Practical Learning 9
Python GUI with tkinter

Radio Button

A radio button, sometimes called option button, is a graphical user interface element of Tkinter, which allows the user to choose (exactly) one of a predefined set of options. Radio buttons can contain text or images. The button can only display text in a single font. A Python function or method can be associated with a radio button. This function or method will be called, if you press this radio button.

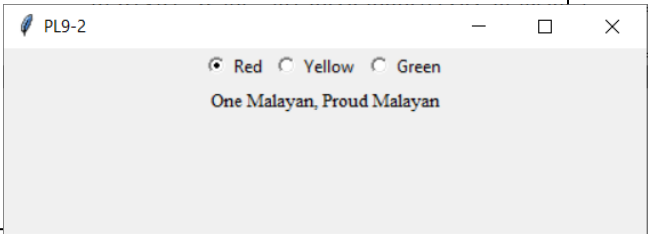
Each group of Radio button widgets has to be associated with the same variable. Pushing a button changes the value of this variable to a predefined certain value.

1. Create a python file named PL9-2_LastnameFirstname.
2. Create a root window.

```
from tkinter import *
class Window(Frame):
    def __init__(self,master=None):
        Frame.__init__(self,master)
        self.master = master


root = Tk()
root.geometry("450x130")
app = Window(root)
root.mainloop()
```

3. Add label and radioButton widgets. In here, were going to add 3 radiobuttons and 1 Message with “OneMalayan, Proud Malayan” text.

```
from tkinter import *
class Window(Frame):
    def __init__(self,master=None):
        Frame.__init__(self,master)
        self.master = master
        self.init_window() #contains the widgets

    def init_window(self):
        #shared variable of the radiobuttons
        self.v = StringVar()
        self.v.set("red") #set default value

        #create Message widget
        self.msgResult = Message(self,text="One Malayan, Proud Malayan",width=250,font="times 10")

        #create radiobuttons
        optRed = Radiobutton(self,text="Red",variable=self.v,value="red")
        optYellow = Radiobutton(self,text="Yellow",variable=self.v,value="yellow")
        optGreen  = Radiobutton(self,text="Green",variable=self.v,value="green")

        #organize widgets
        optRed.grid(row=0,column=1)
        optYellow.grid(row=0,column=2)
        optGreen.grid(row=0,column=3)
        self.msgResult.grid(row=2,column=1,columnspan=3)

        #place widgets on window
        self.pack()


root = Tk()
root.geometry("450x130")
app = Window(root)
root.mainloop()
```



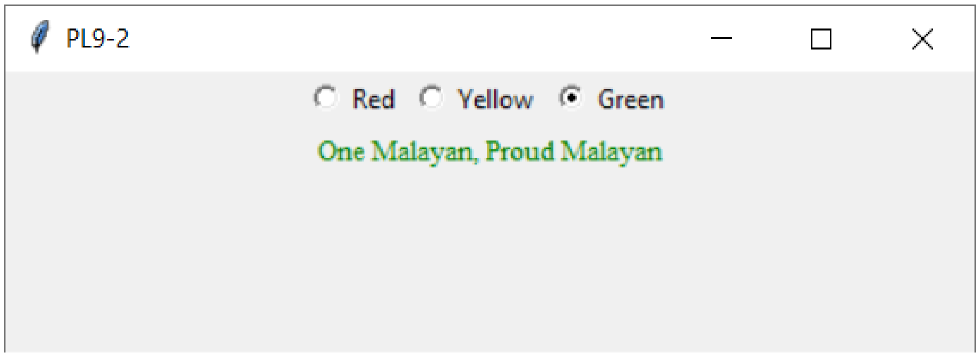
One Malayan, Proud Malayan

4. Add event when a radio button is clicked, change the color of the Message text
   Create a function to change the color using ***.config()*** method. The .get() method will get the value of the shared variable v of the radiobutton.

```
 def changeColor(self):
     self.msgResult.config(fg=str(self.v.get()))
```

   Call the function using the command option:

```
 optRed = Radiobutton(self,text="Red",variable=self.v,value="red", command=self.changeColor)
 optYellow = Radiobutton(self,text="Yellow",variable=self.v,value="yellow",command=self.changeColor)
 optGreen  = Radiobutton(self,text="Green",variable=self.v,value="green",command=self.changeColor)
```



One Malayan, Proud Malayan

CheckButton

The Checkbutton widget is used to display a number of options to a user as toggle buttons. The user can then select one or more options by clicking the button corresponding to each option.

5. Create variable to store checkbuttons value

```
     #variables for checkbuttons
     self.b = StringVar()
     self.i = StringVar()
     self.l = StringVar()
```

6. Create Checkbutton widgets.

```
#create checkbuttons
chkBold = Checkbutton(self, text="BOLD", font="times 12 bold",variable=self.b,onvalue="bold",offvalue="")
chkItalic = Checkbutton(self, text="ITALIC", font="times 12 italic",variable=self.i,onvalue="italic",offvalue="")
chkLine = Checkbutton(self,text="UNDELINE",font="times 12 underline",variable=self.l,onvalue="underline",offvalue="")
```

- **Offvalue**
  Normally, a checkbutton's associated control variable will be set to 0 when it is cleared (off). You can supply an alternate value for the off state by setting offvalue to that value.
- **Onvalue**
  Normally, a checkbutton's associated control variable will be set to 1 when it is set (on). You can supply an alternate value for the on state by setting onvalue to that value.

7. Place checkbuttons on window

```
  chkBold.grid(row=1,column=1)
  chkItalic.grid(row=1,column=2)
  chkLine.grid(row=1,column=3)
```

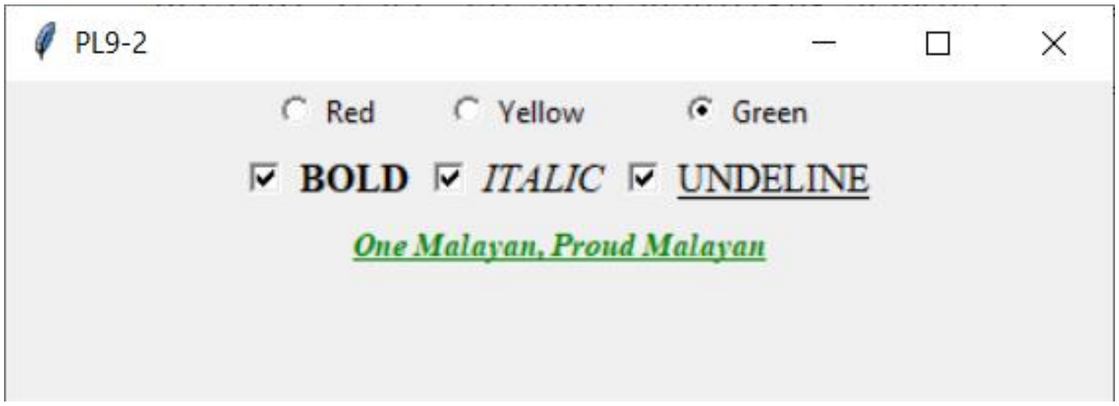
8. Add event when checkbutton is click.
   Create function the will change the font style of the Message text based on the selected checkbox.

```
def changeStyle(self):
    st = "Times 10 {0} {1} {2}".format(str(self.b.get()),str(self.i.get()),str(self.l.get()))
    self.msgResult.config(font=st)
```

9. Use command option to call changeStyle function when checkButton is ticked:

```
#create checkbuttons
chkBold = Checkbutton(self, text="BOLD", font="times 12 bold",variable=self.b,onvalue="bold",offvalue="",command=self.changeStyle)
chkItalic = Checkbutton(self, text="ITALIC", font="times 12 italic",variable=self.i,onvalue="italic",offvalue="",command=self.changeStyle)
chkLine = Checkbutton(self,text="UNDELINE",font="times 12 underline",variable=self.l,onvalue="underline",offvalue="",command=self.changeStyle)
```

Output:



Spinbox

The Spinbox widget allows the user to select values from a given set. The values may be a range of numbers, or a fixed set of strings.

10. Create spinbox widget

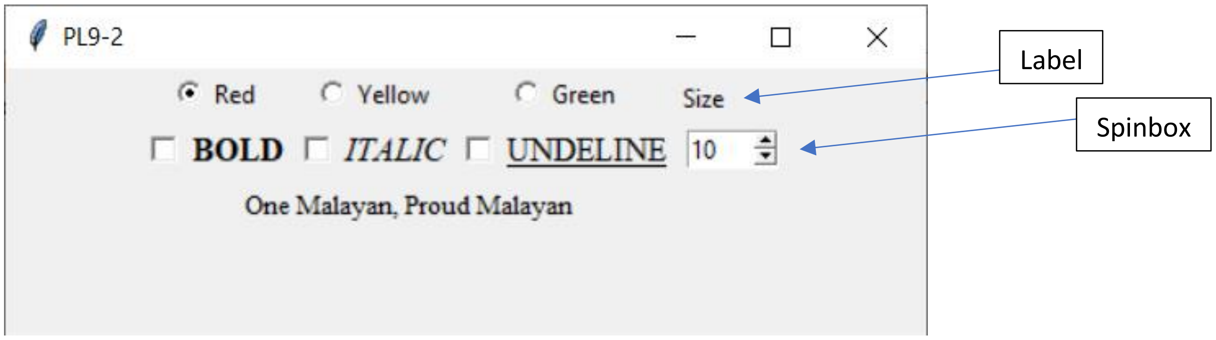
```
#create spinbox
self.spnSize = Spinbox(self,from_=10, to=20,width=5)
```

- **from_** Use this option in combination with the **to** option (described below) to constrain the values to a numeric range.
- **to** This option specifies the upper limit of a range values.

11. Create label and place on window

```
#create label and place on window
Label(self,text="Size").grid(row=0,column=4,sticky="sw")
self.spnSize.grid(row=1,column=4,sticky="nw",pady=5,padx=5)
```

Output:



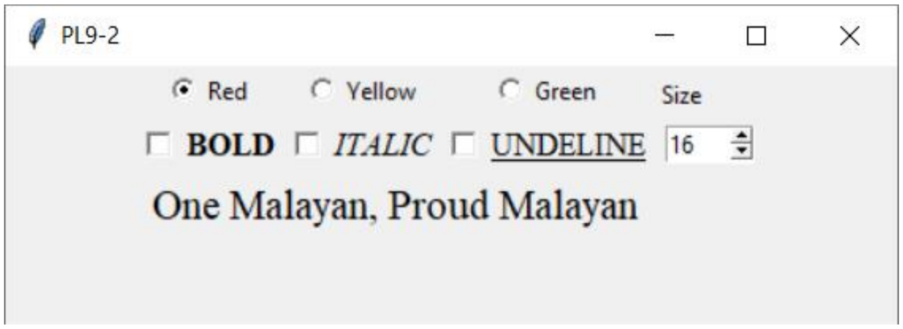
12. Add event when spinbox value increases or decreases. Let’s modify our changeStyle function.

```
def changeStyle(self):
   n=self.spnSize.get() #get the value of the spinbox
   st = "Times {0} {1} {2} {3}".format(str(n),str(self.b.get()),str(self.i.get()),str(self.l.get()))
   self.msgResult.config(font=st)
```

13. Call changeStyle function using command option.

```
self.spnSize = Spinbox(self,from_=10, to=20,width=5,command=self.changeStyle)
```

Output:


PL9-2

● Red ○ Yellow ○ Green Size

☐ BOLD ☐ *ITALIC* ☐ UNDELINE 16

One Malayan, Proud Malayan

ListBox

The Listbox widget is used to display a list of items from which a user can select a number of items.

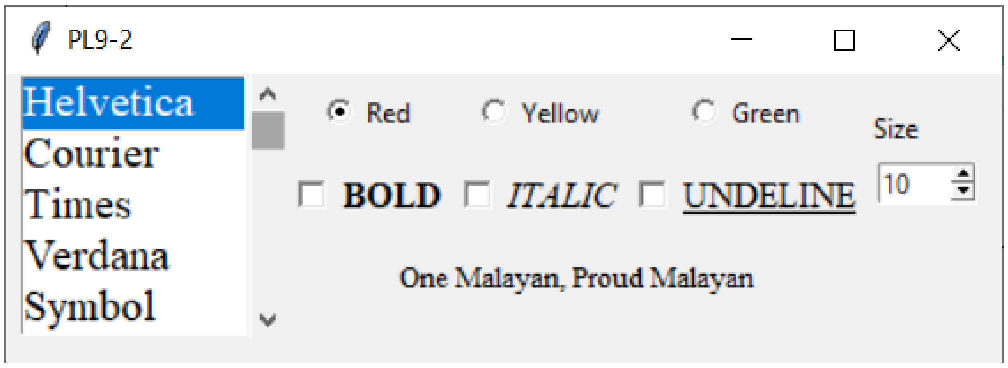
Scrollbar

This widget provides a slide controller that is used to implement vertical scrolled widgets, such as Listbox, Text and Canvas.

14. Let's add listbox with scrollbar on our window.

```
#create listbox with scrollbar in a frame
frmlist = Frame(self)
self.lstStyle = Listbox(frmlist,font="Times 15",width=10,height=5)
self.lstStyle.pack(side=LEFT)
scrollbar = Scrollbar(frmlist,orient=VERTICAL)
scrollbar.pack(side=RIGHT,fill=Y)

#add items in the listbox using insert()
style = ['Helvetica', 'Courier', 'Times', 'Verdana','Symbol','System']
i=0
for x in style:
    self.lstStyle.insert(i,x)
    i+=1

self.lstStyle.selection_set(0) #set first item in the list as default selected

#place widgets  on window
frmlist.grid(row=0,rowspan=4,column=0)
```

Output:


PL9-2

Helvetica
Courier
Times
Verdana
Symbol

● Red ○ Yellow ○ Green Size

☐ BOLD ☐ *ITALIC* ☐ UNDELINE 10

One Malayan, Proud Malayan

15. Let’s connect the scrollbar with the listbox. So that, when we can scroll items in our listbox up and down.

```
scrollbar.config(command=self.lstStyle.yview)
```

After adding the command, the scrollbar will execute the built-in function **yview** of the lstStyle listbox.

16. Now, let’s add an event to the listbox, we want to change the font face of the Message text based on the selected value in the listbox. Since listbox, do not have a command option, we need to bind the event into our widget. The <<ListboxSelect>> event triggers the changeStyle function.

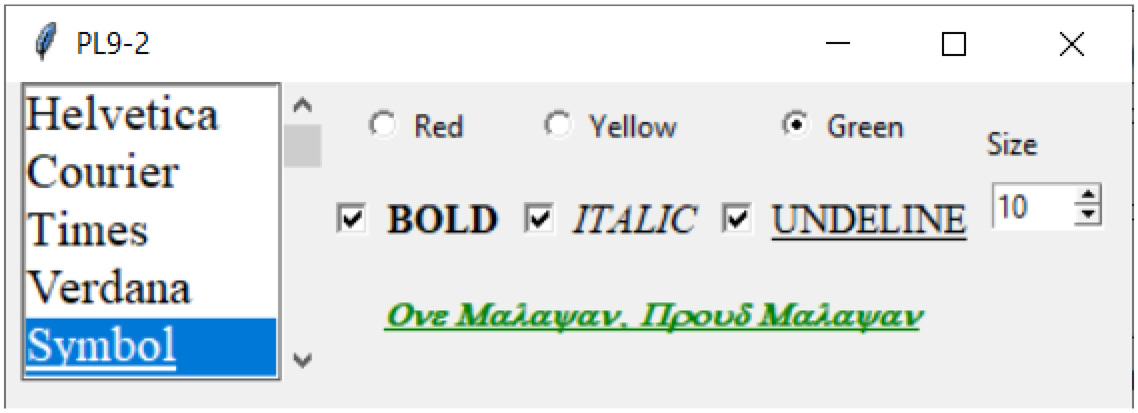
```
self.lstStyle.bind('<<ListboxSelect>>',self.changeStyle)
```

17. Let’s modify our changeStyle function. Bind function to event passes two arguments, so we need to add parameter to our changeStyle function.

```
def changeStyle(self, *args):
    n=self.spnSize.get() #get the value of the spinbox
    stylo =self.lstStyle.get(self.lstStyle.curselection()[0])
    st = "{0} {1} {2} {3} {4}".format(str(stylo),str(n),str(self.b.get()),str(self.i.get()),str(self.l.get()))
    self.msgResult.config(font=st)
```

The ***curselection()*** method returns the index of the selected item.
The ***get()*** method returns the value of the selected item given the index.

Output:



PL9-2

Helvetica
Courier
Times
Verdana
Symbol

Red
Yellow
[✓ Green]
Size
10

[✓] BOLD [✓] ITALIC [✓] UNDELINE

Ovε Μαλαψαν, Προυδ Μαλαψαν

18. Save and upload your work.