杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

第8章 图形化界面设计

8.1 GUI概述

8.2 tkinter常用控件

8.3 控件布局

8.4 事件响应

- 理解tkinter常用控件的功能。
- 理解控件的布局方法。
- 掌握tkinter控件的共同属性和特有属性。
- 掌握几种常用控件在程序设计中的设置和取值方法。
- 学会用户事件响应与自定义函数绑定。

GUI概述

- 图形化用户界面（Graphic User Interface，GUI）

GUI是用户与应用程序之间进行交互控制、并相互传递数据和信息的图形界面。用户通过GUI可以方便、快捷地进行复杂的程序控制。

- Python支持多种图形界面的第三方库
  - Tk：一个开源的GUI工具包
  - Tkinter是Tk的Python接口，可供快速创建GUI应用程序。 简单易学 标准库，无需安装

GUI概述

屏幕上应显示哪些组件？
组件应该放在哪里？
组件如何交互？

GUI设计的基本步骤：

- 导入tkinter模块；
- 创建GUI根窗体；
- 添加控件，设置控件属性；
- 布局管理，设置控件的位置关系；
- 事件处理，设置事件响应函数；
- 进入交互循环控制，等待用户触发事件响应。

GUI概述

❖ 创建GUI根窗口：

```
import tkinter as tk    #导入tkinter模块

root = tk.Tk()          #创建一个根窗体实例root
root.mainloop()         #进入根窗体主循环
                        #除非用户关闭，否则程序始终运行


root.title('我的第一个Python窗体')   #设置标题文字
root.geometry('320x240')   #设置窗体尺寸（像素）
root.config(bg = 'red')       #设置窗体背景颜色
root.mainloop()          #进入根窗体主循环
```

GUI概述

❖ 添加控件（组件），并布局

```
import tkinter as tk
root = tk.Tk()          #创建根窗体

label = tk.Label(root, text="我是一个标签")  #创建一个label实例
button = tk.Button(root, text="我是一个按钮") #创建一个botton实例

label.pack()            #布局标签label
button.pack()           #布局按钮button

root.mainloop()         #进入主循环
```

GUI概述

❖ 事件响应

```
import tkinter as tk
def callback():     #定义事件处理函数
    print("按钮被点击一次")

root = tk.Tk()
label = tk.Label(root, text="我是一个标签")
button = tk.Button(root, text="按钮", command = callback)
label.pack()
button.pack()
root.mainloop()
```

绑定事件

Tkinter常用控件

tkinter中，每种组件都是一个类，创建某个组件其实就是将这个类实例化。

- Tkinter有10多种常用控件
  - 标签Label、消息Message、单行输入框Entry、多行文本框Text
  - 按钮Button、单选按钮Radiobutton、复选按钮Checkbutton
  - 列表框Listbox、组合框Combobox
  - 滑块Scale、滚动条Scrollbar
  - 菜单Menu
  - 框架Frame
  - 子窗体Toplevel

Tkinter常用控件

tkinter中，每种组件都是一个类，创建某个组件其实就是将这个类实例化

通过构造函数给组件设置一些属性，指定一个父容器

为组件设置布局管理器，明确其在父容器中的放置方式。

```
from tkinte
root = Tk(
```

类的一个实例

```
Label(root, text="这是一个标签").pack()
Button(root, text="按钮").pack()
Checkbutton(root, text="多选框").pack()
Entry(root).pack()
Radiobutton(root, text="男", value=1).pack()
Radiobutton(root, text="女", value=2).pack()
Scale(root, orient='horizonta').pack()


root.mainloop()
```

Tkinter常用控件

- **实例化窗体控件的格式：**

控件实例名 = 控件 (父容器, [<属性1=值1>,...,<属性n=值n>])

控件实例名.布局方法( )

- **控件的共同属性**

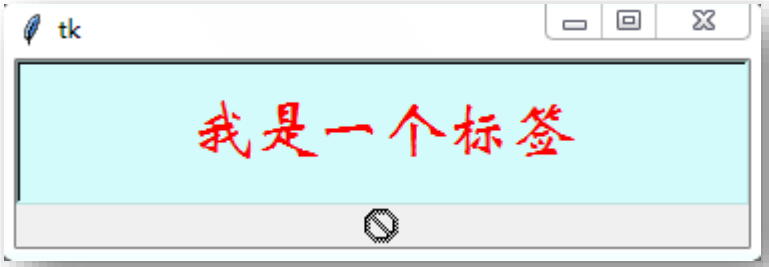
控件通常包括尺寸、颜色、字体、相对位置、浮雕样式、图标样式和悬停光标等共同属性。

不同的控件由于其形状和功能不同，又有其特征属性。

常见的控件共同属性

属性	说明	取值
anchor	文本起始位置	CENTER(默认), E, S, W, N, NE, SE, SW, NW
bg	背景色	
bd	加粗	
bitmap	黑白二值图标	error, info, question, warning, hourglass,.....
cursor	鼠标悬停光标	arrow, circle, cross, heart.....
font	字体	
fg	前景色	
height	高（单位为行）	
width	宽（单位为行）	
image	显示图像	
justify	多行文本的对齐方式	
padx	水平扩展像素	
pady	垂直扩展像素	
relief	3D浮雕样式	FLAT, RAISED, SUNKEN, GROOVE, RIDGE
state	控件实例状态是否可用	NORMAL（默认）, DISABLED

Tkinter常用控件



【例】标签Label及其常见属性示例

```
from tkinter import *
root = Tk()
lb1 = Label(root, text='我是一个标签',\
            bg = '#d3fbfb', fg='red',\
            font=('华文新魏',32),\
            width=20, height=2, relief=SUNKEN)
lb2 = Label(root, bitmap='error', cursor='cross')

lb1.pack()
lb2.pack()
root.mainloop()
```

标签实例lb具有一系列属性，不区分先后顺序

控件布局

❖ 常见的布局方法：pack( )、grid( )、place( )

◆ 1. pack( )方法

默认方式下，按语句先后，自上而下排列控件，只占用能容纳下当前控件的最小空间。

- ➢ 参数fill：可取值X、Y、BOTH，分别表示允许控件向水平方向、竖直方向或二维伸展填充父容器未被占用的空间。
- ➢ 参数side：可取值TOP（默认）、LEFT、RIGHT、BOTTOM，分别表示本控件实例的布局相对于下一个控件实例的方位。

控件布局

【例】不加参数排列的标签控件。

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

lbred = Label(root, text="Red", fg="red",relief=GROOVE)
lbred.pack()
lbgreen = Label(root,text="green",fg="green",relief=GROOVE)
lbgreen.pack (side=RIGHT)
lbblue = Label(root, text="蓝", fg="blue",relief=GROOVE)
lbblue.pack (fill=X)

root.mainloop()
```

观察改变参数之后的布局效果

控件布局

2. grid( )方法

基于网格的布局，根据行号和列号，将控件放置于二维网格中。使用grid()方法布局时，需要指定两个参数：行row，列column，序号从0开始，且每个网格中只能容纳一个控件。

```
from tkinter import *

root = Tk()

Label(root, width=15, height=3, bg="red").grid(row=0, column=0)
Label(root, width=15, height=3, bg="green").grid(row=0, column=1)
Label(root, width=15, height=3, bg="blue").grid(row=0, column=2)
Label(root, width=15, height=3, bg="white").grid(row=1, column=0)
Label(root, width=15, height=3, bg="black").grid(row=1, column=1)
Label(root, width=15, height=3, bg="grey").grid(row=1, column=2)

root.mainloop()
```

控件布局

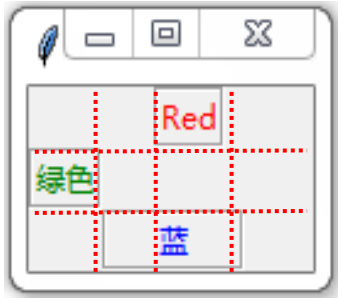
2. grid( )方法

除了row，column，还有其它参数：

- columnspan：跨越的列数，默认为1列
- rowspan：控件实例所跨越的行数，默认为1行。
- sticky：设置组件在网格中的对齐方式。取值：NE（右上角）、SE（右下角）、SW（左下角）、NW（左上角）、E、S、W、N、E+W（水平拉伸）、N+S（垂直拉伸）、E+S+N+S（水平竖直拉伸）。
- ipadx, ipady：组件的内部间隔距离，用来设置控件的大小。
- padx, pady：组件的外部间隔距离，用来设置控件所在网格的大小。

控件布局

【例】 用grid()方法排列标签，效果如图所示。



设想有一个3×4的表格，起始行、列序号均为0。

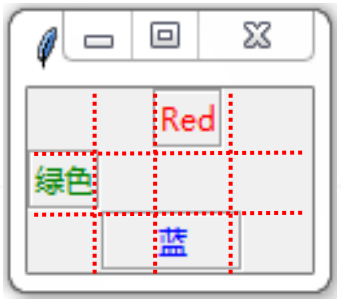
1）将标签lbred置于第2列第0行；

2）将标签lbgreen置于第0列第1行；

3）将标签lbblue置于第1列起跨2列第2行，占20像素宽

控件布局

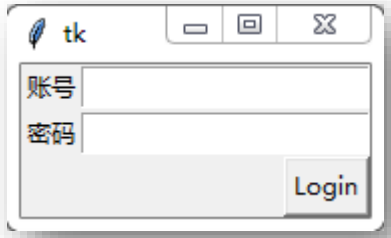
【例】 用grid()方法排列标签，效果如图所示。




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

lbred = Label(root, text="Red", fg="red",relief=GROOVE)
lbred.grid(column=2, row=0)
lbgreen = Label(root,text="绿色", fg="green", relief=GROOVE)
lbgreen.grid(column=0, row=1)
lbblue = Label(root, text="蓝", fg="blue",relief=GROOVE)
lbblue.grid(column=1, columnspan=2, ipadx=20, row=2)

root.mainloop()
```

控件布局

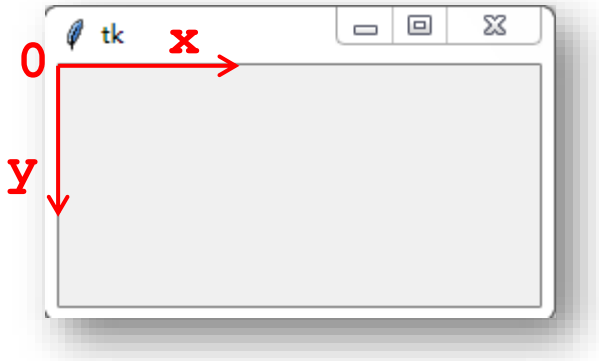
【例】 用grid()方法实现如图的窗体。




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

Label(root, text="账号").grid(row=0, sticky=W)
Label(root, text="密码").grid(row=1, sticky=W)
Entry(root).grid(row=0, column=1, sticky=E)
Entry(root).grid(row=1, column=1, sticky=E)
Button(root, text="Login").grid(row=2, column=1, sticky=E)

root.mainloop()
```

试一试，怎样使Login按钮居中

控件布局



3. place( )方法

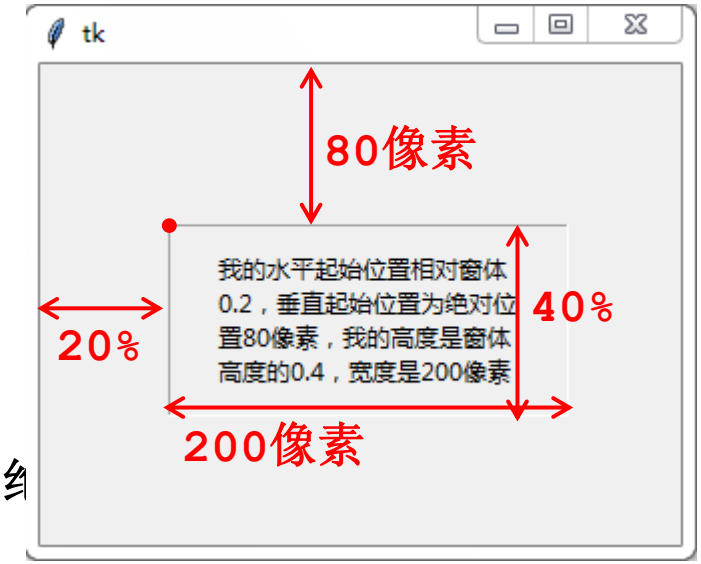
根据控件实例在父容器中的绝对或相对坐标进行布局。常用参数：

- x, y：控件左上角的x和y坐标，起始位置（单位为像素）
- relx, rely：控件相对于父容器的x和y坐标，相对于父容器的宽和高的比例位置，取值在0.0～1.0之间。
- height, width：控件实例的高度和宽度（单位为像素）。
- relheight, relwidth：控件相对于父容器的高度和宽度，取值在0.0～1.0之间。

控件布局

【例】 用place()方法排列消息（多行标签），效果如图。

```
from tkinter import *

root = Tk()
root.geometry('320x240')

msg = Message(root, text='''
我的水平起始位置相对窗体0.2，垂直起始位置为绝
体高度的0.4，宽度是200像素
''', relief=GROOVE)
msg.place(relx=0.2, y=80, relheight=0.4, width=200)

root.mainloop()
```



Tkinter常用控件

- 标签Label、消息Message、输入框Entry、文本框Text
- 按钮Button
- 单选按钮Radiobutton、复选框Checkbutton
- 列表框Listbox、组合框Combobox
- 滑块Scale、滚动条Scrollbar
- 菜单Menu
- 框架Frame
- 子窗体Toplevel
- 模式对话框

文本输入/输出相关控件

- 标签（Label）和消息（Message）

用于呈现文本，两者仅是单行与多行的区别，属性与用法基本一致。重要属性text：控件在第一次呈现时的固定文本。

如要在程序执行后修改属性的值，则可使用以下方法：

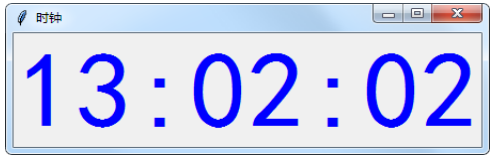
- 1）控件实例名['属性名'] = 新值
- 2）控件实例名.config(属性名=新值)
- 3）修改text属性：先定义一个内部类型变量var = StringVar()，然后将该变量与textvariable属性绑定，textvariable = var，再用var.set()方法改变var的值，达到修改text的目的。

文本输入/输出相关控件

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

lb1 = Label(root, text='我是标签1', bg = '#d3fbfb', fg='red',\
      font=('华文新魏',24), width=20, height=2, relief=SUNKEN)
lb2 = Label(root, text='标签2', width=20, height=2, relief=SUNKEN)
lb3 = Label(root, text='标签3', relief=SUNKEN)

lb1.pack()
lb2.pack()
lb3.pack()

root.mainloop()
```

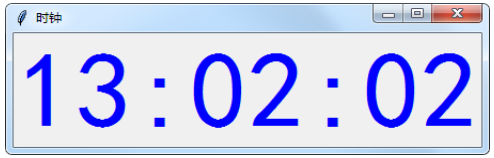
【练习】试一试，用三种方法，分别修改三个标签的背景颜色、宽和高、文本内容。

```
from tkinter import *
import time

def gettime():
    #获取当前时间并转为字符串
    timestr = time.strftime("%H:%M:%S")
    #重新设置标签文本
    lb.config(text=timestr)
    #每隔1秒调用gettime自身获取时间
    root.after(1000,gettime)

root = Tk()
root.title("时钟")

lb = Label(root, text='', fg='blue', font=('黑体',36))
lb.pack()
gettime()   #调用gettime函数获取当前时间，并改变标签文本
root.mainloop()
```



试用textvariable属性来实现

```
from tkinter import *
import time

def gettime():
    #获取当前时间并转为字符串
    timestr = time.strftime("%H:%M:%S")
    #修改val变量的值
    var.set(timestr)
    #每隔1秒调用gettime自身获取时间
    root.after(1000,gettime)

root = Tk()
root.title("时钟")

var = StringVar() #定义var变量
lb = Label(root, textvariable=var, fg='blue', font=('黑体',36))
lb.pack()
gettime()  #调用gettime函数获取当前时间，并改变标签文本
root.mainloop()
```



文本输入/输出相关控件

- 输入框（Entry）

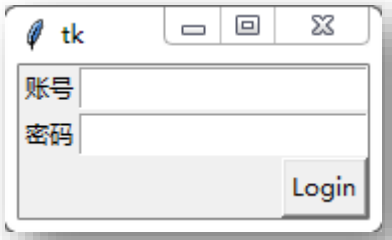
用来接收单行的文本输入，功能较为单一。常用方法有：

- 获取输入框中的文本：get()
- 清空输入框中的文本：delete(0,END)
- 控件设为焦点：focus_set()

参数：

- show: 文本框显示的字符，若为*，则表示文本框为密码框
- selectbackground: 选定文本背景色
- selectbackground: 选定文本前景色

文本输入/输出相关控件



❖ 输入框（Entry）

【例】输入账号和密码信息，点击Login，即可将信息输出到屏幕。

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

def login():
    print(en1.get())
    print(en2.get())

Label(root, text="账号").grid(row=0, sticky=W)
Label(root, text="密码").grid(row=1, sticky=W)
en1 = Entry(root)
en2 = Entry(root, show='*')
Button(root, text="Login", command=login ).grid(row=2, column=1, sticky=E)

en1.grid(row=0, column=1, sticky=E)
en2.grid(row=1, column=1, sticky=E)

root.mainloop()
```

试一试：增加一个按钮Clear，点击即可清空输入框的信息。

文本输入/输出相关控件

- 输入框（Entry）

  获取和设置输入框中的文本，有另一种方法：

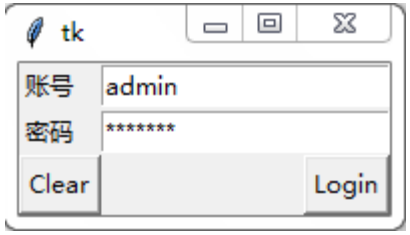
  ent = Entry(root, textvariable = var)

  绑定变量：

  var = StringVar()

  - var.get() : 获取文本框中的内容
  - var.set(value): 设置文本框中的内容

```
from tkinter import *
root = Tk()
def login():  #事件响应函数
    print(var1.get())
    print(var2.get())

def clear():  #事件响应函数
    var1.set('')
    var2.set('')

var1 = StringVar()  #定义变量var1
var2 = StringVar()

Label(root, text="账号").grid(row=0, sticky=W)
Label(root, text="密码").grid(row=1, sticky=W)
en1 = Entry(root, textvariable=var1)
en2 = Entry(root, textvariable=var2, show='*')
var1.set('admin')
en2.focus_set()   #输入框2设为焦点

Button(root, text="Login", command=login ).grid(row=2, column=1, sticky=E)
Button(root, text="Clear", command=clear ).grid(row=2, column=0)
en1.grid(row=0, column=1, sticky=E)
en2.grid(row=1, column=1, sticky=E)

root.mainloop()
```



文本输入/输出相关控件

❖ 文本框（Text）

不仅可以用来显示多行文本，还可以编辑多行文本。

表 8-5　文本框（Text）的常用方法

方　　法	功　　能
delete(起始位置 [,终止位置])	删除指定区域文本
get(起始位置 [,终止位置])	获取指定区域文本
insert(位置 [,字符串]…)	将文本插入到指定位置
see(位置)	在指定位置是否可见文本，返回布尔值
index(标记)	返回标记所在的行和列
mark_names()	返回所有标记名称
mark_set(标记，位置)	在指定位置设置标记
mark_unset(标记)	去除标记

注：表中位置的取值可为整数、浮点数或 END（末尾），例如 0.0 表示第 0 行第 0 列。

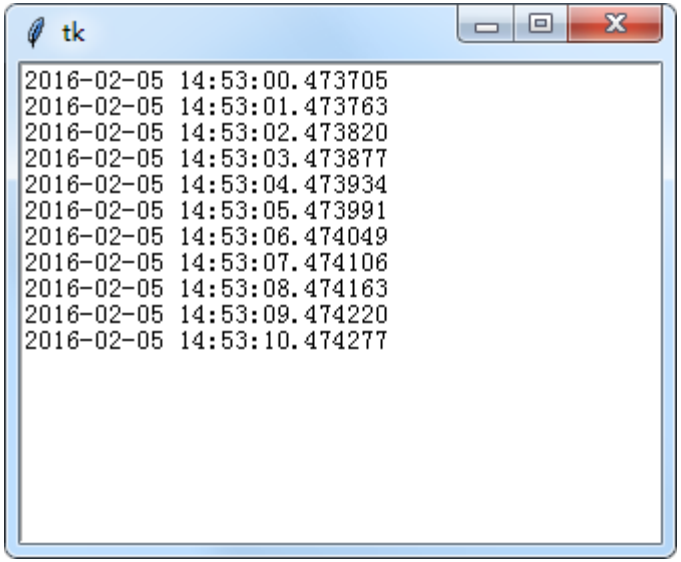
文本输入/输出相关控件

❖ 文本框（Text）

【例】运行程序，理解文本插入方法insert()，文本获取方法get()

```
from tkinter import *

root = Tk()

txt = Text(root, font=('consolas',14), height=5)
txt.pack()
txt.insert('1.0', 'Hello\n')
txt.insert('1.2','Python\n')
txt.insert(END, 'Hi!\n')

print(txt.get('1.0',END))

root.mainloop()
```

```
from tkinter import *
import time
import datetime

def gettime():
    #获取当前日期时间，转为字符串，末尾加换行符
    s = str(datetime.datetime.now())+'\n'
    #将文本插入文本框的末尾
    txt.insert(END, s)
    #每隔1秒调用gettime自身获取时间
    root.after(1000,gettime)

root = Tk()
root.geometry("320x240")
txt = Text(root)
txt.pack()
gettime()  #调用gettime函数获取当前日期时间，并改变文本框内容

root.mainloop()
```



Tkinter常用控件

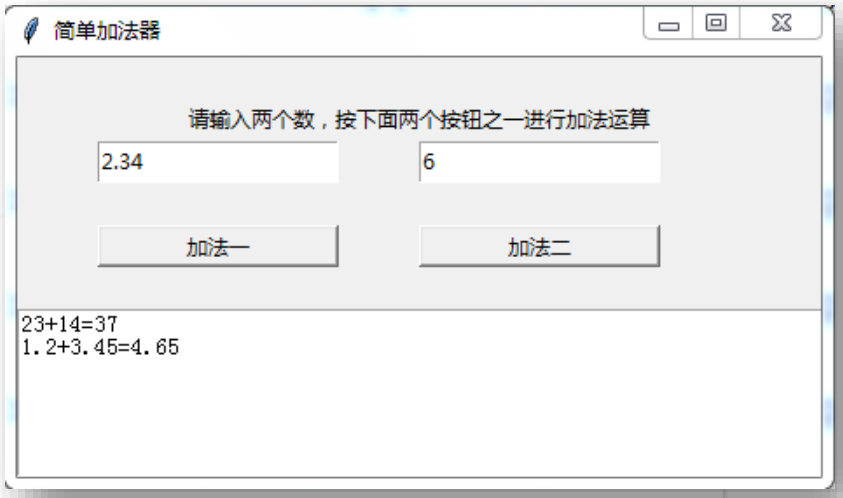
❖ 按钮（Button）

用来响应用户的一个鼠标单击操作，能够与一个函数关联。当按钮被按下时，自动调用该函数。属性command是最为重要的属性。

通常，按钮要触发执行的程序**以函数形式预先定义**，然后可用以下两种方法调用函数：

- 1）直接调用函数。参数表达式为“command＝函数名”，注意函数名后面不要加括号，也不能传递参数。
- 2）利用匿名函数调用函数。参数表达式为“command＝lambda：函数名（参数列表）”，函数名后可带参数。

Tkinter常用控件

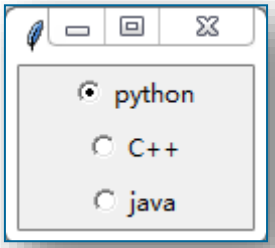


【例】实现简单的加法器

```
from tkinter import *

def add1():
    a = ent1.get()
    b = ent2.get()
    c = eval(a)+eval(b)
    txt.insert(END, "%s+%s=%s\n" %(a, b, c))
    ent1.delete(0,END)
    ent2.delete(0,END)

def add2(x,y):
    c = eval(x)+eval(y)
    txt.insert(END, "%s+%s=%s\n" %(x, y, c))
    ent1.delete(0,END)
    ent2.delete(0,END)
```

```
root = Tk()
root.title("简单加法器")
root.geometry("460x240")

lb = Label(root, text="请输入两个数，按下面两个按钮之一进行加法运算")
lb.place(relx=0.1, rely=0.1, relwidth=0.8, relheight=0.1)
ent1 = Entry(root)
ent1.place(relx=0.1, rely=0.2, relwidth=0.3, relheight=0.1)
ent2 = Entry(root)
ent2.place(relx=0.5, rely=0.2, relwidth=0.3, relheight=0.1)

btn1 = Button(root, text="加法一", command = func1)
btn1.place(relx=0.1, rely=0.4, relwidth=0.3, relheight=0.1)
btn2 = Button(root, text="加法二", \
              command = lambda:func2(ent1.get(), ent2.get()))
btn2.place(relx=0.5, rely=0.4, relwidth=0.3, relheight=0.1)

txt = Text(root)
txt.place(rely=0.6, relheight=0.4)

root.mainloop()
```

Tkinter常用控件



❖ 单选按钮（Radiobutton）

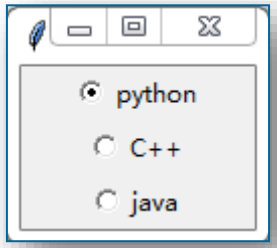
它有组的概念，在同一组中只能选中一个按钮。一组单选按钮和同一个变量关联，当选中其中一个按钮，该变量就被设置为某一个预定的值。

【例】创建一组单选按钮。

```
from tkinter import *
root = Tk()
#将三个单选按钮与同一变量var绑定
var = IntVar()   #声明整型变量var
var.set(1)     #预选中value=0的选项
Radiobutton(root, text='python', variable=var, value=0).pack()
Radiobutton(root, text='C++', variable=var, value=1).pack()
Radiobutton(root, text='java', variable=var, value=2).pack()
root.mainloop()
```

若变量var未绑定，则每个Radiobutton自成一组。

Tkinter常用控件

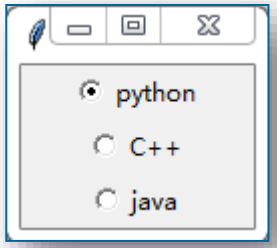


❖ 单选按钮（Radiobutton）

该控件除具有共有属性外，还具有：显示文本（text）、返回变量（variable）、返回值（value）和响应函数名（command）等属性。

- 属性“command=相应函数名”。所有单选项共享同一函数。
- 属性“variable=var”。需预先声明变量的类型 `var = IntVar()` 或 `var = StringVar()`。
- 属性“value=实例被选中时的值”。
  - `var.get()`方法可读取当前选中的值。
  - `var.set()`方法可设置选中项。

Tkinter常用控件



❖ 单选按钮（Radiobutton）

IntVar()与StringVar()的区别

【例】在上例中，增加单击事件响应函数：点击单选按钮，会将选中结果输出到屏幕上。

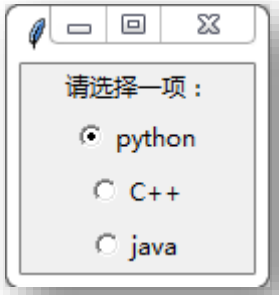
```
def mySel():   #事件响应函数
    ls = ['python','C++','java']
    print( ls[var.get()] )

var = IntVar()
Radiobutton(root, text='python', variable=var, value = 0, \
            command = mySel).pack()
Radiobutton(root, text='C++', variable=var, value = 1, \
            command = mySel).pack()
Radiobutton(root, text='java', variable=var, value = 2, \
            command = mySel).pack()
```

用var.get()读取value值

一组单选按钮共享同一响应函数

Tkinter常用控件

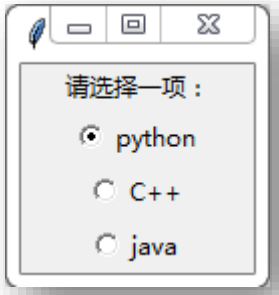
【例】利用单选按钮实现：单击选项后，将选中结果显示在标签上。



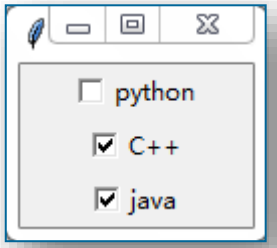
【分析】

- 增加一个Label标签，设置text属性。
- 事件响应函数：获得选中项的值，更新标签的text属性。

```
from tkinter import *

def mySel():
    s = "您选中了" + var.get()
    lb.config(text = s)

root = Tk()
lb = Label(root, text="请选择一项：")
lb.pack()

var = StringVar()  #字符串类型的变量var
ls = ['python', 'C++', 'java']
var.set(ls[0])     #预设值
for i in ls:       #循环创建一组单选按钮，与变量var绑定
    Radiobutton(root, text=i, variable=var, value=i, command=mySel).pack()

root.mainloop()
```



Tkinter常用控件



❖ 复选按钮（Checkbutton）

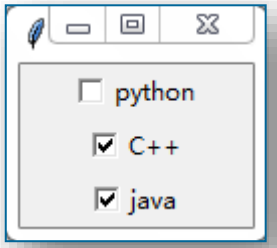
可以返回多个选项值，每个选项可以表示两种状态：On和Off。

可以设置单击选项时的响应函数，但通常并不直接触发函数的执行。

该控件除具有共有属性外，还具有：显示文本（text）、返回变量（variable）、选中返回值（onvalue）和未选中默认返回值（offvalue）等重要的属性。

- **属性"variable=var"：需要预先逐项分别声明变量的类型**
  var=IntVar()或var=StringVar()。
- **用var.get()方法取得被选中实例的onvalue或offvalue值。**

Tkinter常用控件



❖ 复选按钮（Checkbutton）

```
# 创建三个复选按钮

from tkinter import *

root = Tk()
var1 = StringVar()
var2 = StringVar()
var3 = StringVar()
ls = ['python', 'C++', 'java']
ch1=Checkbutton(root, text = ls[0], variable = var1, onvalue = ls[0], offvalue = '')
ch2=Checkbutton(root, text = ls[1], variable = var2, onvalue = ls[1], offvalue = '')
ch3=Checkbutton(root, text = ls[2], variable = var3, onvalue = ls[2], offvalue = '')
ch1.pack()
ch2.pack()
ch3.pack()

root.mainloop()
```

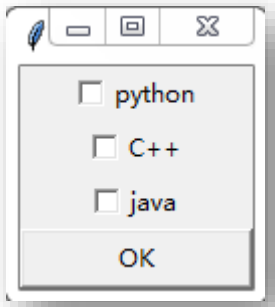
声明三个字符串类型的变量

分别与选项绑定

设置选中值和未选中值

Tkinter常用控件

【例】 在复选按钮的下方，增加一个Button。点选复选框，再按下Button，将复选的结果输出屏幕上。

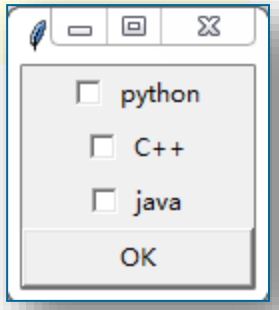


【分析】

- 增加一个Button按钮，设置其command属性。
- 按钮的回调函数：读取三个选项按钮的值（选中或未选中），print到屏幕。

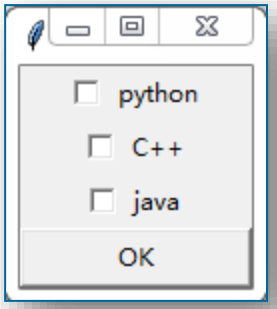
```
from tkinter import *
root = Tk()

def mySel():   #回调函数
    pass

var1 = StringVar()   #声明要绑定的变量
var2 = StringVar()
var3 = StringVar()
ls = [' python',' C++',' java']     #文本的列表
ch1=Checkbutton(root,text=ls[0],variable=var1,onvalue=ls[0],offvalue='')
ch2=Checkbutton(root,text=ls[1],variable=var2,onvalue=ls[1],offvalue='')
ch3=Checkbutton(root,text=ls[2],variable=var3,onvalue=ls[2],offvalue='')
ch1.pack()
ch2.pack()
ch3.pack()
btn = Button(root, text='OK', command = mySel)
btn.pack(fill=X)

root.mainloop()
```

```
def mySel():
    s = var1.get() + var2.get() + var3.get()
    if s=='':
        print("None")
    else:
        print(s)
```



```
# 将绑定的变量var声明为IntVar类型

from tkinter import *
root = Tk()

def mySel():
    v = var1.get()+var2.get()+var3.g
    if v==0:
        print("None")
    else:
        s1 = ls[0] if var1.get()==1 else ''
        s2 = ls[1] if var2.get()==1 else ''
        s3 = ls[2] if var3.get()==1 else ''
        print(s1+s2+s3)

var1 = IntVar() #声明变量
var2 = IntVar()
var3 = IntVar()
ls = [' python',' C++',' java']
ch1=Checkbutton(root,text=ls[0],variable=var1,onvalue=1,offvalue=0)
ch2=Checkbutton(root,text=ls[1],variable=var2,onvalue=1,offvalue=0)
ch3=Checkbutton(root,text=ls[2],variable=var3,onvalue=1,offvalue=0)
ch1.pack()
ch2.pack()
ch3.pack()
btn = Button(root, text='OK', command = mySel)
btn.pack(fill=X)

root.mainloop()
```

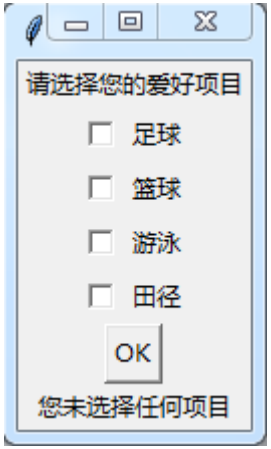
如果被选中，则为ls中的某一项，否则为空串。

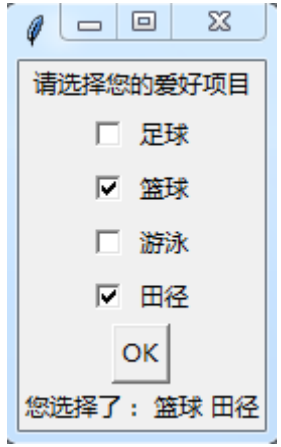


选中值和未选中值都是整数

Tkinter常用控件

【练习】 在上例代码的基础上，利用复选框实现下图效果。单击OK后，可以将选中结果显示在标签上。

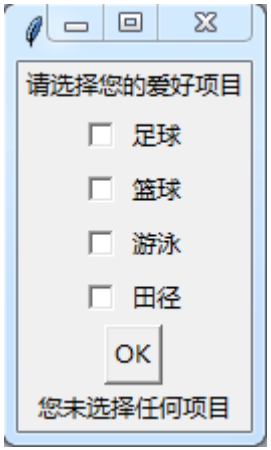


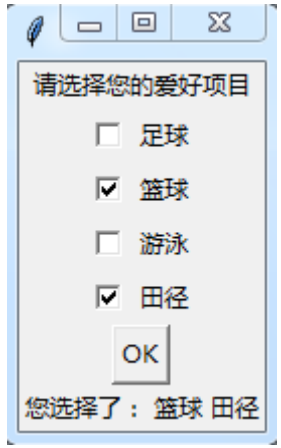


【分析】

- 窗口顶部和底部分别增加一个标签。
- 增加一个Checkbutton实例。
- 修改回调函数：读取四个选项的值，用相应的文本去更新Label的text属性。

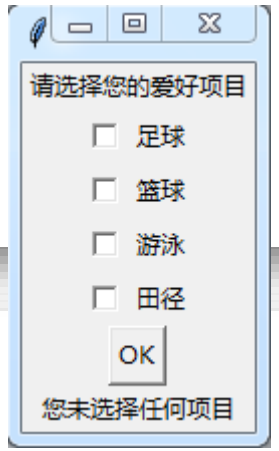
```
def mySel():
    v = var1.get()+var2.get()+var3.get()+var4.get()
    if v==0:
        s = "您未选中任何项目"
    else:
        s1 = ls[0] if var1.get()==1 else ''
        s2 = ls[1] if var2.get()==1 else ''
        s3 = ls[2] if var3.get()==1 else ''
        s4 = ls[3] if var4.get()==1 else ''
        s = "您选择了: "+s1+s2+s3+s4
    lb.config(text = s)

Label(root, text='请选择您的爱好项目').pack()
var1 = IntVar()  #声明变量为Int型
var2 = IntVar()
var3 = IntVar()
var4 = IntVar()
ls = [' 足球',' 篮球',' 游泳', ' 田径']
Checkbutton(root,text=ls[0],variable=var1,onvalue=1,offvalue=0).pack()
Checkbutton(root,text=ls[1],variable=var2,onvalue=1,offvalue=0).pack()
Checkbutton(root,text=ls[2],variable=var3,onvalue=1,offvalue=0).pack()
Checkbutton(root,text=ls[3],variable=var4,onvalue=1,offvalue=0).pack()
btn = Button(root, text='OK', command = mySel)
btn.pack()
lb = Label(root)
lb.pack()
```

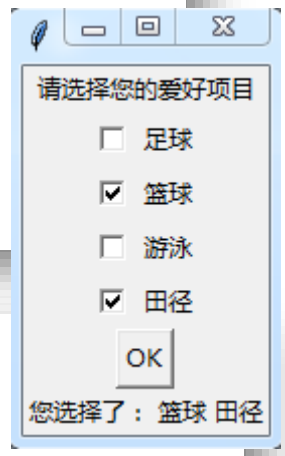




利用字典实现多项显示，根据键（选中项的值），选择显示的内容。

```
def mySel():
    dic = {0:'', 1:' 足球', 2:' 篮球', 3:' 游泳', 4:' 田径'}  #定义字典
    values = [var1.get(), var2.get(), var3.get(), var4.get()]
    s = ''
    for i in values:
        s += dic[i]
    if s=='':
        lb.config(text = '您未选择任何项目')
    else:
        lb.config(text = '您选择了：'+s)
```






```
var1 = IntVar()  #声明变量为Int型
var2 = IntVar()
var3 = IntVar()
var4 = IntVar()
ls = [' 足球',' 篮球',' 游泳', ' 田径']
ch1=Checkbutton(root, text=ls[0], variable=var1, onvalue=1, offvalue=0)
ch2=Checkbutton(root, text=ls[1], variable=var2, onvalue=2, offvalue=0)
ch3=Checkbutton(root, text=ls[2], variable=var3, onvalue=3, offvalue=0)
ch4=Checkbutton(root, text=ls[3], variable=var4, onvalue=4, offvalue=0)
```

选中项的值各不相同