# tkinter图形化界面介绍

## 1.1 认识tkinter图形化界面开发

tkinter是实现图形化界面开发python内置的模块，无需安装。

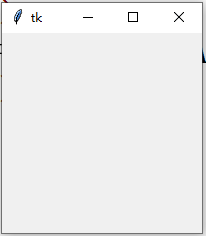
我们在使用这个模块的时候直接进行导入即可：

导入的方式：from tkinter import \*

创建一个窗口对象，即可显示该窗口：root = Tk()

root叫根窗口，在这个基础上可以创建n多个窗口，和n多个组件。

让程序持续执行：root.mainloop()，点击窗口的关闭按钮才能关闭程序。



## 1.2 窗口的属性和方法设置

|  |  |
| --- | --- |
| title() | 设置窗口的标题 |
| geometry("200x300") | 表示宽度为200，高度为300像素的窗口大小 |
| maxsize(400,500) | 只能将窗口拖拽扩大400x500的大小 |
| minsize(200,100) | 只能将窗口拖拽缩小200x100的大小 |
| configure(bg = "red") | 更改窗口的背景颜色，可以是颜色名称，也可以是16进制颜色 |
| resizable(0,0) | 不可以更改窗口大小 |
| state("zoomed") | 最大化窗口 |
| iconify() | 最小化窗口 |
| iconbitmap("文件路径") | 更改默认图标, 扩展名是·ico的文件 |

# Label（标签文字）

## 1.1 lable属性表格

|  |  |  |
| --- | --- | --- |
| **属性** | **含义** | **备注** |
| anchor | 1. 控制文本（或图像）在 Label 中显示的位置 2. "n", "ne", "e", "se", "s", "sw", "w", "nw", 或者 "center" 来定位（ewsn 代表东西南北，上北下南左西右东） 3. 默认值是 "center" |  |
| background（bg） | 1. 设置背景颜色 2. 默认值由系统指定 |  |
| bitmap | 1. 指定显示到 Label 上的位图 2. 如果指定了 image 选项，则该选项被忽略 |  |
| borderwidth (bd) | 1. 指定 Label 的边框宽度 2. 默认值由系统指定，通常是 1 或 2 像素 |  |
| compound | 1. 控制 Label 中文本和图像的混合模式 2. 默认情况下，如果有指定位图或图片，则不显示文本 3. 如果该选项设置为 "center"，文本显示在图像上（文本重叠图像） 4. 如果该选项设置为 "bottom"，"left"，"right" 或 "top"，那么图像显示在文本的旁边（如 "bottom"，则图像在文本的下方） 5. 默认值是 NONE |  |
| cursor | 1. 指定当鼠标在 Label 上飘过的时候的鼠标样式 2. 默认值由系统指定 |  |
| disabledforeground | 1. 指定当 Label 不可用的时候前景色的颜色 2. 默认值由系统指定 |  |
| font | 1. 指定 Label 中文本的字体(注：如果同时设置字体和大小，应该用元组包起来，如（"楷体", 20） 2. 一个 Label 只能设置一种字体 3. 默认值由系统指定 |  |
| foreground(fg) | 1. 设置 Label 的文本和位图的颜色 2. 默认值由系统指定 |  |
| height | 1. 设置 Label 的高度 2. 如果 Label 显示的是文本，那么单位是文本单元 3. 如果 Label 显示的是图像，那么单位是像素（或屏幕单元） 4. 如果设置为 0 或者干脆不设置，那么会自动根据 Label 的内容计算出高度 |  |
| width | 1. 设置 Label 的宽度 2. 如果 Label 显示的是文本，那么单位是文本单元 |  |
| highlightbackground | 1. 指定当 Label 没有获得焦点的时候高亮边框的颜色 2. 默认值由系统指定，通常是标准背景颜色 |  |
| highlightcolor | 1. 指定当 Label 获得焦点的时候高亮边框的颜色 2. 默认值由系统指定 |  |
| highlightthickness | 1. 指定高亮边框的宽度 2. 默认值是 0（不带高亮边框） |  |
| image | 1. 指定 Label 显示的图片 2. 该值应该是 PhotoImage，BitmapImage，或者能兼容的对象 3. 该选项优先于 text 和 bitmap 选项 |  |
| justify | 1. 定义如何对齐多行文本 2. 使用 "left"，"right" 或 "center" 3. 注意，文本的位置取决于 anchor 选项 4. 默认值是 "center" |  |
| padx/pady | 1. 指定 Label 水平(垂直)方向上的额外间距（内容和边框间） 2. 单位是像素 |  |
| relief | 1. 指定边框样式 2. 默认值是 "flat" 3. 另外你还可以设置 "groove", "raised", "ridge", "solid" 或者 "sunken" |  |
| state | 1. 指定 Label 的状态 2. 这个标签控制 Label 如何显示 3. 默认值是 "normal 4. 另外你还可以设置 "active" 或 "disabled" |  |
| takefocus | 1. 如果是 True，该 Label 接受输入焦点 2. 默认值是 False |  |
| text | 1. 指定 Label 显示的文本 2. 文本可以包含换行符 3. 如果设置了 bitmap 或 image 选项，该选项则被忽略 |  |
| textvariable | 1. Label 显示 Tkinter 变量（通常是一个 StringVar 变量）的内容 2. 如果变量被修改，Label 的文本会自动更新 |  |
| underline | 1. 跟 text 选项一起使用，用于指定哪一个字符画下划线（例如用于表示键盘快捷键）  2. 默认值是 -1 3. 例如设置为 1，则说明在 Button 的第 2 个字符处画下划线 |  |

#Label是标签的意思，这个标签中我们可以放置文本、图片

## 1.2 lable文本属性

#将Tk类实例化没有参数。这将创建Tk的顶级窗口小部件，该窗口小部件通常是应用程序的主窗口

from tkinter import \*

#初始化Tk

root = Tk()

#创建一个label

label = Label(root,text = 'Hello Tkinter')

参数1：指的是我们要将这个标签文字放置在哪个地方，肯定是窗口中了，所以写root;

参数2：指的是我们要放置的文本内容是啥，这里使用了关键字参数。

#label仅仅是一个返回的对象，如果要显示这个对象，比需要要把这个对象装包显示。

label.pack()#显示label对象

效果如下：



#我们也可以设置字体的颜色

label = Label(root,fg = 'White',bg = 'Red',text = "我爱中国")

fg关键字参数：指的是字体本身的颜色

ba关键字参数：指的是字体的背景颜色

相同点：都可以使用表示颜色的单词，比如：Red Green Blue Yellow等，也可以使用16进制颜色的表示方法：#02B434 #020234 #EA1734

下面的网站是16进制颜色转换的：

<https://www.sioe.cn/yingyong/yanse-rgb-16/>

效果如下：



使用width = 10,height = 13，这个两个属性值可以更改，背景颜色的大小（大小为像素单位）

label = Label(root,fg = 'White',width = 20,height = 34,bg = 'Red',text = "我爱中国")

参考效果如下：



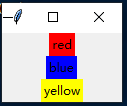
并且我们可以一次性创建多个lable对象：

Label(root,text = 'red',bg = 'red').pack()

Label(root,text = 'blue',bg = 'blue').pack()

Label(root,text = 'yellow',bg = 'yellow').pack()

效果如下图所示：



显示多行文本：

Label(root,text = ' I like learning python very much, because it is very interesting',width = 10,height = 3).pack()

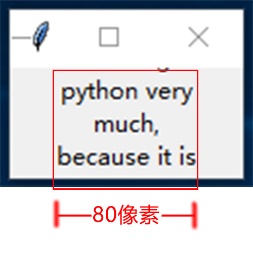
这个text文本显然过长，都被width和height的属性限制在10\*3的区域范围内。

wraplength：指定多少单位后开始换行，单位是屏幕单元，就是我们所说的屏幕像素。

Label(root,text = ' I like learning python very much, because it is very interesting',

width = 10,height = 3,wraplength = 80,).pack()

效果如下：

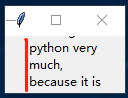


justify: 指定多行的对齐方式有3个参数分别是：center，left， right

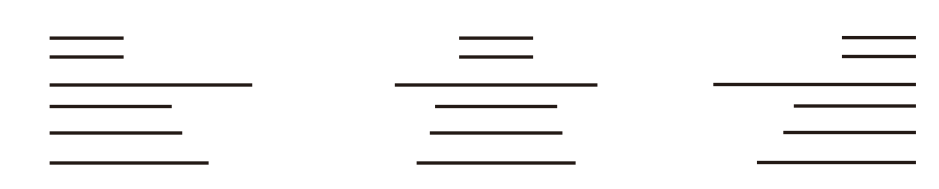
Label(root,text = ' I like learning python very much, because it is very interesting',

width = 10,height = 3,wraplength = 80, justify = "left").pack()

效果图参考：



参考以下图示你就明白了多行对齐方式：



ahchor：指定文本(text)或图像(bitmap/image)在Label中的显示位置，值所在的布局如下所示：

nw n ne

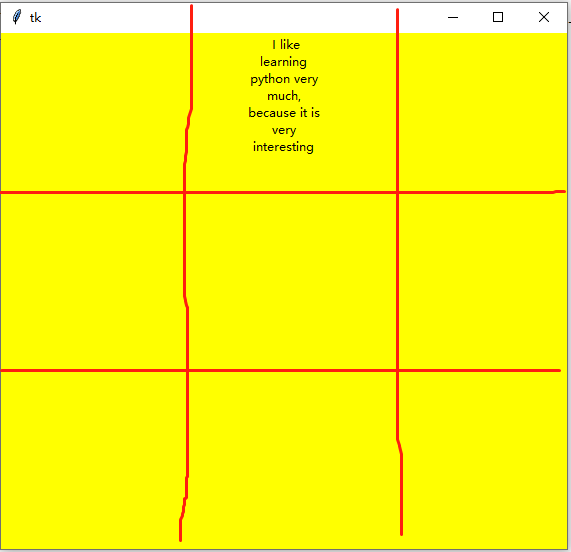
w center e

sw s se

Label(root,text = ' I like learning python very much, because it is very interesting',width = 80,height = 30,bg = "Yellow",wraplength = 80, anchor = 'n' ).pack()

重复不断更换anchor = 'n'，文字会出现不同的位置。

效果如下图所示：



## 1.3 Label图片属性

'''

2.在label上使用内置图片，使用bitmap的使用方法

'''

label = Label(root,bitmap = 'error')

参数1：上述已经介绍

参数2：关键字参数bitmap，内置的图片名error

label.pack()#不要忘记显示一下

效果如下：



'''

使用位图一览表看，直接将error进行替换即可：

\* error

\* hourglass

\* info

\* questhead

\* question

\* warning

\* gray12

\* gray25

\* gray50

\* gray75

'''

#label的第二个例子—使用image属性显示图片

bm = PhotoImage(file = r'图片文件全路径')

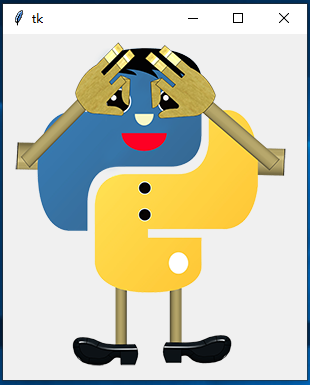
r是用来取消转义，图片只能是png或者是gif格式。全路径指的是包含文件名的路径

比如：C:\Users\Administrator.DESKTOP-MCTPOSE\Desktop\py.png

label = Label(root,image = bm)

label.pack()#不要忘记显示一下

效果如下所示：



## 1.4 同时使用图像与文本

Compound（英文混合的意思）:指定文本(text)与图像(bitmap/image)是如何在Label上显示的，可以使用的值如下：

left： 图像居左

right: 图像居右

top： 图像居上

bottom：图像居下

center：文字覆盖在图像上

图像与文本在Label中的位置演示：

#图像居下：Label(root,text = 'botton',compound = 'bottom',bitmap = 'error').pack()

#图像居上：Label(root,text = 'top',compound = 'top',bitmap = 'error').pack()

#图像居右：Label(root,text = 'right',compound = 'right',bitmap = 'error').pack()

#图像居左：Label(root,text = 'left',compound = 'left',bitmap = 'error').pack()

#文字覆盖在图像上：Label(root,text = 'center',compound = 'center',bitmap = 'error').pack()

更多标签请参考：<https://blog.csdn.net/python_dream/article/details/80932152>

# 三、Button(按钮)

## 1.1简单的Button应用

from tkinter import \*

#定义Button的回调函数（就是当时间启动所做的事情）

def helloButton():

print ('我是一名中国人')

root = Tk() #创建窗体对象

root.title("你好") #设置窗体标题

root.geometry("300x500") #设置窗体大小

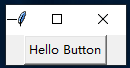
#通过command属性来指定Button的回调函数

Button(root,text = 'Hello Button',command = helloButton).pack()

注意：每点击一次，回调函数就调用一次

效果如下图所示：





## 1.2 Button的属性介绍

注：gui中所有的颜色名称或者16进制颜色都可以

长度单位有的时候是像素，有的时候是字符个数。

|  |  |
| --- | --- |
| 1 | **activebackground**  当鼠标点击的时候，按钮的背景色 |
| 2 | **activeforeground**  当鼠标点击的时候，字体的颜色 |
| 3 | **bd**  按钮边框的粗细，默认为 2 个像素 |
| 4 | **bg**  按钮的背景色 |
| 5 | **command 修改属性的两种方式**  回调函数，command = 函数名，当按钮被点击的时候就会执行函数中的代码块  = |
| 6 | **fg**  按钮上字体的颜色（按钮文本的颜色） |
| 7 | **font**  文本字体 font = 字体名称的字符串，字体名称可以在options—configure idle—Fonts中找 |
| 8 | **height width**  按钮的高度和宽度 |
| 9 | **image**  按钮上要显示的图片，图片的使用方式参考：**button按钮图片使用**  的说法是否 |
| 11 | **justify**  显示多行文本的时候,设置不同行之间的对齐方式，可选项包括left, right, center |
| 12 | **padx padx\_pady**  按钮在x轴方向上的内边距(padding)，pady表示y方向，请参考图示：padx\_pady  按钮在y轴方向上的内边距(padding) |
| 13 | **relief button按钮浮雕样式**  边框样式，设置控件3D效果，可选的有：FLAT、SUNKEN、RAISED、GROOVE、RIDGE。默认 FLAT |
| 14 | **state**  设置按钮组件状态，可选NORMAL（取消）ACTIVE（确认）DISABLED(禁用)。参考：**button按钮状态** |
| 15 | **underline**  下划线。默认值为-1不带下划线。对应下标索引的下划线 |
| 16 | **wraplength**  限制按钮每行显示的字符的数量 |
| 19 | **text**  设置按钮的文本内容 |
| 19 | **anchor**  锚选项，控制文本的位置，默认为中心，可以参考lable的属性。 |

**Button按钮图片**

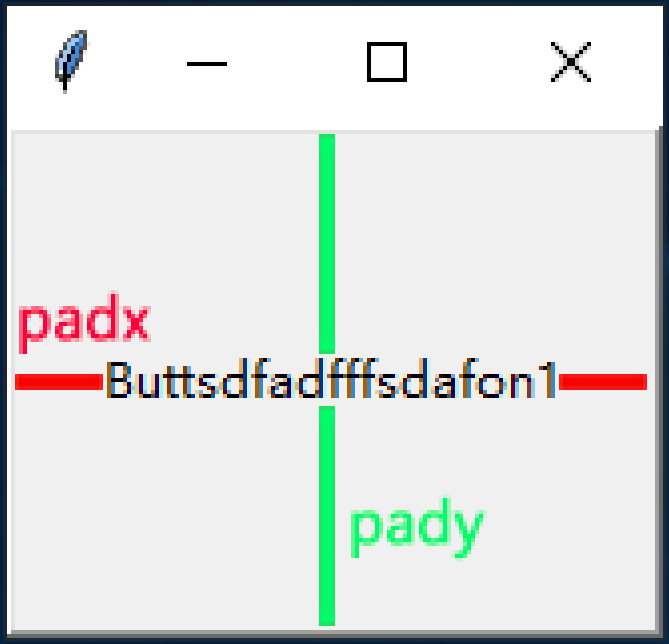
使用可以使用png、gif这两种格式的图片。如果没有设置按钮的height和width属性，图片的源尺寸就是按钮的大小。

img = PhotoImage(file="py.png") #创建图片对象，如果图片和所编辑的代码文本在同一个文件夹中，可以直接写图片的文件名。

# 可以指定按钮图片的大小，不指定就是源图片尺寸。

button = Button(root, text="Button1", height = 20,width = 20,image=img).pack()

**padx\_pady**



**button按钮状态**



**Button按钮relief（浮雕）样式：**

设置relief的值会呈现不同的样式效果：FLAT、SUNKEN、RAISED、GROOVE、RIDGE。默认为 FLAT。

Button(root,text = 'Hello Button',command = helloButton,relief= FLAT).pack()

**修改属性的两种方式**

**方式1：组件对象[“关键字参数”] = 所要修改的值 注意：中括号中的值是一个字符串类型**

**方式2：组件对象.config(关键字参数 = 所要修改的值)**

## 1.3 窗口控件配置管理员

我们可以使用三种方式定位，组件在窗口或者容器（还未讲解）中的位置，这三种方法又叫做在窗口中配置管理员，管理这些组件的位置。方法分别为：pack、grid、place

### 1.4.1 pack方法

Pack()方法的options有：side、fill、padx/pady

**side 参数**

from tkinter import \*

root = Tk()

label1 = Label(root,text = "天津")

label2 = Label(root,text = "郑州")

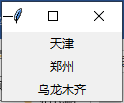
label3 = Label(root,text = "乌龙木齐")

label1.pack(side = TOP)

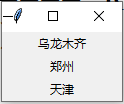
label2.pack(side = TOP)

label3.pack(side = TOP)

**TOP 默认值，由上往下排列**



**BOTTOM由下往上排列**



**LEFT由左往右排列**



**RIGHT由右往左排列**

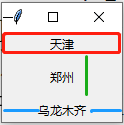


**fill 参数**

**label1.pack(fill = X) #填充x轴**

**label2.pack(pady = 10) #y轴增加10个像素**

**label3.pack(padx = 10) #x轴增加10个像素**



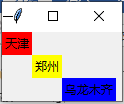
### 1.4.2 grid方法

Grid网格的意思，有两个属性row(行)，column(列)。

label1.grid(row = 1,column = 0)

label2.grid(row = 2,column = 1)

label3.grid(row = 3,column = 2)



Grid中的columnspan(跨列) rowspan(跨行)

创建9个标签：

from tkinter import \*

root = Tk()

label1 = Label(root,bg = "#FF33FF",text = "标签1")

label2 = Label(root,bg = "#666699",text = "标签2")

label3 = Label(root,bg = "#3399CC",text = "标签3")

label4 = Label(root,bg = "#6600FF",text = "标签4")

label5 = Label(root,bg = "#0000FF",text = "标签5")

label6 = Label(root,bg = "#FF6600",text = "标签6")

label7 = Label(root,bg = "#33FF33",text = "标签7")

label8 = Label(root,bg = "#CCCC66",text = "标签8")

label1.grid(row =1,column = 0)

label2.grid(row = 1,column = 1)

label3.grid(row = 1,column = 2)

label4.grid(row = 1,column = 3)

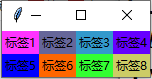
label5.grid(row = 2,column = 0)

label6.grid(row = 2,column = 1)

label7.grid(row = 2,column = 2)

label8.grid(row = 2,column = 3)

效果如下图所示：



label1.grid(row = 1,column = 0)

label2.grid(row = 1,column = 1,columnspan = 2)

label4.grid(row = 1,column = 3)

label5.grid(row = 2,column = 0)

label6.grid(row = 2,column = 1)

label7.grid(row = 2,column = 2)

label8.grid(row = 2,column = 3)效果如下图所示：



label1.grid(row = 1,row = 2,column = 0)

label2.grid(row = 1,column = 1)

label3.grid(row = 1,column = 2)

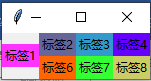
label4.grid(row = 1,column = 3)

label6.grid(row = 2,column = 1)

label7.grid(row = 2,column = 2)

label8.grid(row = 2,column = 3)

效果如下图所示：



### 1.4.3 place方法

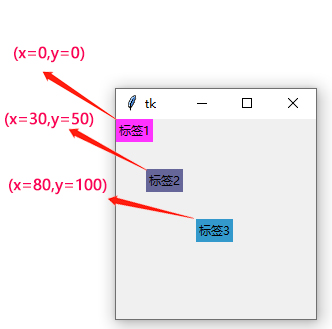
指定插入标签的具体位置，表示控件的具体位置，单位是像素。

label1.place(x = 0,y = 0)

label2.place(x = 30,y = 50)

label3.place(x = 80,y = 100)

效果如下图所示：



# 四、Entry（单行输入文本）

## 1.1 第一个Entry程序

from tkinter import \*

root = Tk()

Entry(root,text = 'input your text here').pack()

root.mainloop()

#上面的代码目的是创建一个Entry对象，并在Entry上显示'input your text here',运行此代码，并没有看到文本的显示，由此可知与Lable和Button不同，Entry的text属性不可以设置Entry的文本

在Entry中设定初始值，使用textvariable将变量与Entry绑定'''

from tkinter import \*

root = Tk()

e = StringVar()

entry = Entry(root,textvariable = e)

e.set('input your text here')

entry.pack()

root.mainloop()

#上面的例子中将变量e与Entry绑定，然后将e的值设置为'input your text here'，程序运行时的初始值便设置了。

## 1.2 entry的属性介绍

|  |  |
| --- | --- |
| 1 | **bg**  输入框背景颜色 |
| 2 | **bd**  边框的大小，默认为 2 个像素 |
| 3 | **cursor**  光标的形状设定，如arrow, circle, cross, plus 等 |
| 4 | **font**  文本字体 font=('Arial', 14) 和字体的大小 |
| 5 | **fg**  文字颜色。 |
| 6 | **justify**  和按钮标签中的justify一样的：属性有left, right, center |
| 7 | **relief**  和按钮的一致可选的有：FLAT、SUNKEN、RAISED、GROOVE、RIDGE。默认为 FLAT。 |
| 8 | **selectbackground selectborderwidth selectforeground**  选中文字的背景颜色 选中文字的背景边框宽度 选中文字的颜色 |
| 9 | **show**  指定文本框内容显示为字符，值随意，满足字符即可。如密码可以将值设为 show="\*" |
| 10 | **state**  默认为 state=NORMAL,只读和可写，值分别为：disabled / normal |
| 11 | **textvariable**  文本框的值，是一个StringVar()对象,如果要显示默认的输入，需要通过StringVar()对象set一下 |
| 12 | **width**  文本框宽度 |

## 1.3 entry的方法介绍

|  |  |
| --- | --- |
| 1 | **delete ( first, last=None )**  删除文本框里直接位置值  entry.delete(10) # 删除索引值为10的值  entry.delete(10, 20) # 删除索引值从10到20之前的值  entry.delete(0, END) # 删除所有值 |
| 2 | **get()**  获取文件框的值 |
| 3 | **icursor ( index )**  将光标移动到指定索引位置，只有当文框获取焦点后成立 |
| 4 | **insert ( index, s )**  向文本框中插入值，index：插入位置，s：插入值 |
| 6 | **select\_adjust ( index )**  选中指定索引和光标所在位置之间的值 |
| 9 | **select\_present()**  判断是否有选择，返回 true，否则返回 false。 |
| 10 | **select\_range ( start, end )**  选中指定索引位置的值，start(包含) 为开始位置，end(不包含) 为结束位置start必须比end小 |
| 11 | **select\_to ( index )**  选中指定索引与光标之间的值 |

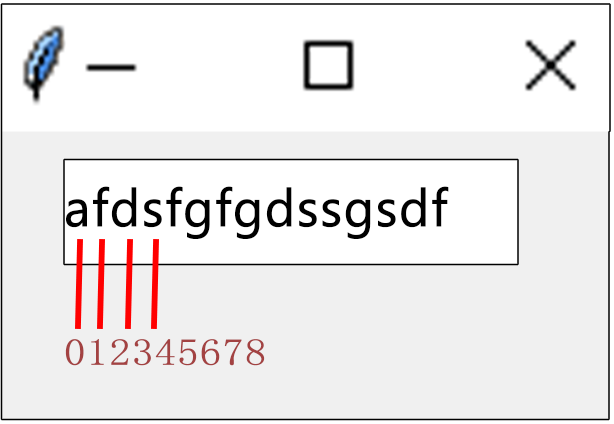
from tkinter import \*

root = Tk()

# 删除输入框中的字符串

def deleteEntry():

entry.delete(0, 1)



#获取值输入框中的字符串

def getEntry():

print(entry.get())



#将光标移动的索引位置

def icursorEntry():

entry.icursor(2)



# 插入输入框中字符串

def insertEntry():

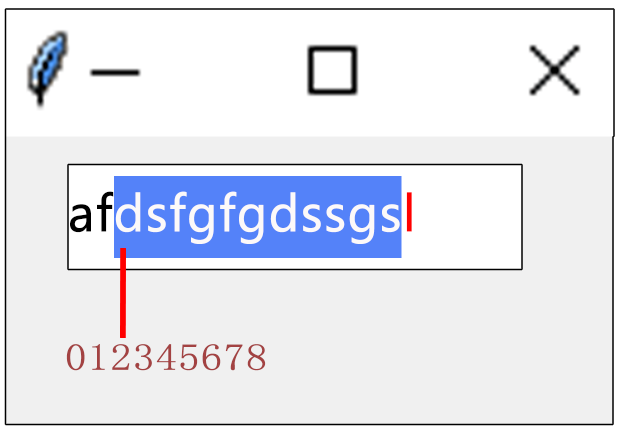
entry.insert(0, '你好')



# 选中指定索引和光标所在位置之前的值

def select\_adjustEntry():

entry.select\_adjust(2)



# 判断是否有选择，返回布尔类型

def select\_presentEntry():

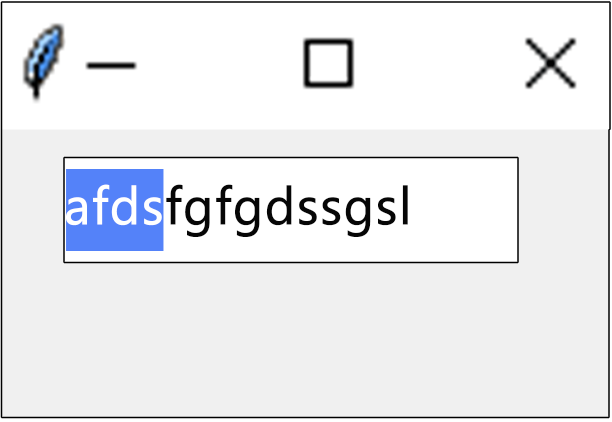
print(entry.select\_present())



# 选中指定索引位置的值

def select\_rangeEntry():

entry.select\_range(0,4)



# 返回指定索引与光标的位置

def select\_toEntry():

entry.select\_to(4)



content = StringVar()

entry = Entry(root, textvariable=content)

entry.pack()

Button(root, text='删除', command=deleteEntry).pack()

Button(root, text='获取', command=getEntry).pack()

Button(root, text='移光标', command=icursorEntry).pack()

Button(root, text='插入', command=insertEntry).pack()

Button(root, text='选择', command=select\_adjustEntry).pack()

Button(root, text='是否选则', command=select\_presentEntry).pack()

Button(root, text='指定值', command=select\_rangeEntry).pack()

Button(root, text='指定索引与光标值', command=select\_toEntry).pack()

root.mainloop()

# 五、Text（多行输入文本）

Text 控件用来显示多行文本.，功能强大、灵活，还可以用于简单的文本编辑器、网页浏览器，可以显示网页链接, 图片, HTML页面。Text控件的高度和宽度是字符个数。

## 1.1 第一个Text程序

from tkinter import \*

root = Tk()

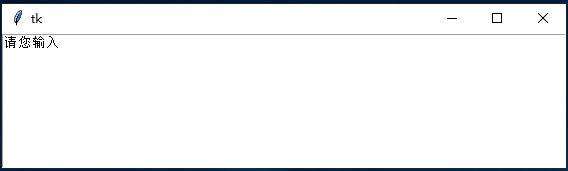
text = Text(root, height=10, width=80)

text.pack()

text.insert(END, "请您输入\n")

root .mainloop()

测试效果如下：



## 1.2 为Text设置滚动条

Tkinter 提供了 Scrollbar() 方法来实现滚动条

**from** tkinter **import** \*  
*#创建窗体对象*root = Tk()  
*#创建滚动条对象，所创建的滚动条要建立在哪个窗体里。*s = Scrollbar(root)  
*#创建多行文本对象*text = Text(root, height=10, width=80,yscrollcommand=s.set)  
*#布局函数，side停靠在窗体的哪个位置，  
#fill填充X方向、Y方向、Both水平和垂直方向、None*s.pack(side=RIGHT, fill=Y)  
poem = **"""轻轻的我走了，  
正如我轻轻的来  
我轻轻的招手，  
作别西天的云彩。  
那河畔的金柳，  
是夕阳中的新娘；  
波光里的艳影，  
在我的心头荡漾。  
软泥上的青荇，  
油油的在水底招摇；  
在康河的柔波里，  
我甘心做一条水草！  
那榆荫下的一潭，  
不是清泉，是天上虹；  
揉碎在浮藻间，  
沉淀着彩虹似的梦。  
寻梦？撑一支长篙，  
向青草更青处漫溯；  
满载一船星辉，  
在星辉斑斓里放歌。  
但我不能放歌，  
悄悄是别离的笙箫；  
夏虫也为我沉默，  
沉默是今晚的康桥！  
悄悄的我走了，  
正如我悄悄的来；  
我挥一挥衣袖，  
不带走一片云彩。  
"""**text.insert(END,poem)  
text.pack(side=RIGHT)  
*#调用config函数让两者产生关联*s.config()  
root.mainloop()

## 1.3 在Text中插入按钮和图片组件

### 1.3.1 插入按钮组件

from tkinter import \*

root = Tk()

#创建text文本框控件

text1 = Text(root,height = 10,width = 80)

text1.pack()

#创建按钮控件

def clickfunc():

print("你点击我了")

button = Button(root,text = "点击我",command = clickfunc)

button.pack()

#将按钮插入到文本控件中

text1.window\_create(INSERT,window=button)

root.mainloop()

### 1.3.2 插入图片组件

#创建一个图片文件

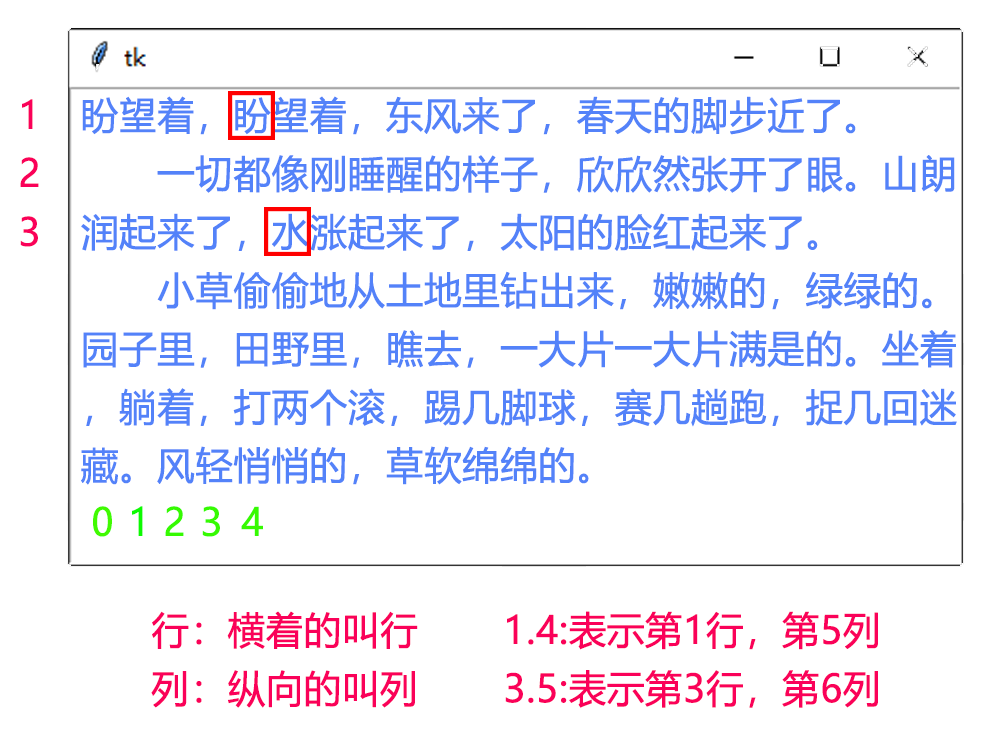
photo = PhotoImage(file='py.png')

#将图片文件插入

text1.image\_create(END,image=photo)

### 1.3.3 多行文本字符处理

**行号以1开始 列号以0开始**



通过行和列可以指定平面内具体位置的字，两个点可以指定某一区域的字。

text1.get(1.5,2.8):指从第1行第6列那个字截取到第2行，第9列的那个字，但不包含第9列的那个字。

# 六、Radiobutton（选项按钮）

## 1.1 第一个Radiobutton选项按钮

我们已经学习了那么多的组件，你会发现组件都有很多相似特征，比如都是类，都需要创建对象，大多数属性都可以进行公用。公共格式如下：

Zujian = Zujian(父对象，options)而options有很多共性。

radiobutton = Radiobutton(root)

from tkinter import \*

root = Tk()

def printselect():

num = v.get()

if num == 1:

print("你是男孩")

else:

print("你是女孩")

v = IntVar() #选项按钮绑定的变量

v.set(1) #默认第一个选项

rboy = Radiobutton(root,text = "男孩",variable =v ,value = 1,command = printselect)

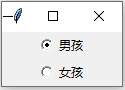
rboy.pack()

rgirl = Radiobutton(root,text = "女孩",variable =v,value = 2,command = printselect)

rgirl.pack()

root.mainloop()

效果如下：



## 1.2 字典与选项按钮

from tkinter import \*

root = Tk()

#通过点击选项，打印输出字典的值（根据字典的键取值）

def printselect():

print(persons[v.get()])

Label(root,text = "选择你喜欢的人物角色").pack()

persons = {0:"孙悟空",1:"猪八戒",2:"唐玄奘",3:"沙和尚",4:"白龙马"}

v = IntVar() #选项按钮绑定的变量

v.set(0) #默认选择变量为0的组件。

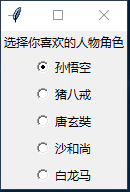
for key,value in persons.items():

rperson = Radiobutton(root,text = value,variable =v ,value = key,command = printselect)

rperson.pack()

root.mainloop()

效果如下：

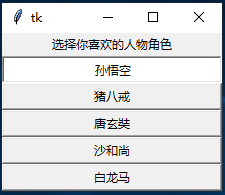


## 1.3 盒子选项按钮

默认情况下就是上图所示指示灯的形式，如果使用盒子选项按钮，仅仅是改变一下形式其他都不用改变。

所以设置Radiobutton的两个options属性就行：Indicatoron指示灯和width宽度

indicatoron = 0, width = 30 效果如下图所示：



## 1.4 图像选项按钮

简单思路：我们将文字对象更改为图片对象就行了

图片对象原来讲解1.2Label图片属性的时候已经讲解过了：

bm = PhotoImage(file = r'图片文件全路径')

from tkinter import \*

root = Tk()

def printselect():

bm = PhotoImage(file = r'图片文件全路径')

bm = PhotoImage(file = r'图片文件全路径')

Label(root,text = "选择你喜欢的图片").pack()

#将所有的图片对象放在列表中

images = []

for i in range(3):

images.append(PhotoImage(file = str(i+1)+".png"))

v = IntVar() #选项按钮绑定的变量

v.set(0) #默认第一个选项

#以此创建Radiobutton对象。

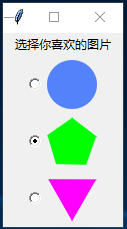
for img in range(len(images)):

rperson = Radiobutton(root,image = images[img],variable =v ,value = img,command = printselect)

rperson.pack()

root.mainloop()

效果如下图所示：



# 七、Checkbutton（复选框）

## 1.1 第一个checkbutton复选框

#创建一个Checkbutton,显示文本为"python"

from tkinter import \*

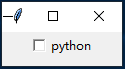
root = Tk()

Checkbutton(root,text = 'python').pack()

root.mainloop()

我们可以通过command设置它的回调函数，且不管它处在什么状态下，都会执行。

执行效果如下：



from tkinter import \*

root = Tk()

def selectfunc():

for key,value in selects.items():

if value.get() == True:

print(food[key])

Label(root,text = "选择你喜欢的食物").grid(row = 0)

#创建食物的字典

food = {0:"蛋炒饭",1:"红烧肉",2:"盖浇饭",3:"油泼面",4:"烧烤",5:"大盘鸡",6:"清蒸鱼"}

#创建字典被选的项目

selects = {}

#以此创建Checbutton对象。

for i in range(len(food)):

selects[i] = BooleanVar()

Checkbutton(root,text = food[i] ,variable =selects[i] ).grid(row = i+1,sticky = W)

#grid为组件对象的网络布局，row行，

#variable 记录你所选择的变量就是BooleanVar()，并且将你所选择的变量放在字典中，它的get

#方法返回的结果就是布尔类型，选中的布尔类型为True

Button(root,text = "确定",command = selectfunc).grid(row = len(food)+1)

root.mainloop()

程序运行的效果如下图所示：



# 八、Frame（容器控件）

## 1.1 Frame容器控件的介绍

Frame是一个容器控件，方便与我们管理控件组件。它也是一个组件，所以具备组件的特性，但它又是一个比较特殊的组件，是一个容器组件。

Frame(父对象，options,….) 父对象表示将这个框架建立在哪个父对象里。下面是options的参数：

bg (background)：背景颜色

bd (borderwidth): 边界宽度，默认为2个像素

cursor:鼠标光标在框架上的光标形状

height：框架的高度，单位是像素

width:框架的宽度，单位像素

highlightbackground:框架没有取得焦点的颜色；

highlightcolor:当框架取得焦点时的颜色；

highlighthickness:当框架取得焦点时的厚度；

relief:框架样式

## 1.2 创建Frame容器控件

from tkinter import \*

root = Tk()

for f in ["#FF69B4","#EE82EE","#00008B"]:

Frame(root,bg = f,height = 30,width = 200).pack()

root.mainloop()



## 1.3 在容器中创建组件

root相对于Frame控件是父控件，Button如果在容器中，那么Frame容器控件相对于Button是父控件。

F = Frame(root,…)

b = Button(F,…)

#创建两个框架，第1个框架放置2个按钮，第2个框架放置2个标签

from tkinter import \*

root = Tk()

#建立上层框架

f\_upper = Frame(root,bg = "#FF1493")

f\_upper.pack()

button1 = Button(f\_upper,text = "按钮1").pack(side = LEFT,padx = 4,pady =4)

button2 = Button(f\_upper,text = "按钮2").pack(side = LEFT,padx = 4,pady =4)

#建立下层框架

f\_lower = Frame(root,bg = "#87CEEB")

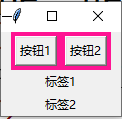
f\_lower.pack()

label1 = Label(f\_lower,text = "标签1").pack()

label2 = Label(f\_lower,text = "标签2").pack()

root.mainloop()

运行效果如下：



# 九、Toplevel（窗口控件）

## 1.1 Toplevel窗口控件的介绍

和Frame类似，但是它能产生独立的窗口，有自己的标题和边框。

Toplevel(options,…)

options的常用方法有：

bg(bakcground)：背景色

bd(borderwidth) :组件边框厚度，默认2

cursor:当鼠标光标在Toplevel上的形状；

fg:文字前景颜色；

font:字形；

height：窗体高度

width：窗体宽度

## 1.2 Toplevel窗口控件案例

代码如下：

from tkinter import \*

root = Tk()

t = Toplevel()

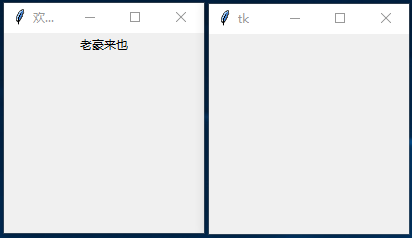
t.title("欢迎来到优频课教育")

t.geometry("200x200")

Label(t,text = "老豪来也").pack()

root.mainloop()

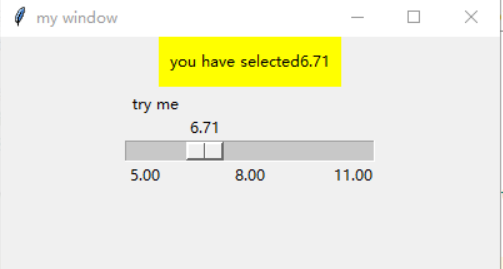
效果显示如下



注意：如果关闭了Toplevel窗口，root窗口继续可以使用，如果关闭了root窗口，Toplevel窗口也将关闭。

# 十、Scale（尺度条控件）

## 1.1 Scale尺寸条属性的介绍



上图所示的就是一个尺度条，通过左右移动改变数值。当然也可以上下移动。

Scale(父对象，options,..) 父对象，表示这个尺度条要建立在哪个父对象里。

|  |  |
| --- | --- |
| 1 | activebackground  鼠标光标在尺寸条上的背景颜色 |
| 2 | **bg（background）**  背景颜色 |
| 3 | **bd（borderwidth）**  边界宽度 |
| 4 | **command**  **用户更改数值会自动调用此函数，移动的时候会自动触发事件** |
| 5 | **cursor**  鼠标在尺寸条上的光标的形状，设定，如arrow, circle, cross, plus 等 |
| 6 | digits  尺度数值，读取需要使用IntVar,DoubleVar、StringVar变量类型读取。 |
| 7 | **fg**  文字颜色。 |
| 8 | **font**  **字体** |
| 9 | **from\_**  **尺度条范围值得初始值** |
| 10 | **highlightbackground**  **尺度条取得焦点时的背景颜色** |
| 11 | **highlightcolor**  **尺度条获得焦点时的颜色** |
| 12 | **label**  默认没有标签文字，如果尺度条是水平的则此标签出现在左上角，如果是垂直的出现在右上角 |
| 13 | length  默认是100像素 |
| 14 | orient(确定方位)  默认是水平的，HROIZONTAL—水平 VERTICAL—垂直 |
| 15 | relief  默认FLAT，更改边框样式 |
| 16 | repeatdelay  可设置按住尺寸条多久后才可以，移动此尺寸条，单位是ms，默认300 |
| 17 | resolution  每次更改的数值，比如：from\_ = 1.0 to = 3.0 resolution = 0.5 尺度1.0 1.5 2.0 2.5 3 |
| 18 | showvalue  正常会显示尺度条的目前值，如果设置为0不显示 |
| 19 | takefocus  正常时此刻度条可以循环取得焦点，如果设置为0无法取得焦点 |
| 20 | tickinterval  尺度条的标记刻度，比如：from\_ = 1.0 to = 3.0 tickinterval= 0.5 尺度1.0 1.5 2.0 2.5 3 |
| 21 | variable  设置或取得目前选取刻度的值，值通常为IntVar StringVar |
| 22 | width  对于垂直尺寸条，它是宽度，对于水平尺寸条，它是高度。 |

## 1.2 第一个Scale尺寸条案例

from tkinter import \*

root = Tk()

scale = Scale(root,

from\_ = 0, #起始值

to = 20, #终点值

troughcolor = "red", #设置槽的颜色

width = "30", #槽的高度

tickinterval = 2, #刻度

label = "我的尺度", #设置标尺标签值

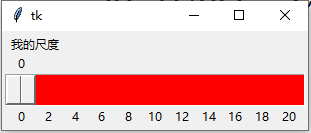
length = 300, #设置标尺长度

orient = HORIZONTAL) #设置标尺的水平方向

scale.pack()

root.mainloop()

执行效果如下：

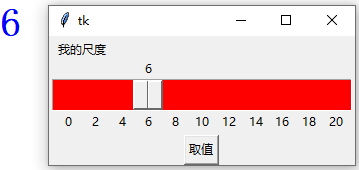


scale.set(3) 我们可以通过set方法直接设置刻度的值；也可以通过get方法获取它的值。通过一个按钮进行获取值。

def getvalue():

print(scale.get())

Button(root,text = "取值",command = getvalue).pack()



## 1.3 设计一个颜色调试器

from tkinter import \*

root = Tk()

root.title("你好")

root.geometry("220x500")

color = "#DC143C"

def change\_color(source):

red = r\_scale.get()

green = g\_scale.get()

blue = b\_scale.get()

#x表示16进制输出，02表示不足两位补0，%表示格式化输入的意思

color = "#%02x%02x%02x"%(red,green,blue)

frame.config(bg = color)

#创建尺度对象

r\_scale = Scale(root,from\_ = 0,to = 255, tickinterval = 50,

label = "red",length = 200,orient = HORIZONTAL,command = change\_color)

g\_scale = Scale(root,from\_ = 0,to = 255, tickinterval = 50,

label = "green",length = 200,orient = HORIZONTAL,command = change\_color)

b\_scale = Scale(root,from\_ = 0,to = 255, tickinterval = 50,

label = "blue",length = 200,orient = HORIZONTAL,command = change\_color)

#将尺度对象放入列表

scales = [r\_scale,g\_scale,b\_scale]

#通过循环

for i in range(len(scales)):

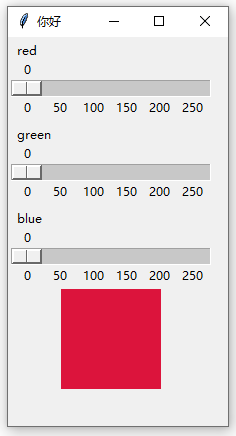
scales[i].grid(row = i+1,column = 0)

frame = Frame(root,bg = color,height = 100,width = 100)

frame.grid(row = 4,column = 0)

root.mainloop()

执行效果如下：



## 1.4 colorchooser颜色选项卡

在tkinter模块中有一个colorchooser颜色选项模块，该模块调用askcolor()方法，就可以开启颜色对话框模块，我们就能选择各种各样的模块。

from tkinter import \*

from tkinter.colorchooser import \*

root = Tk()

def colorful():

#调出颜色选项卡对象

color = askcolor()

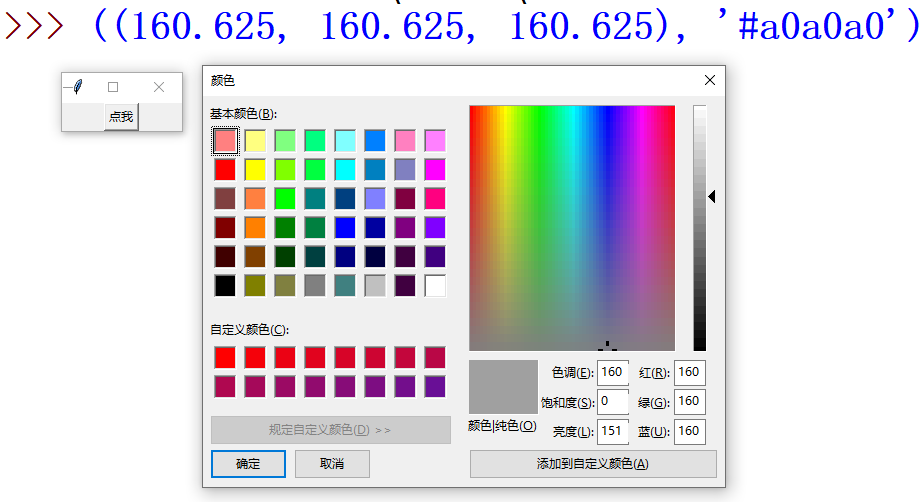
print(color)

button = Button(root,text = "点我" ,command = colorful)

button.pack()

root.mainloop()

效果图入下所示：



# 十一、事件和绑定

## 1.1 鼠标事件绑定第1个案例

用户化界面就是以事件驱动作为导向，就像我们可以将所有的组件都显示在容器容器中，但是我们要对容器中的组件进行各种各样的操作，用户要和界面进行交互，就构成了各种各样的事件。

from tkinter import \*

root = Tk()

def click\_func(event):

print(event.x,event.y)

frame = Frame(root,width = 200,height = 300)

#建立绑定事件

"""

注意事项：

组件调用bind方法，传递2个参数

参数1：事件类型；

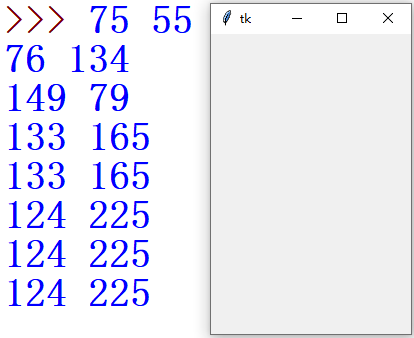
参数2：事件所执行的函数，并且函数需要传递事件参数，event的形参可以更改

"""

frame.bind("<Button-1>",click\_funtcuntcuntcuntc

frame.pack()

程序执行效果如下图所示：



## 1.2 控件事件一览表

|  |
| --- |
| 鼠标相关事件 |
| <Button-1> |
| 一个鼠标点击事件，（1代表左键，2代表中键，3代表右键，4代表滑轮向上滚动，5表示向下滚动）。当你在widget点击鼠标按钮，tkinter会自动捕获并触发event,当按键别抬起的时候才会执行handler |
| <B1-Motion> |
| 鼠标拖动事件，（1代表左键，2代表中键，3代表右键） |
| <ButtonRelease-1> |
| 鼠标按下之后释放（1代表左键，2代表中键，3代表右键） |
| <Double-Button-1> |
| 双击鼠标（1代表左键，2代表中键，3代表右键） |
| <Enter> |
| 这里是鼠标指针进入到widget里，并不是代表按下键盘上的Enter键。 |
| <Leave> |
| 这里是鼠标光标离开widget |
| 键盘相关事件 |
| FocusIn |
| 键盘焦点进入widget控件 |
| FocusOut |
| 键盘焦点离开widget控件 |
| <Return> |
| 按下enter键所有的键都可以被绑定 |
| <Key> |
| 按下的键被存储在event对象中 |
| 控件相关事件 |
| <Configure> |
| 更改Widget控件大小位置，新控件大小的width与height会存储在event对象内。 |