentage is: {percentage}. Fail!..")

result = ResultCalculator()
```

5. Create a GUI for converting feet to meters. It should be able to validate the user input. If user enters nothing in the text box or the input value is not a number then the appropriate message should be displayed using message box.

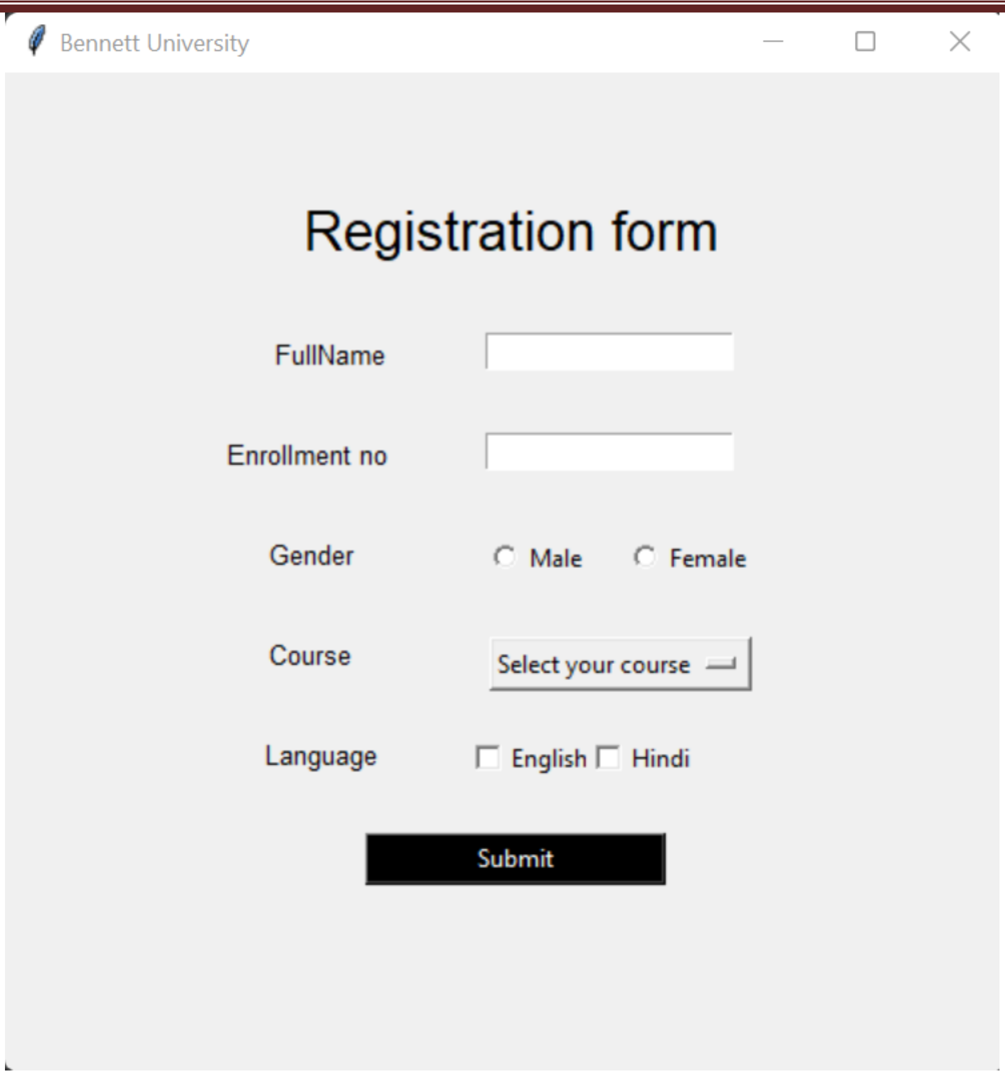


```
from tkinter import *
from tkinter.ttk import *
from tkinter import messagebox

root = Tk()
root.geometry("200x200")

cvt_from = StringVar()
cvt_to = StringVar()

def do_convert():
    cvt_to.set('')
    try:
        feet_val = float(cvt_from.get())
        meters_val = feet_val * 0.3048
        cvt_to.set(str(meters_val))
```



```
    except:
        messagebox.showerror('Bad Input!','Input must be a number!')

lbl = Label(root, text='Convert from feet to meters:')
lbl.grid(row=0, column=0, columnspan=2)

from_lbl = Label(root, text='Enter feet:')
from_lbl.grid(row=1, column=0)

from_entry = Entry(root, textvariable=cvt_from)
from_entry.grid(row=1, column=1)

to_lbl = Label(root, text='Result:')
to_lbl.grid(row=2, column=0)

result_lbl = Label(root, textvariable=cvt_to)
result_lbl.grid(row=2, column=1)

cvt_btn = Button(root, text='Convert!',
    command=do_convert)
cvt_btn.grid(row=3, column=1)

root.mainloop()
```

6. Design registration form using Python Tkinter. Registration form details are given below:

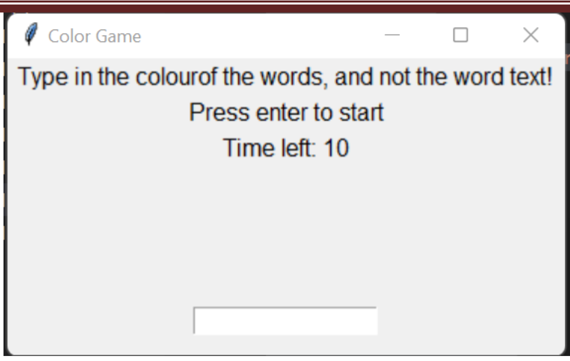


```
from tkinter import *
root = Tk()
root.geometry("500x500")
root.title('Bennett University')
label_0 =Label(root,text="Registration form", width=20,font=("bold",20))
label_0.place(x=90,y=60)
label_1 =Label(root,text="FullName", width=20,font=("bold",10))
label_1.place(x=80,y=130)
entry_1=Entry(root)
entry_1.place(x=240,y=130)
label_3 =Label(root,text="Enrollment no", width=20,font=("bold",10))
label_3.place(x=68,y=180)
```

```
entry_3=Entry(root)
entry_3.place(x=240,y=180)
label_4 =Label(root,text="Gender", width=20,font=("bold",10))
label_4.place(x=70,y=230)
var=IntVar()
Radiobutton(root,text="Male",padx= 5, variable= var,
value=1).place(x=235,y=230)
Radiobutton(root,text="Female",padx= 20, variable= var,
value=2).place(x=290,y=230)
label_5=Label(root,text="Course",width=20,font=("bold",10))
label_5.place(x=70,y=280)
list_of_course=[ 'B.Tech (CSE)', 'B.Tech(EC)', 'B.Tech (ME)','BCA' ,'BBA',
'M.Tech' , 'MBA', 'BA LLB','Mass COMM' ,'PhD']
#the variable 'c' mentioned here holds String Value, by default ""
c=StringVar()
droplist=OptionMenu(root,c, *list_of_course)
droplist.config(width=15)
c.set('Select your course')
droplist.place(x=240,y=280)
label_6=Label(root,text="Language",width=20,font=('bold',10))
label_6.place(x=75,y=330)
var1=IntVar()
Checkbutton(root,text="English", variable=var1).place(x=230,y=330)
var2=IntVar()
Checkbutton(root,text="Hindi", variable=var2).place(x=290,y=330)
Button(root, text='Submit' ,
width=20,bg="black",fg='white').place(x=180,y=380)
root.mainloop()
```

7. Design a Color game using Tkinter in Python: Game rule is defined below:

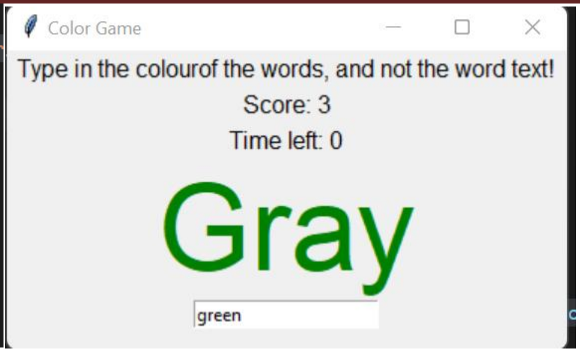
   In this game, the player has to enter the color of the word that appears on the screen, and hence the score increases by one, the total time to play this game is 10 seconds. Colors used in this game are 'Red', 'Blue', 'Green', 'Black', 'Yellow', 'Purple', 'Gray', 'Orange'. The interface will display the name of different colors in different colors. The Player has to identify the color and enter the correct color name to win the game.


Color Game

Type in the colourof the words, and not the word text!

Press enter to start

Time left: 10


Color Game

Type in the colourof the words, and not the word text!

Score: 3

Time left: 0

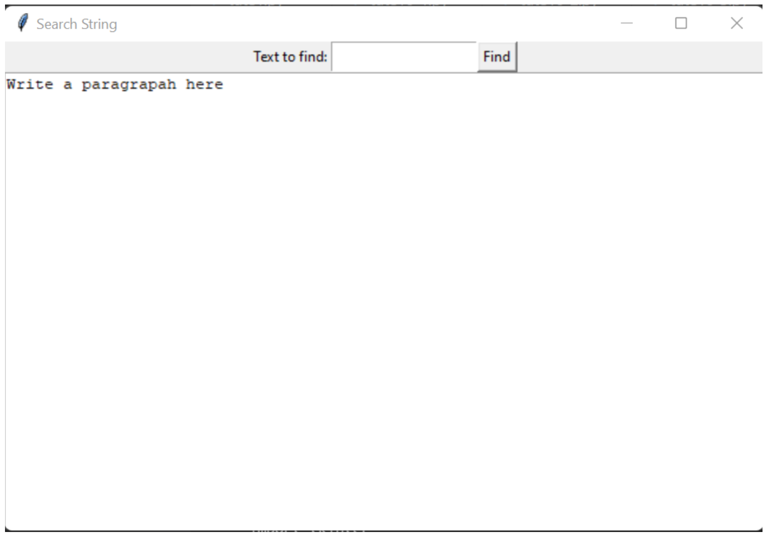
Gray

green

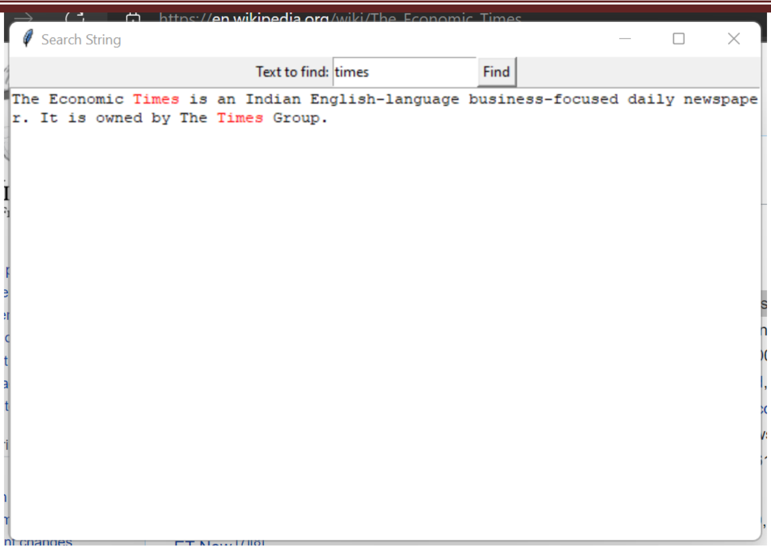
```
import tkinter
import random
colours = ['Red','Blue','Green','Black','Yellow', 'Purple', 'Gray',
'Orange']
score = 0
timeleft = 10
def startGame(event):
    if timeleft == 10:
        countdown()
    nextColour()
def nextColour():
    global score
    global timeleft
    if timeleft > 0:
        e.focus_set()
        if e.get().lower() == colours[1].lower():
            score += 1
        e.delete(0, tkinter.END)
        random.shuffle(colours)
        label.config(fg = str(colours[1]), text = str(colours[0]))
        scoreLabel.config(text = "Score: " + str(score))
def countdown():
    global timeleft
    if timeleft > 0:
        timeleft -= 1
        timeLabel.config(text = "Time left: "+
str(timeleft))
        timeLabel.after(1000, countdown)
```

```
root = tkinter.Tk()
root.title("Color Game")
root.geometry("375x200")
instructions = tkinter.Label(root, text = "Type in the colour"
                        "of the words, and not the word text!",
                                    font = ('Helvetica', 12))
instructions.pack()
scoreLabel = tkinter.Label(root, text = "Press enter to start",
                                    font = ('Helvetica', 12))
scoreLabel.pack()
timeLabel = tkinter.Label(root, text = "Time left: " +
            str(timeleft), font = ('Helvetica', 12))
timeLabel.pack()
label = tkinter.Label(root, font = ('Helvetica', 60))
label.pack()
e = tkinter.Entry(root)
root.bind('<Return>', startGame)
e.pack()
# set focus on the entry box
e.focus_set()
root.mainloop()
```

8. Design GUI and write a program to search string in Text using Python-Tkinter.



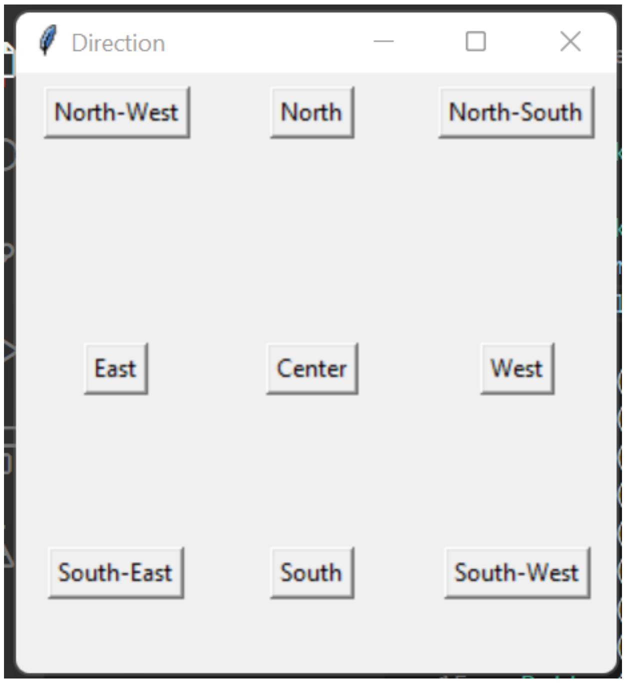
```
Write a paragrapah here
```



```
times
The Economic Times is an Indian English-language business-focused daily newspape
r. It is owned by The Times Group.
```

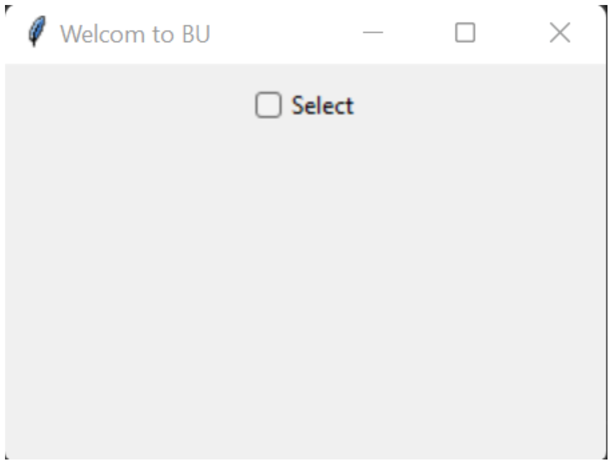
```
from tkinter import *
root = Tk()
root.title("Search String")
fram = Frame(root)
Label(fram,text='Text to find:').pack(side=LEFT)
edit = Entry(fram)
edit.pack(side=LEFT, fill=BOTH, expand=1)
edit.focus_set()
butt = Button(fram, text='Find')
butt.pack(side=RIGHT)
fram.pack(side=TOP)
text = Text(root)
text.insert('1.0','''Write a paragrapah here''')
text.pack(side=BOTTOM)
def find():
    text.tag_remove('found', '1.0', END)
    s = edit.get()
    if s:
        idx = '1.0'
        while 1:
            idx = text.search(s, idx, nocase=1,
                                stopindex=END)
            if not idx: break
            lastidx = '%s+%dc' % (idx, len(s))
            text.tag_add('found', idx, lastidx)
            idx = lastidx
    text.tag_config('found', foreground='red')
    edit.focus_set()
butt.config(command=find)
root.mainloop()
```

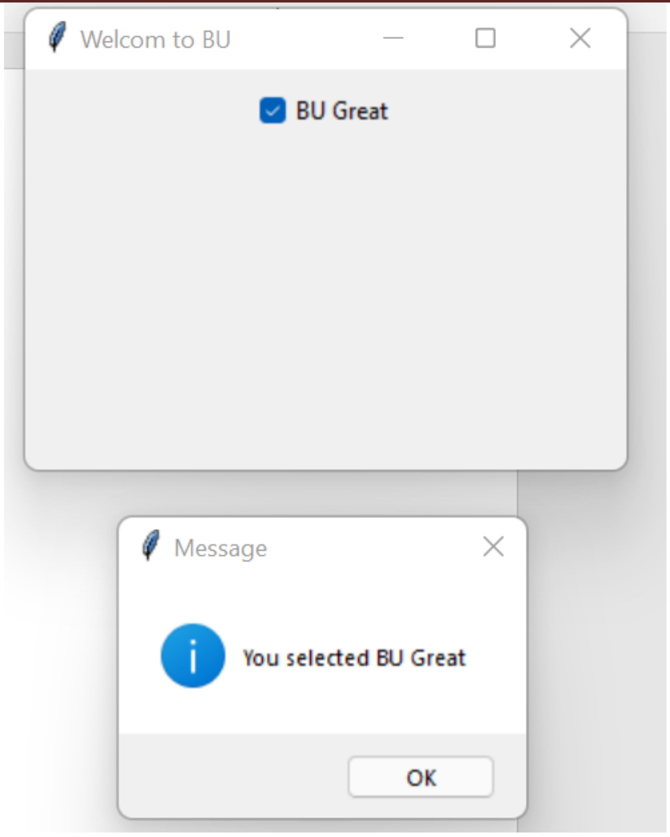
9. Create nine buttons and implement it on the screen. Each one is placed in the direction i.e., North, South, East, West, Center, North-West, North-South, South-East, South-West. Design following GUI using Python-Tkinter.



```
from tkinter import *

ws = Tk()
ws.geometry("300x300")
ws.title("Direction")

Button(ws, text="North-West").place(x=50, y=20, anchor="center")
Button(ws, text="North").place(x=148, y=20, anchor="center")
Button(ws, text="North-South").place(x=250, y=20, anchor="center")
Button(ws, text="East").place(x=50, y=148, anchor="center")
Button(ws, text="West").place(x=250, y=148, anchor="center")
Button(ws, text="South-East").place(x=50, y=250, anchor="center")
Button(ws, text="South").place(x=148, y=250, anchor="center")
Button(ws, text="South-West").place(x=250, y=250, anchor="center")
Button(ws, text="Center").place(x=148, y=148, anchor="center")

ws.mainloop()
```

10. Write a program dynamically change text of Check button using Python-tkinter, For Example,





You selected BU Great

```
from tkinter import *
from tkinter import ttk
from tkinter.messagebox import _show

root = Tk()
root.geometry('300x200')
root.title("Welcom to BU")
text1 = StringVar()
text1.set('Select')

def show(event):
    string = event.get()
    _show('Message', 'You selected ' + string)

chkbtn1 = ttk.Checkbutton(root, textvariable = text1, variable = text1,
                        offvalue = 'BU Good',
                        onvalue = 'BU Great',
                        command = lambda : show(text1))
chkbtn1.pack(side = TOP, pady = 10)

root.mainloop()
```