Graphical User Interface (GUI)

NEE2106 Computer Programming for Electrical Engineers

Prepared by: Dr. Rui Li (rui.li@vu.edu.au)

VICTORIA UNIVERSITY
MELBOURNE AUSTRALIA

< Contents >

< Terminal-Based Programs>

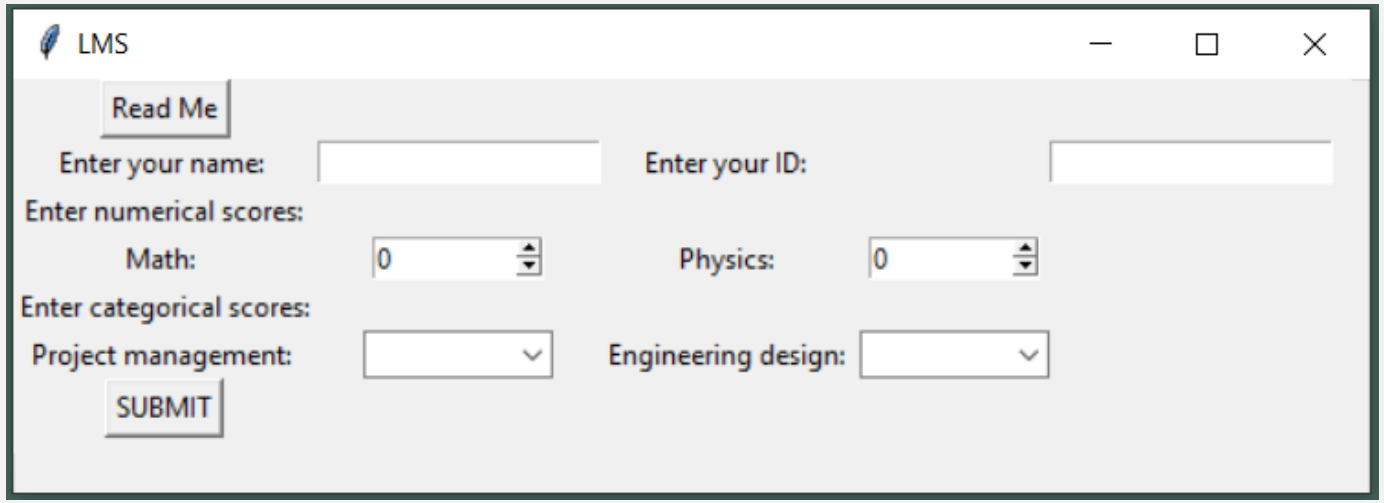
- Line programming or Console programming
- Text-based terminal, typically within a terminal or command prompt

```
name = input("Enter your name: ")
id = input("Enter your ID: ")
math = input("Enter your Math score (numerical): ")
physics = input("Enter your Pysics score (numerical): ")
pm = input ("Project Management score (categorical): ")
design = input ("Engineering Design score (categorical): ")
display("Student: "+name+", ID: "+id)
display("Math: "+math)
display("Physics: "+physics)
display("Project Management: "+pm)
display("Engineering Design: "+design)
```

```
Enter your name:  Rui
Enter your ID:  s123456
Enter your Math score (numerical):  95
Enter your Pysics score (numerical):  76
Project Management score (categorical):  C
Engineering Design score (categorical):  HD
'Student: Rui, ID: s123456'
'Math: 95'
'Physics: 76'
'Project Management: C'
'Engineering Design: HD'
```

< GUI-Based Programs>

- Graphic User Interface
- Allow users to interact with the program through windows, buttons, entries, sliders, and more



< Contents >

< Simple GUI-Based Programs>

Import Tkinter module

Create an GUI main window

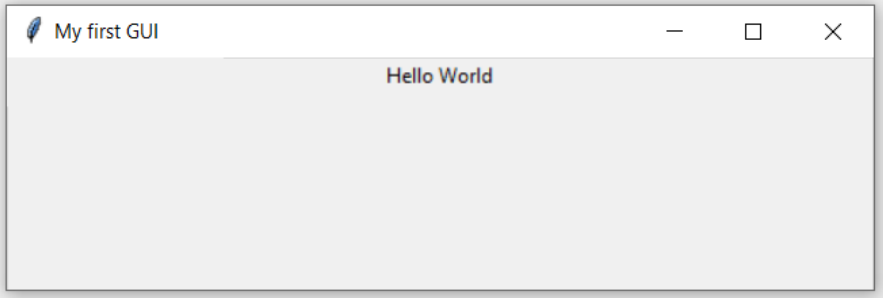
Add widgets

Enter the main event loop

```python
import tkinter
window = tkinter.Tk()
window.title("My first GUI") # rename the GUI window
label = tkinter.Label(window,text="Hello World") # create a label
label.pack() # show the object in the window
window.mainloop()
```

OR

```python
# imports all names from the module & no need to add "tkinter" prefix when using
from tkinter import *
window = Tk()
window.title("My first GUI") # rename the GUI window
label = Label(window,text="Hello World") # create a label
label.pack() # show the object in the window
window.mainloop()
```



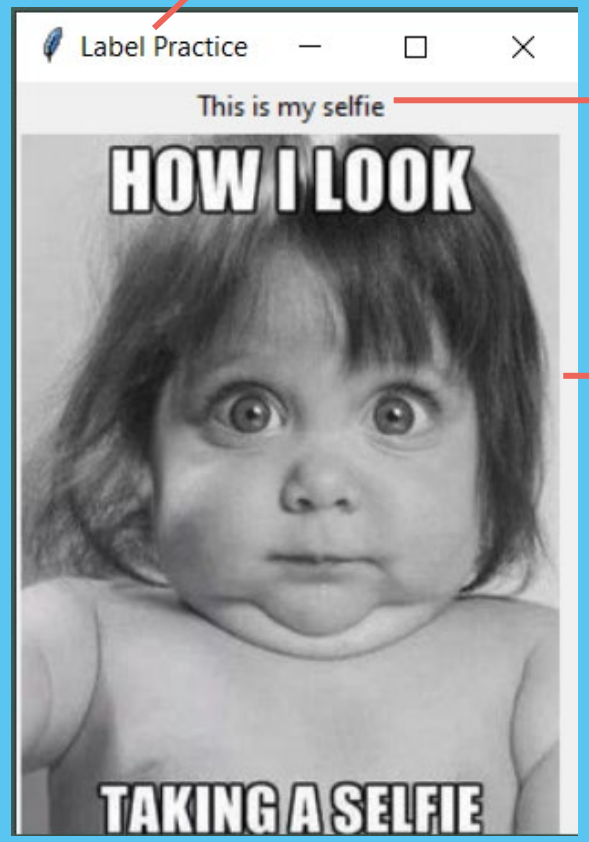
< Widget – Label >

- Display static text or images
- Provide information, instructions, or headings within a GUI application

```python
from tkinter import *
from PIL import Image, ImageTk  # Import Image and ImageTk from PIL library

# Create a Tkinter window
window = Tk()
window.title("A Calculator") # rename the GUI window\

# Load the text
label1 = Label(window,text="This program calculates the area of a circle \n based on the given radius.",font=("Cabibri",10))
label1.pack() # Pack the label widget to display it in the window

# Load the image
image = Image.open("AllCircle.png")  # Replace "example_image.jpg" with your image file path
image = image.resize((300, 200))  # Resize the image as needed
tk_image = ImageTk.PhotoImage(image) # Convert the image to Tkinter-compatible format
label2 = Label(window, image=tk_image) # Create a Label widget to display the image
label2.pack()

# Run the Tkinter event loop
window.mainloop()
```

< Exercise>

Write a Python GUI to show the following interface.




```python
#%% widget - label
from tkinter import *
from PIL import Image, ImageTk

# Create a Tkinter window
window = Tk()
window.title("Label Practice") # rename the GUI window

# text label
label1 = Label(window,text="This is my selfie")
label1.grid(row = 0, column = 0)

# image label
image = Image.open("C:/Users/e5107499/Desktop/NEE2106/2024/selfie.jpg")
label_image = ImageTk.PhotoImage(image)    # Convert the image to Tkinter-compatible format
label2 = Label(window, image=label_image)  # Create a Label widget to display the image
label2.grid(row = 1, column = 0)

# Start the main loop
window.mainloop()
```

< Widget – Button >

- Allow users to interact with by clicking or pressing it
- Buttons are common elements in GUI and are used to trigger actions

Call back function

```
# button
def call_back(txt): # when butoon is clicked, execute this function
    label3 = Label(window,text=txt) # define and show a new txt label
    label3.pack()
bt = Button(window,text="Press to Calculate",bg="green",fg="yellow",command=lambda:call_back("just clicked!"))
bt.pack()
```

Q: what happens without “lambda”?

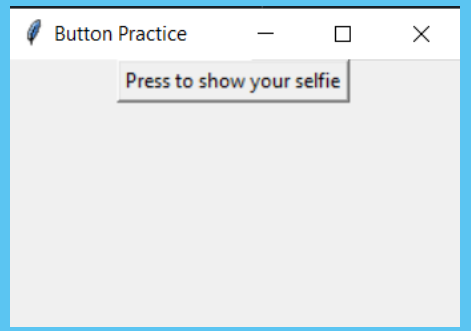
Background colour

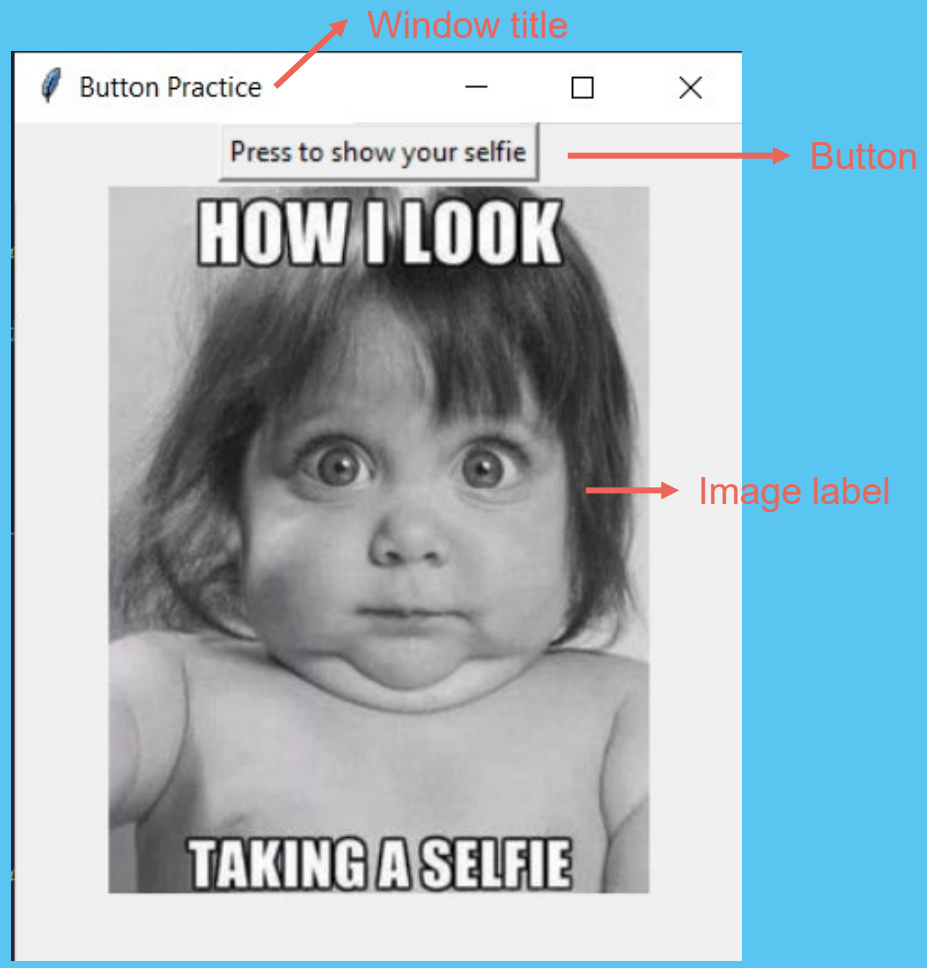
Font colour

The command to execute after clicking

< Exercise>

Write a Python GUI to show your selfie after pressing the button.






```python
#%% widget - button
from tkinter import *
from PIL import Image, ImageTk

# Create a Tkinter window
window = Tk()
window.title("Button Practice") # rename the GUI window

# load the image to tkinter image frame
image = Image.open("C:/Users/e5107499/Desktop/NEE2106/2024/selfie.jpg")
label_image = ImageTk.PhotoImage(image)    # Convert the image to Tkinter-compatible format

# button call back
def call_back():
    label2 = Label(window, image=label_image) # display the image
    label2.pack()

# define the button
bt = Button(window,text="Press to show your selfie", command=call_back)
bt.pack()

# Start the main loop
window.mainloop()
```

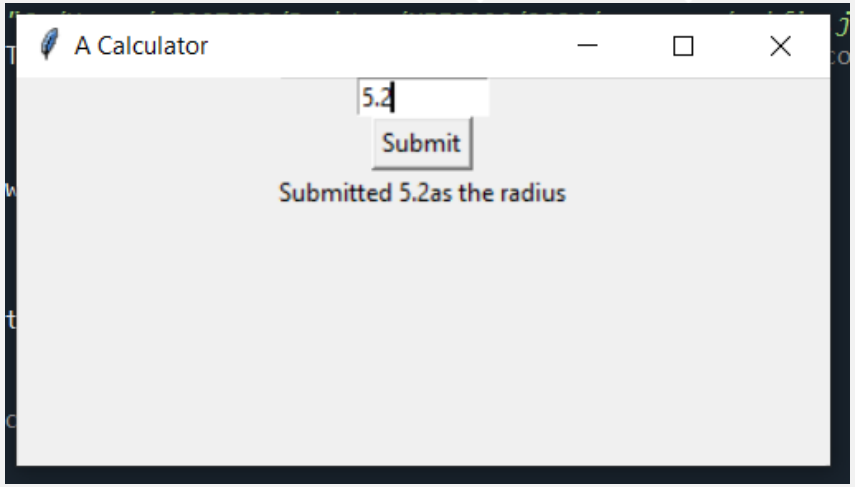
< Widget – Entry >

- Allow users to type inputs in the field, i.e. radius

```python
#%% entry
from tkinter import *
from PIL import Image, ImageTk  # Import Image and ImageTk from PIL library

# Create a Tkinter window
window = Tk()
# window = Toplevel()
window.title("A Calculator") # rename the GUI window

# entry
msg = Entry(window,width=10) # define a Entry field
msg.pack()

# button call back
def call_back(txt):
    label2 = Label(window, text=txt) # display the image
    label2.pack()

# button to submit the entry
submit_button = Button(window, text="Submit", command=lambda:call_back(txt="Submitted "+msg.get()+"as the radius"))
submit_button.pack()

# Start the main loop
window.mainloop()
```

Get the msg typed in the Entry

Create a button widget to submit the input



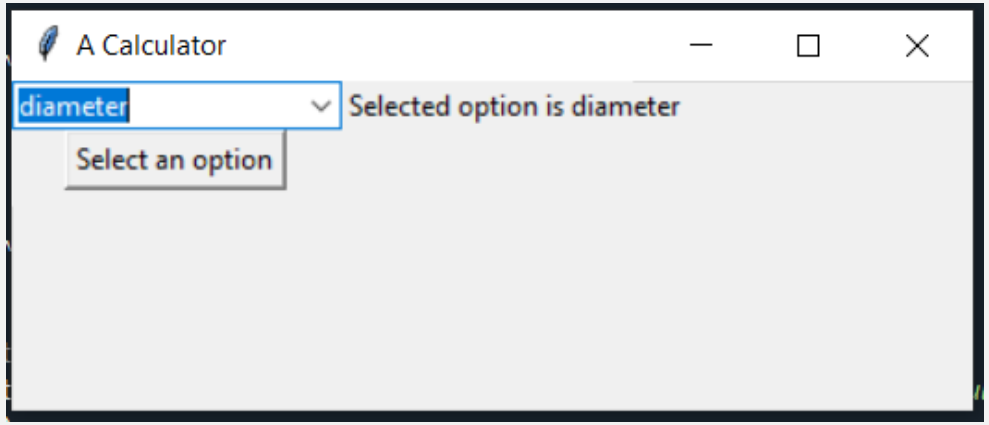
< Widget – ComboBox/ Dropdown List >

- Allows users to select an item from a list of pre-defined options

```python
#%% ComboBox (dropdown list)
from tkinter import ttk
from tkinter import *

# Create a Tkinter window
window = Tk()
# window = Toplevel()
window.title("A Calculator") # rename the GUI window

# drop-down list
box = ttk.Combobox(window,values=["radius","diameter"]) # 2 options in the list
box.current(0) # default selection
box.grid(row = 0, column = 0) # specify the position

# create a Label to display selected result
result_label = Label(window,text="Waiting for selection.") # default display
result_label.grid(row = 0, column = 1) # specify the position

# button to submit the selection
submit_button2 = Button(window, text="Select an option",
                        command=lambda:result_label.config(text="Selected option is "+box.get()))
submit_button2.grid(row = 1, column = 0) # specify the position

# Start the main loop
window.mainloop()
```

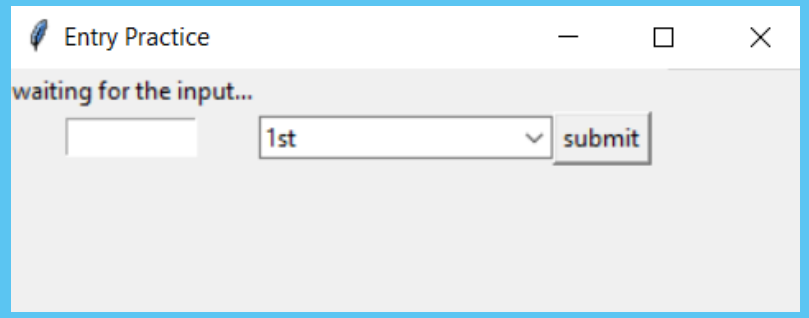
Options in the list

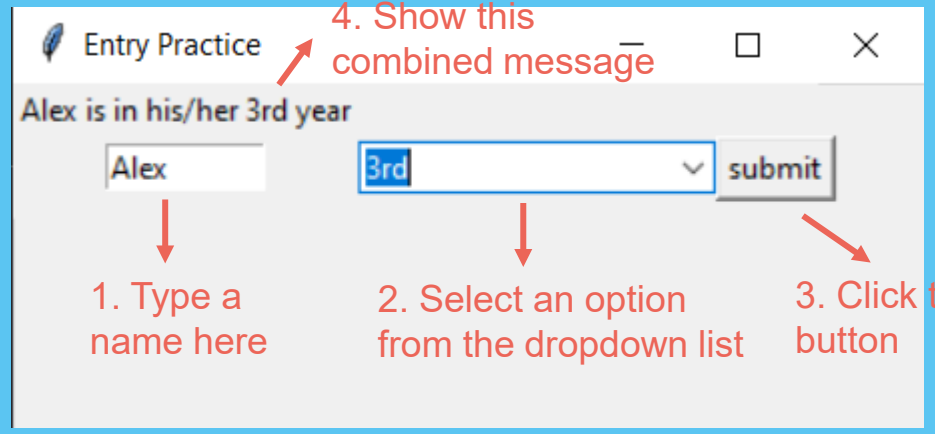
Get the selected string



< Exercise>

Write a Python GUI to display the following text message after pressing the button.






```python
#%% widget - entry, combo box
from tkinter import *
from tkinter import ttk

# Create a Tkinter window
window = Tk()
window.title("Entry Practice") # rename the GUI window

# text label
msg = Label(window,text="waiting for the input...")
msg.grid(row=0,column=0)

# enter a name
name = Entry(window, width=10)
name.grid(row=1,column=0)

# select a yr level
yr = ["1st","2nd","3rd","4th"]
box = ttk.Combobox(window,values=yr)
box.current(0)
box.grid(row=1,column=1)

# click on a button to update the label
def call_back():
    msg.config(text=name.get()+" is in his/her "+box.get()+" year")
bt = Button(window,text="submit", command=call_back)
bt.grid(row=1,column=2)


# Start the main loop
window.mainloop()
```

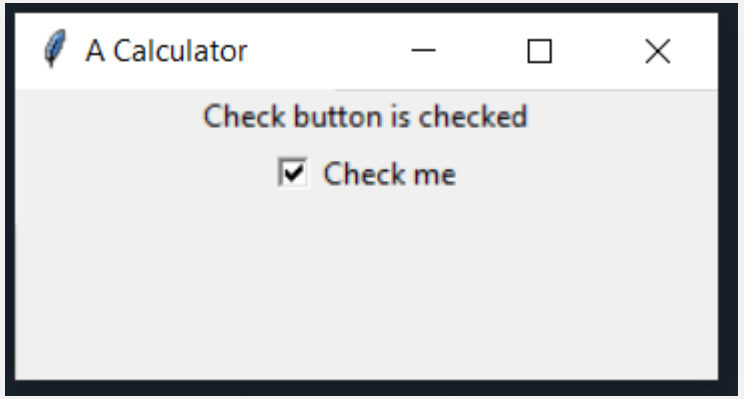
< Widget – Check Button>

- Allows users to tick a Boolean option (T/F)

What to do when checked/ unchecked

```
# check button
def update_label():     # Function to execute based on check button state
    if flag.get() == 1: # when button is check
        label3.config(text="Check button is checked")
    else:               # when unchecked
        label3.config(text="Check button is unchecked")

# create a Label to display the check button state
label3 = tk.Label(window, text="waiting for the check")
label3.pack()

# Create a Checkbutton widget
flag = IntVar()         # "flag" stores an ingeter variable (1 or 0)
check_button = Checkbutton(window, text="Check me", variable=flag, command=update_label)
check_button.pack()
```

Store checked/ uncheck as a Boolean T/F



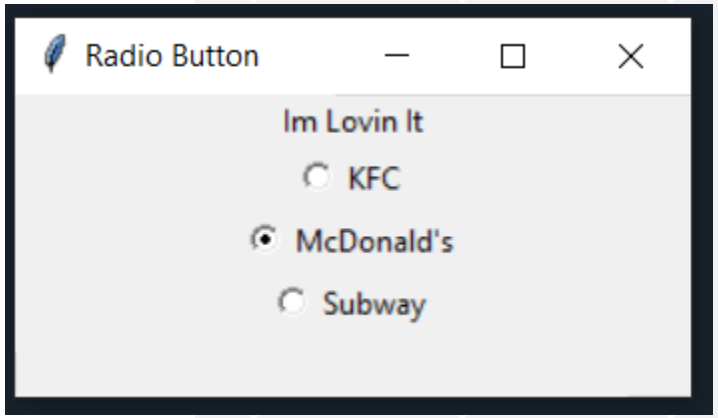
< Widget – Radio Button >

- Allows users to select one option from a group of mutually exclusive options

What to do when each option is selected

Define the radio buttons

```
# radio button
def update_radio():  # Function to update the label text based on radio button selection
    selected_option = food_option.get() # get the value in "food_option" variable
    if selected_option == 1:
        label4.config(text="Who says KFC?")
    elif selected_option == 2:
        label4.config(text="Im Lovin It")
    elif selected_option == 3:
        label4.config(text="Eat Fresh")

food_option = IntVar() # integer variable that stores the selection (i.e. 1,2,3)
# This command associates all the options with a common variable

# Label to display the selected option
label4 = Label(window, text="No option selected")
label4.pack()

# Create radio buttons and associate them with the variable
# "value" is passed onto the common variable "food_option", which will be further defined in if-else statement
radio_button1 = Radiobutton(window, text="KFC", variable=food_option, value=1, command=update_radio)
radio_button1.pack()

radio_button2 = Radiobutton(window, text="McDonald\'s", variable=food_option, value=2, command=update_radio)
radio_button2.pack()

radio_button3 = Radiobutton(window, text="Subway", variable=food_option, value=3, command=update_radio)
radio_button3.pack()
```



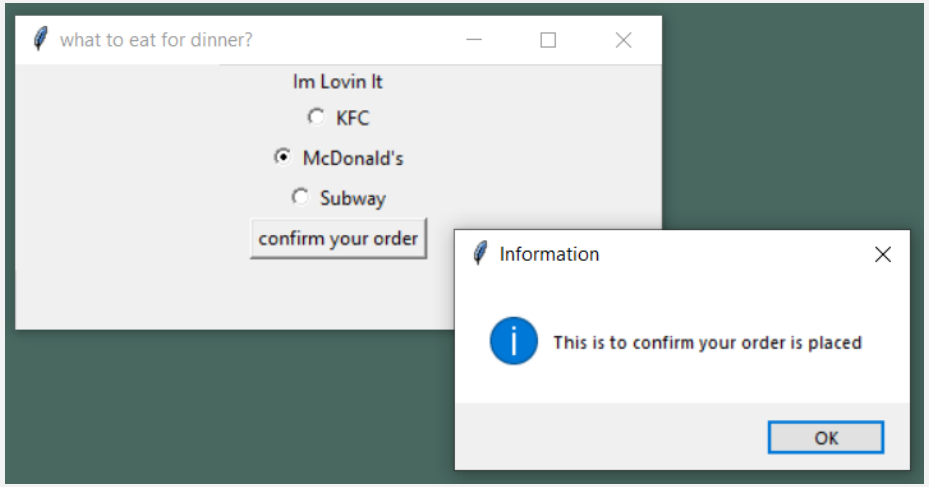
< Widget – Dialog Box>

- A pre-defined window to provide information, warning, error, or ask for confirmation.

Provide information

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

# information on the dialog box
def show_info():
    messagebox.showinfo("Information", "This is to confirm your order is placed")

# Button to trigger the dialog box
info_button = Button(window, text="confirm your order", command=show_info)
info_button.pack()
```

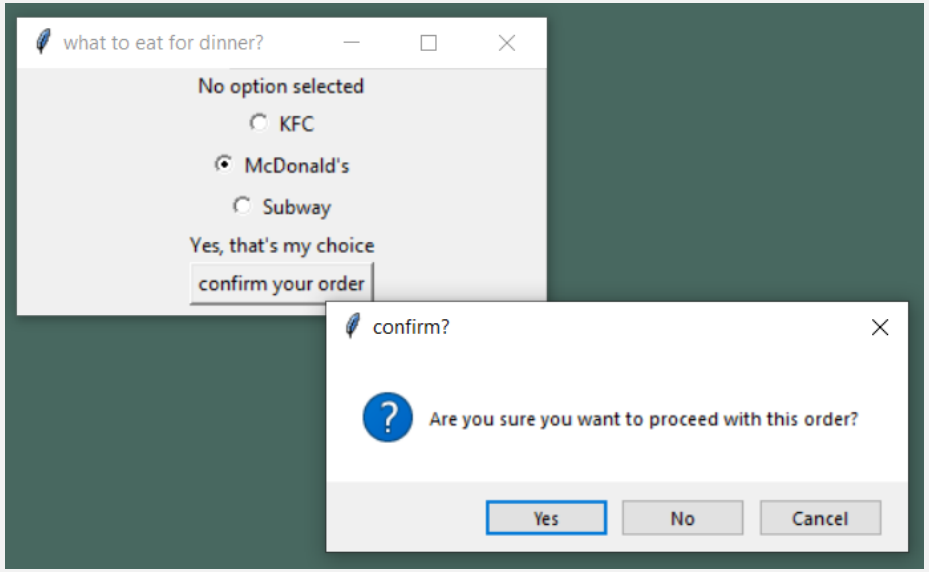
You can also **showwarning()** or **showerror()**



Ask for confirmation

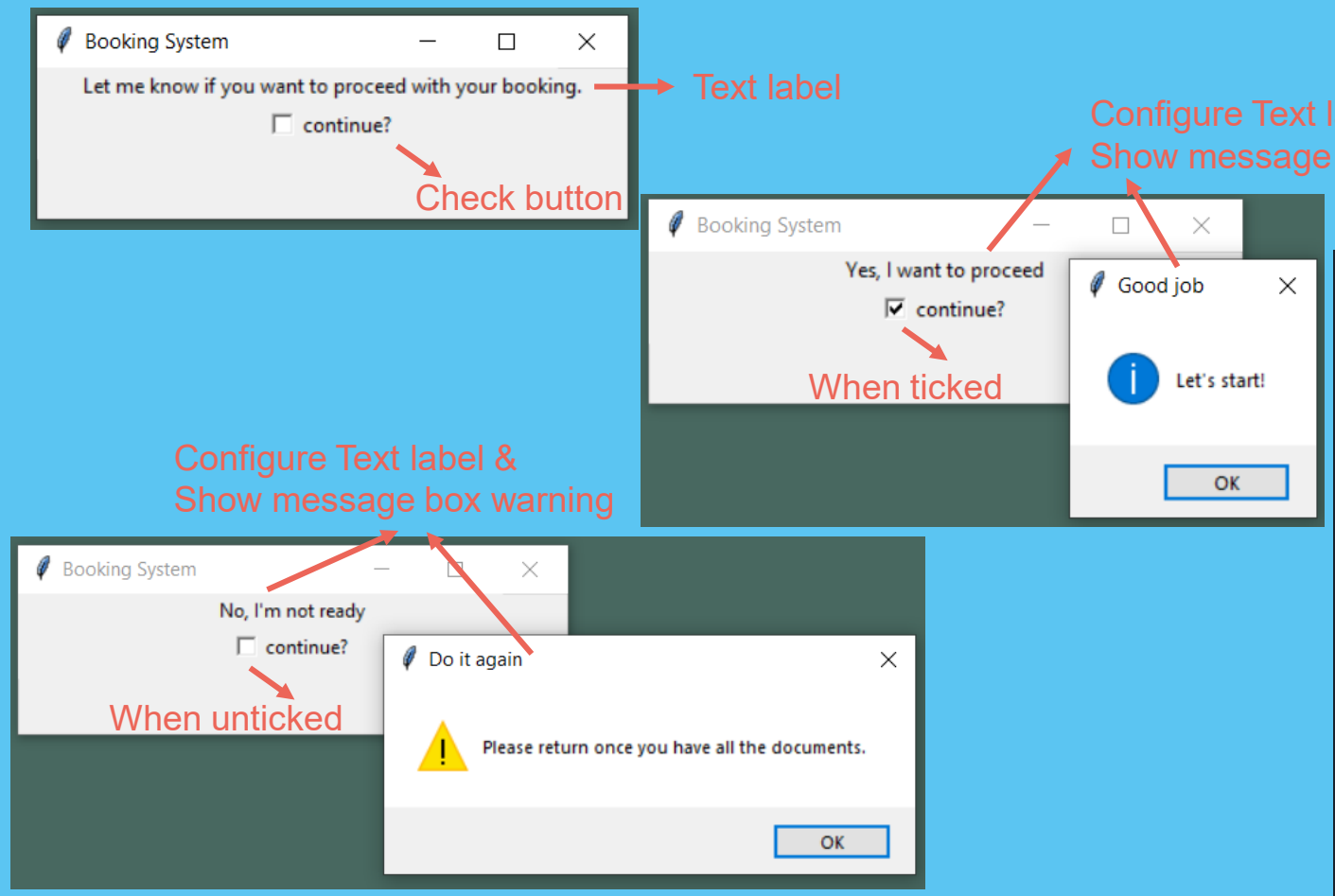
```python
from tkinter import *
from tkinter import messagebox

# ask a question in the dialog box
def show_question():
    ans = messagebox.askyesnocancel("confirm?", "Are you sure you want to proceed with this order?")
    if ans == 1:
        label4.config(text="Yes, that's my choice")
    elif ans == 0:
        label4.config(text="No, I decide to have something healthier")
    else:
        label4.config(text="Order cancelled")

# Label to display the selected option
label4 = Label(window, text="no order yet")
label4.pack()

# Button to trigger the dialog box
question_button = Button(window, text="confirm your order", command=show_question)
question_button.pack()
```

You can also **askyesno()** or **askokcancel()**



< Exercise>

Write a Python GUI to display the following messages based on the check button values.

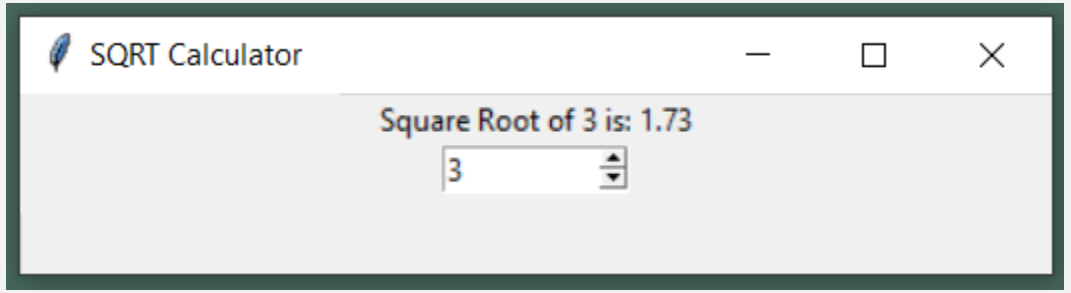



```python
#%% widget - check box, dialog box
from tkinter import *
from tkinter import messagebox

# Create a Tkinter window
window = Tk()
window.title("Booking System") # rename the GUI window

def update_check():
    if flag.get()==1:
        msg.config(text="Yes, I want to proceed")
        messagebox.showinfo("Good job","Let's start!")
    else:
        msg.config(text="No, I'm not ready")
        messagebox.showwarning("Do it again","Please return once you have all the documents.")

# create a text label
msg = Label(window, text="Let me know if you want to proceed with your booking.")
msg.pack()

# Create a Check button
flag = IntVar() # "flag" stores an ingeter variable (1 or 0)
check_button = Checkbutton(window, text="continue?", variable=flag, command=update_check)
check_button.pack()

# Start the main loop
window.mainloop()
```

< Widget – Spin Box>

- Allow users to input numeric values by either typing in a field or spinning up/ down arrows

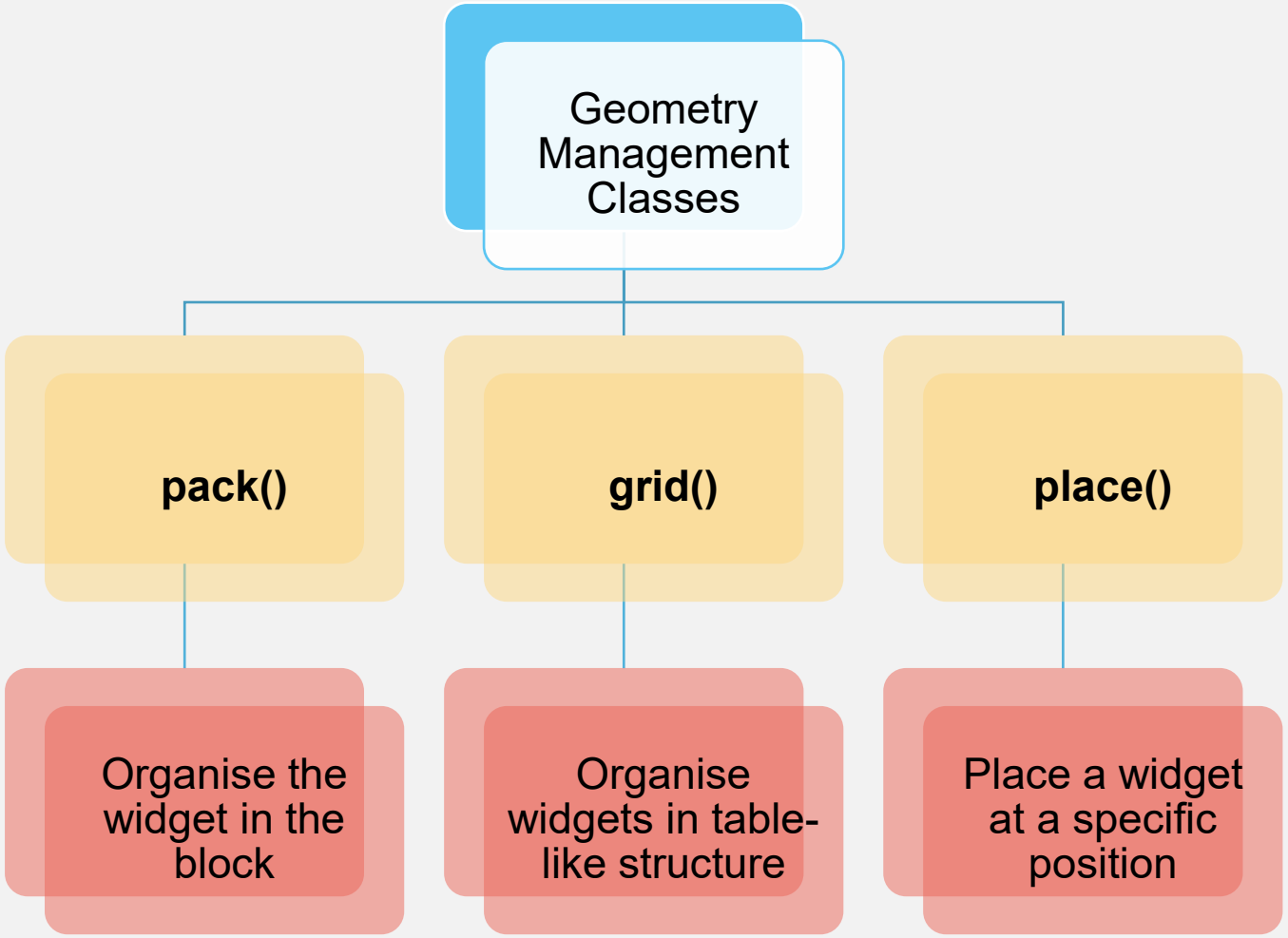
```python
# Function to handle spin box value change
def spin_value():
    value = int(spinbox.get()) # get value in the spin box and convert to integer
    label5.config(text=f"Square Root of {value} is: {math.sqrt(value):.2f}") # calculate and edit the label
    # use f string (formatted string) to conveniently display a string with dynamic contents

# Label to display the result
label5 = Label(window, text="no value yet")
label5.pack()

# Create a Spinbox widget
spinbox = Spinbox(window, from_=0, to=100, increment=1, width=10, command=spin_value)
spinbox.pack()
```



< Contents >

- Terminal vs. GUI-based Programs
- Simple GUI-based programs
- Geometry Management

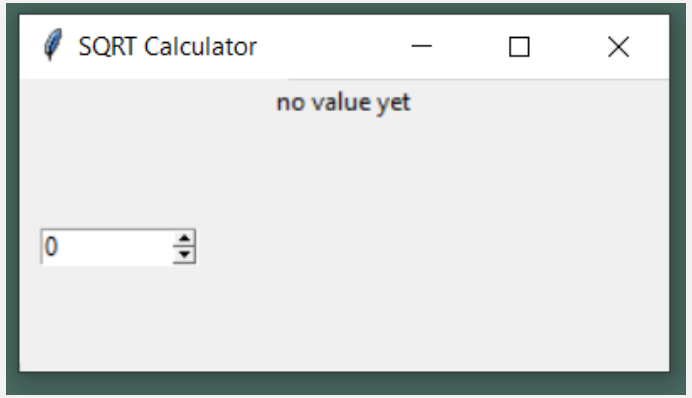
< Widget Geometry>



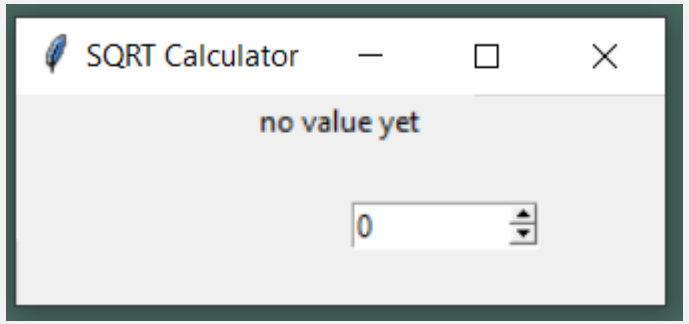
< pack() >

- Organise the widget within the window
- “padx” and “pady” specify the padding (horizontal and vertical margin around the widget), unit: pixel
- “side” specify the alignment: LEFT, RIGHT, TOP, BOTTOM

```
spinbox.pack(padx=10,pady=50,side=LEFT)
```



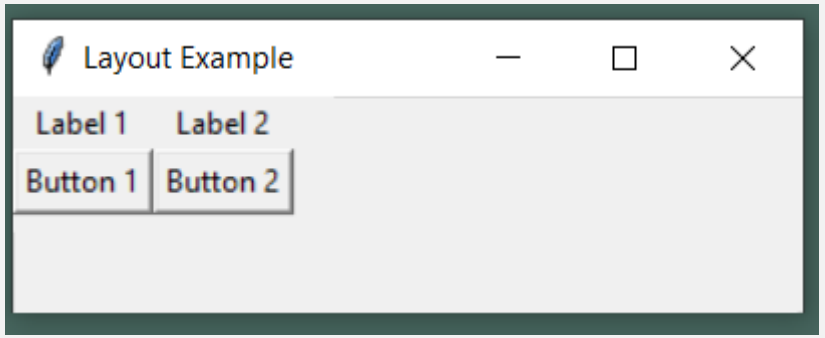

```
spinbox.pack(padx=50,pady=5,side=RIGHT)
```



< grid() >

- More flexible and powerful way to organise widgets
- Define the position using “row = xx, column = xx” in a grid-like table format

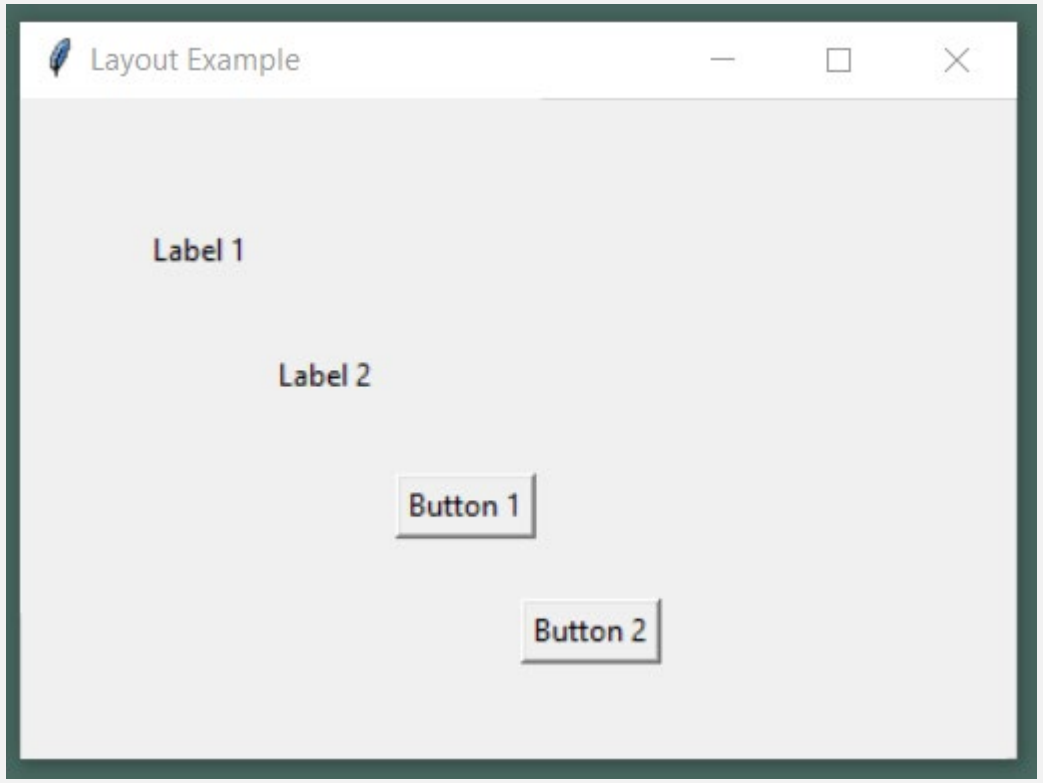
```
# Create widgets
label1 = Label(window, text="Label 1")
label2 = Label(window, text="Label 2")
button1 = Button(window, text="Button 1")
button2 = Button(window, text="Button 2")

# Place widgets using grid()
label1.grid(row=0, column=0)
label2.grid(row=0, column=1)
button1.grid(row=1, column=0)
button2.grid(row=1, column=1)
```



< place() >

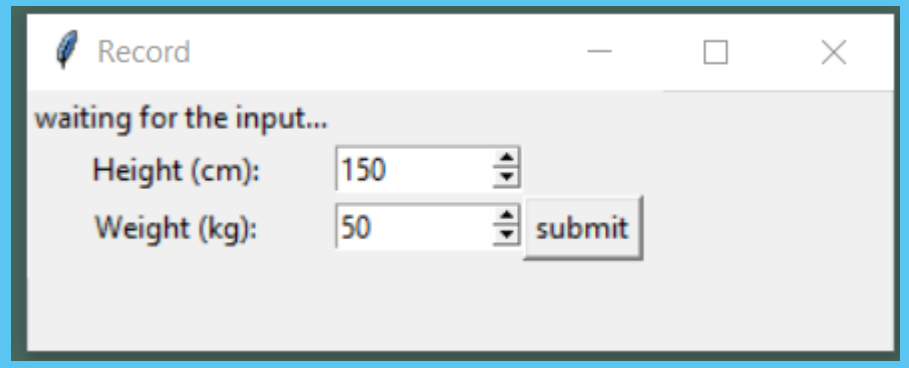
- More precise control over the position
- Define a position using exact coordinates

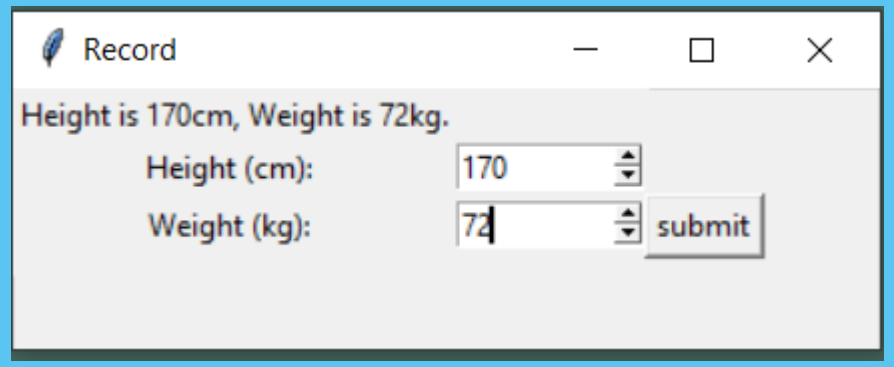
```
# Place widgets using place()
label1.place(x=50, y=50)
label2.place(x=100, y=100)
button1.place(x=150, y=150)
button2.place(x=200, y=200)
```



< Exercise>

Write a Python GUI to display the following messages based on input values.






```python
#%% widget - spin box
from tkinter import *

# Create a Tkinter window
window = Tk()
window.title("Record") # rename the GUI window

# text label
msg = Label(window,text="waiting for the input...")
msg.grid(row=0,column=0)

# spin boxes
height_label = Label(window,text="Height (cm): ")
height_label.grid(row=1,column=0)
height = Spinbox(window, from_=150, to=200, increment=1, width=10)
height.grid(row=1,column=1)

weight_label = Label(window,text="Weight (kg): ")
weight_label.grid(row=2,column=0)
weight = Spinbox(window, from_=50, to=80, increment=1, width=10)
weight.grid(row=2,column=1)

# click a button to display the msg correspondingly
def call_back():
    msg.config(text="Height is "+height.get()+"cm, Weight is "+weight.get()+"kg.")
bt = Button(window,text="submit", command=call_back)
bt.grid(row=2,column=2)

# Start the main loop
window.mainloop()
```