**tkinter介绍**

tkinter是python自带的GUI库，是对图形库TK的封装 tkinter是一个跨平台的GUI库，开发的程序可以在win，linux或者mac下运行

除此之外还存在很多图形库，例如

pythonWin 仅适合window的界面编程库

wxPython 第三方界面编程库

**组件概念**

一个窗口中任意内容都可以称之为一个组件

tkinter的组件包含以下几种：

按钮组件

Button 按钮组件

RadioButton 单选框组件

CheckButton 选择按钮组件

Listbox 列表框组件

文本输入框组件

Entry 单行文本框组件

Text 多行文本框组件

标签组件

Label 标签组件，可以显示图片和文字

Message 标签组件，可以根据内容将文字换行

菜单组件

Menu 菜单组件

MenuButton 菜单按钮组件，可以使用Menu代替

滚动条组件

scale 滑块组件

Scrollbar 滚动条组件

其他组件

Canvas 画布组件

Frame 框架组件，将多个组件编组

Toplevel 创建子窗口容器组件

**创建简单的窗口**

import tkinter

#生成主窗口对象

root = tkinter.Tk()

#保持主窗口一直消息循环中。。

root.mainloop()

**带有组件的窗口**

import tkinter

#生成主窗口对象

root = tkinter.Tk()

#创建标签 并且添加到主窗口中

label = tkinter.Label(root,text = '爷来了')

label.pack()

#创建按钮，并且添加到主窗口中

btn1 = tkinter.Button(root,text = '按钮1')

btn1.pack()

btn2 = tkinter.Button(root,text = '按钮2')

btn2.pack()

#保持主窗口一直消息循环中。。

root.mainloop()

**组件布局**

组件布局一共三种方式：

pack() 按照方位布局

place() 按照坐标布局

grid() 按照网格布局

**1.pack布局方法**

所有的Tkinter组件都包含专用的几何管理方法，这些方法是用来组织和管理整个父配件区中子配件的布局的。Tkinter提供了截然不同的三种几何管理类：pack、grid和place。

pack几何管理采用块的方式组织配件，在快速生成界面设计中广泛采用，若干组件简单的布局，采用pack的代码量最少。pack几何管理程序根据组件创建生成的顺序将组件添加到父组件中去。通过设置相同的锚点（anchor）可以将一组配件紧挨一个地方放置，如果不指定任何选项，**默认在父窗体中自顶向下添加组件。**

**pack()布局的通用公式为：**

组件对象.pack(设置, …)

|  |  |  |
| --- | --- | --- |
| 名称 | 描述 | 取值范围 |
| expand | 当值为“yes”时，side选项无效。组件显示在父配件中心位置；若fill选项为”both”,则填充父组件的剩余空间。 | “yes”, 自然数, “no”, 0   （默认值为“no”或0） |
| fill | 填充x(y)方向上的空间，当属性side=”top”或”bottom”时，填充x方向；当属性side=”left”或”right”时，填充”y”方向；当expand选项为”yes”时，填充父组件的剩余空间。 | “x”, “y”, “both”  (默认值为待选) |
| ipadx, ipady | 组件内部在x(y)方向上填充的空间大小，默认单位为像素，可选单位为c（厘米）、m（毫米）、  i（英寸）、p（打印机的点，即1/27英寸），用法为在值后加以上一个后缀既可。 | 非负浮点数  （默认值为0.0） |
| padx, pady | 组件外部在x(y)方向上填充的空间大小，默认单位为像素，可选单位为c（厘米）、m（毫米）、  i（英寸）、p（打印机的点，即1/27英寸），用法为在值后加以上一个后缀既可。 | 非负浮点数  （默认值为0.0） |
| side | 定义停靠在父组件的哪一边上。 | “top”, “bottom”, “left”, “right”  （默认为”top”） |
| before | 将本组件于所选组建对象之前pack，类似于先创建本组件再创建选定组件。 | 已经pack后的组件对象 |
| after | 将本组件于所选组建对象之后pack，类似于先创建选定组件再本组件。 | 已经pack后的组件对象 |
| in\_ | 将本组件作为所选组建对象的子组件，类似于指定本组件的master为选定组件。 | 已经pack后的组件对象 |
| anchor | 相对于摆放组件的位置的对齐方式，左对齐”w”，右对齐”e”，顶对齐”n”，  底对齐”s” | “n”, “s”, “w”, “e”, “nw”, “sw”, “se”, “ne”, “center”  (默认为” center”) |

**注：以上选项中可以看出expand、fill和side是相互影响的。**

**pack类提供了下列函数：**

|  |  |
| --- | --- |
| 函数名 | 描述 |
| slaves() | 以列表方式返回本组件的所有子组件对象。 |
| propagate(boolean) | 设置为True表示父组件的几何大小由子组件决定（默认值），反之则无关。 |
| info() | 返回pack提供的选项所对应得值。 |
| forget() | Unpack组件，将组件隐藏并且忽略原有设置，对象依旧存在，可以用pack(option, …)，将其显示。 |
| location(x, y) | x, y为以像素为单位的点，函数返回此点是否在单元格中，在哪个单元格中。返回单元格行列坐标，(-1, -1)表示不在其中。 |
| size() | 返回组件所包含的单元格，揭示组件大小。 |

**2.grid布局方法**

grid几何管理采用类似表格的结构组织配件，使用起来非常灵活，用其设计对话框和带有滚动条的窗体效果最好。grid采 用行列确定位置，行列交汇处为一个单元格。每一列中，列宽由这一列中最宽的单元格确定。每一行中，行高由这一行中最高的单元格决定。组件并不是充满整个单 元格的，你可以指定单元格中剩余空间的使用。你可以空出这些空间，也可以在水平或竖直或两个方向上填满这些空间。你可以连接若干个单元格为一个更大空间， 这一操作被称作跨越。创建的单元格必须相临。

**grid()布局的通用公式为：**

组件对象.grid(option, …)

**grid类提供了下列设置属性：**

|  |  |  |
| --- | --- | --- |
| 名称 | 描述 | 取值范围 |
| column | 组件所置单元格的列号。 | 自然数（起始默认值为0，而后累加） |
| columnspan | 从组件所置单元格算起在列方向上的跨度。 | 自然数（起始默认值为0） |
| ipadx, ipady | 组件内部在x(y)方向上填充的空间大小，默认单位为像素，可选单位为c（厘米）、m（毫米）、  i（英寸）、p（打印机的点，即1/27英寸），用法为在值后加以上一个后缀既可。 | 非负浮点数  （默认值为0.0） |
| padx, pady | 组件外部在x(y)方向上填充的空间大小，默认单位为像素，可选单位为c（厘米）、m（毫米）、  i（英寸）、p（打印机的点，即1/27英寸），用法为在值后加以上一个后缀既可。 | 非负浮点数  （默认值为0.0） |
| row | 组件所置单元格的行号。 | 自然数（起始默认值为0，而后累加） |
| rowspan | 从组件所置单元格算起在行方向上的跨度。 | 自然数（起始默认值为0） |
| in\_ | 将本组件作为所选组建对象的子组件，类似于指定本组件的master为选定组件。 | 已经pack后的组件对象 |
| sticky | 组件紧靠所在单元格的某一边角。 | “n”, “s”, “w”, “e”, “nw”, “sw”, “se”, “ne”, “center”  (默认为” center”) |

**grid类提供了下列函数：**

|  |  |
| --- | --- |
| 函数名 | 描述 |
| slaves() | 以列表方式返回本组件的所有子组件对象。 |
| propagate(boolean) | 设置为True表示父组件的几何大小由子组件决定（默认值），反之则无关。 |
| info() | 返回pack提供的选项所对应得值。 |
| forget() | Unpack组件，将组件隐藏并且忽略原有设置，对象依旧存在，可以用pack(option, …)，将其显示。 |
| grid\_remove () |  |

**3.place布局方法**

这个的几何管理器组织放置在一个特定的位置，在他们的父widget部件.

**place()布局的通用公式为：**

组件对象.place(option, …)

|  |  |  |
| --- | --- | --- |
| 名称 | 描述 | 取值范围 |
| anchor | 描述 | 相对于摆放组件的坐标的位置，请参阅：可能是N，E，S，W，东北，西北，东南或西南，罗盘方向指示的widget的角落，双方默认是净重（部件上左上角） |
| height | 以像素为单位的高度. （绝对布局专用） | 像素 |
| width | 以像素为单位的宽度. （绝对布局专用） | 像素 |
| relheight | 组件相对于窗口的的高度 （相对布局专用） | 0～1 |
| relwidth | 组件相对于窗口的的宽度 （相对布局专用） | 0～1 |
| relx | 水平偏移为0.0和1.0之间浮动，父widget的一小部分的高度和宽度. （相对布局专用） | 0～1 |
| rely | 垂直偏移为0.0和1.0之间浮动，父widget的一小部分的高度和宽度. （相对布局专用） | 0～1 |
| x | 组件距离左上角的x坐标 （绝对布局专用） | 像素 |
| y | 组件距离左上角的y坐标 （绝对布局专用） | 像素 |

**place类提供了下列函数（使用组件实例对象调用）：**

|  |  |
| --- | --- |
| 函数名 | 描述 |
| place\_slaves() | 以列表方式返回本组件的所有子组件对象。 |
| place\_configure(option=value) | 给pack布局管理器设置属性，使用属性（option）= 取值（value）方式设置 |
| propagate(boolean) | 设置为True表示父组件的几何大小由子组件决定（默认值），反之则无关。 |
| place\_info() | 返回pack提供的选项所对应得值。 |
| grid\_forget() | Unpack组件，将组件隐藏并且忽略原有设置，对象依旧存在，可以用pack(option, …)，将其显示。 |
| location(x, y) | x, y为以像素为单位的点，函数返回此点是否在单元格中，在哪个单元格中。返回单元格行列坐标，(-1, -1)表示不在其中 |
| size() | 返回组件所包含的单元格，揭示组件大小。 |

**组件1 按钮(button)**

用于定义gui界面中的按钮组件,

tkinter.Button（用于存放的父组件,属性参数...）

具备以下属性：

anchor 设置按钮中文字的对其方式，相对于按钮的中心位置

background(bg) 设置按钮的背景颜色

foreground(fg) 设置按钮的前景色（文字的颜色）

borderwidth(bd) 设置按钮边框宽度

cursor 设置鼠标在按钮上的样式

command 设定按钮点击时触发的函数

bitmap 设置按钮上显示的位图

font 设置按钮上文本的字体

width 设置按钮的宽度 (字符个数)

height 设置按钮的高度 (字符个数)

state 设置按钮的状态

text 设置按钮上的文字

image 设置按钮上的图片

**组件2 文本框(Entry)和多行文本(Text)**

用于定义页面中文本的单行输入框

#单行文本

tkinter.Entry(用于存放的父组件,属性参数...)

#多行文本

tkinter.Text(用于存放的父组件,属性参数...)

具备以下属性：

background(bg) 设置文本框的背景色

foreground(bg ) 设置文本框的前景色

borderwidth(bd) 设置文本输入框的边框

font 设置文本框中的字体

width 设置文本框的宽度(字符个数）

height 设置文本框的高度(字符个数),仅限于text

state 设置文本框的状态

selectbackground 选中文字时文本框的背景色

selectforeground 选中文字时文字的颜色

show 指定文本框显示的字符，若为\*，则表示为密码框

textvariable 设置文本对应的变量，可以通过修改变量改变文字显示，必须使用tkinter.IntVar() 或者tkinter.StringVar()产生的变量 entry可以使用

**组件3 标签(Lebal)**

标签用语在页面中显示文字或者图片

tkinter.Label(用于存放的父组件,属性参数...)

具备以下属性：

anchor 设置文本相对于标签中心的位置

background 设置标签的背景色

foreground 设置标签的前景色

borderwidth 设置标签的边框宽度

width 设置标签的宽度(字符个数）

height 设置标签的高度(字符个数）

text 设置标签中文本内容

font 设置标签中文字的字体类型

bitmap 设置标签的现实的位图

image 设置标签中显示的图片

justify 是设置标签中多行文本的对其方式

textvariable 设置文本对应的变量，可以通过修改变量改变文字显示，必须使用tkinter.IntVar() 或者tkinter.StringVar()产生的变量

**组件4 菜单(Menu）**

菜单用语在界面中设置菜单，和多级子菜单 在tkinter中，菜单组件的添加与其他组件有所不同。 菜单需要使用所创建的主窗口的 config方法添加到窗口中。

这个小工具的目标是，让我们来创建我们的应用程序，可以通过使用各种菜单。核心功能，提供的方式来创建三个菜单类型：弹出式，顶层，和下拉.

thinter.Menu(用于存放的父组件,属性参数...)

具有以下属性

activebackground 背景颜色，当它在鼠标下时将出现在选择上。

activeborderwidth 指定在鼠标下方选择的边框的宽度。默认值为1像素。

activeforeground 当它在鼠标下方时，将出现在选择上的前景色。

background(bg) 用于不在鼠标下的选择的背景颜色。

borderwidth(bd) 围绕所有选择的边框的宽度。默认值为1。

cursor 当鼠标在选项上方时出现的光标，但仅当菜单已被关闭时显示。

disabledforeground 状态为DISABLED的项目的文本颜色。

font 设置文本的字体

foreground(fd) 用于不在鼠标下的选择的前景色。

postcommand 您可以将此选项设置为过程，并且每当有人启动此菜单时，将调用该过程。

relief 菜单的默认3-D效果为relief = RAISED。

image 在这个menubutton上显示一个图像。

selectcolor 指定在检查按钮和单选按钮中显示的颜色。

tearoff 通常情况下，菜单可以被拆除，如果设置tearoff = 0，菜单将不会有撕纸功能，并且从0位置开始添加选择。

title 通常，撕下菜单窗口的标题将与导致此菜单的menubutton或级联文本相同。如果要更改该窗口的标题，请将标题选项设置为该字符串。

具有以下方法

add\_command（选项） 在菜单中添加一个菜单项。

add\_radiobutton（选项） 创建单选按钮菜单项。

add\_checkbutton（选项） 创建一个检查按钮菜单项。

add\_cascade（选项） 通过将给定的菜单与父菜单相关联来创建新的分层菜单

add\_separator（） 在菜单中添加分隔线。

add（类型，选项） 在菜单中添加一个特定类型的菜单项。

delete（startindex [，endindex]） 删除从startindex到endindex的菜单项。

entryconfig（index，options） 允许您修改由索引标识的菜单项，并更改其选项。

index(item) 返回给定菜单项标签的索引号。

insert\_separator(index) 在index指定的位置插入一个新的分隔符。

invoke（index） 调用与位置索引选择相关联的命令回调。如果一个检查按钮，其状态在设置和清除之间切换; 如果一个单选按钮，那个选择是设置的。

type（index） 返回由index指定的选项的类型：“cascade”，“checkbutton”，“command”，“radiobutton”，“separator”或“tearoff”。

万能实例:

from tkinter import \*

root = Tk()

#定义点击菜单触发的方法

def hello():

print("hello!")

#创建总菜单

menubar = Menu(root)

# 创建一个下拉菜单，并且加入文件菜单

filemenu = Menu(menubar)

#创建下来菜单的选项

filemenu.add\_command(label="Open", command=hello)

filemenu.add\_command(label="Save", command=hello)

#创建下拉菜单的分割线

filemenu.add\_separator()

filemenu.add\_command(label="Exit", command=root.quit)

#将文件菜单作为下拉菜单添加到总菜单中，并且将命名为File

menubar.add\_cascade(label="File", menu=filemenu)

# 显示总菜单

root.config(menu=menubar)

root.mainloop()

**组件5 单选框（Radiobutton）与复选框（Checkbutton）**

thinter.Radiobutton(用于存放的父组件,属性参数...)

thinter.Checkbutton(用于存放的父组件,属性参数...)

具有以下属性

anchor 设置组件中文字的对其方式

background(bg） 指定组件的背景色。

borderwidth(bd） 指定组件边框的宽度。

bitmap 指定组件中的位图。

font 指定组件中文本的字体。

foreground(fg) 指定组件的前

height 指定组件的高度。

image 指定组件中的图片。

justify 指定组件中多行文本的对齐方式。

text 指定组件中的文本，可以 使用“\ n” 表示换行。

value 指定组件被选中后(单选框)

onvalue 组件勾选状态值（复选框）

offvalue 组件取消勾选状态的值（复选框）

variable 指定组件所关联的变量。 　需要使用tkinter. IntVar()或者tkinter. StringVar()创建的值

width 指定组件的宽度。

command 设置复选框操作的触发命令（复选框）

**组件6 消息对话框(messagebox)**

用于界面提示成功，失败，警告等相关信息提示框

tkinter。messageBox.FunctionName(title, message [, options])

参数:

FunctionName: 这是相应的消息框函数的名称.

title: 这是在一个消息框，标题栏显示的文本.

message: 这是要显示的文字作为消息.

options: 选项有替代的选择，你可以用它来定制一个标准的消息框。一些可以使用的选项是默认和家长。默认选项是用来指定默认的按钮，如中止，重试，或忽略在消息框中。父选项是用来指定要显示的消息框上的顶层窗口.

你可以使用以下功能之一对话方块:

showinfo() 显示信息对话框

showwarning() 提示警告对话框

showerror () 显示错误对话

askquestion() 问题对话框

askokcancel() 确定还是取消对话框

askyesno () 是不是对话框

askretrycancel () 重试或者取消对话框

除了上述6个标准消息框外，还可以使用tkinter.messagebox.*show函数创建其他类型 的消息框。tkinter.messagebox.*show函数的控制参数如下。

　 default 指定消息框的按钮。

　 icon 指定消息框的图标。

　 message 指定消息框所显示的消息。

　 parent 指定消息框的父组件。

　 title 指定消息框的标题。

　 type 指定消息框的类型。

**使用标准对话框（简单对话/文件对话/颜色对话）**

使用 tkinter. simpledialog 模块、 tkinter. filedialog 模块、 tkinter. colorchooser 模块可以 创建 标准 的 对话框。

tkinter.simpledialog模块 可以创建标准的输入对话框。

tkinter.filedialog模块 可以创建文件打开和保存文件对话框。

tkinter.colorchooser模块 可以创建颜色选择对话框。

**tkinter.simpledialog模块**

tkinter.simpledialog模块可以使用函数创建三种类型的对话框：

tkinter.simpledialog.askstring(标题,提示文字,初始值） 输入字符串

tkinter.simpledialog.askinteger(title,prompt,initialvalue) 输入整数

tkinter.simpledialog.askfloat(title,prompt,initialvalue) 输入浮点型

注意：第三个参数为关键字参数

**tkinter.filedialog模块**

tkinter.filedialog模块中可以使用两种类型的对话框

tkinter.filedialog.askopenfilename(关键字参数传入) 创建标准的【 打开 文件】对话框。

tkinter.filedialog.asksaveasfilename(关键字参数传入) 可以创建标准的【 保存 文件】对话框。

其具有以下几个 相同的可选参数。

　 filetypes 指定文件类型。

　 initialdir 指定默认目录。

　 initialfile 指定默认文件。

　 title 指定对话框标题。

**tkinter.colorchooser模块** tkinter.colorchooser模块中的askcolor函数可以创建标准的【 颜色 选择】 对话框， 其具有以下几个可选参数。

tkinter.colorchooser.askcolor(关键字参数传入)

initialcolor 指定 初始化 颜色。

title 指定 对话框 标题。

**组件6 Frame 框架**

thinter.Menu(用于存放的父组件,属性参数...)

具有以下属性

background(bg) 正常的背景颜色显示在标签和指示器后面。

borderwidth(bd) 指标周围边界的大小。默认值为2像素。

cursor 如果将此选项设置为光标名称（箭头，点等），则鼠标光标将在检查按钮上方更改为该模式。

height 新框架的垂直尺寸。

highlightbackground 当框架没有焦点时，焦点的颜色突出显示。

highlightcolor 当框架具有焦点时，焦点突出显示的颜色。

highlightthickness 焦点亮点的厚度。

relief 使用默认值，relief = FLAT，检查按钮不会从背景中脱颖而出。您可以将此选项设置为任何其他样式

width checkbutton的默认宽度取决于所显示的图像或文字的大小。您可以设置此选项的字符数和checkbutton的，总是有许多字符的空间。

**组件7 Toplevel**

顶层部件的工作，直接由窗口管理器管理的窗口。

thinter.Toplevel ( option, ... )

具有以下属性

background(bg) 窗口的背景颜色。

borderwidth(bd） 边框宽度（以像素为单位）默认为0。

cursor 当鼠标在此窗口中时出现的光标。

class\_ 通常，文本窗口小部件中选择的文本将导出为窗口管理器中的选择。如果您不想要该行为，请设置exportselection = 0。

font 插入窗口小部件的文本的默认字体。

froeground(fg) 在窗口小部件中用于文本（和位图）的颜色。您可以更改标记区域的颜色; 这个选项只是默认值。

height 窗口高度

relief 通常，顶级窗口将不会有三维边框。要获得阴影边框，请将bd选项设置为较大的默认值为零，并将relief选项设置为其中一个常量。

width 所需的窗口宽度。

具有以下方法

deiconify（） 在使用iconify或提取方法后显示窗口。

frame（） 返回系统特定的窗口标识符。

group（window） 将窗口添加到给定窗口管理的窗口组中。

iconify（） 将窗口转换为图标，而不会破坏它。

协议（名称，函数） 将函数注册为将为给定协议调用的回调函数。

iconify（） 将窗口转换为图标，而不会破坏它。

state（） 返回窗口的当前状态。可能的值是正常的，标志性的，撤销的和图标。

transient（[master]）当没有给出参数时，将窗口转换为给定主机或窗口的父级的临时（瞬态）窗口。

withdraw（） 从屏幕中删除窗口，而不会破坏它。

maxsize（width，height） 定义此窗口的最大大小。

minsize（width，height) 定义此窗口的最小大小。

positionfrom（who) 定义位置控制器。

resizable（width，height） 定义resize标志，用于控制窗口是否可以调整大小。

sizefrom（who） 定义大小控制器。

title（string） 定义窗口标题。

**事件绑定**

之前能够触发操作的只有2个组件，一个按钮一个菜单的选项卡 command属性 设置操作对应的函数

鼠标事件类型：

<Button-1> 按下了鼠标左键 <ButtonPress-1>

<Button-2> 按下了鼠标中键 <ButtonPress-2>

<Button-3> 按下了鼠标右键 <ButtonPress-3>

<Enter> 鼠标进入组件区域

<Leave> 鼠标离开组件区域

<ButtonRelease-1> 释放了鼠标左键

<ButtonRelease-2> 释放了鼠标中键

<ButtonRelease-3> 释放了鼠标右键

<B1-Moion> 按住鼠标左键移动

<B2-Moion> 按住鼠标中键移动

<B3-Moion> 按住鼠标右键移动

<Double-Button-1> 双击鼠标左键

<Double-Button-2> 双击鼠标中键

<Double-Button-3> 双击鼠标右键

<MouseWheel> 滚动鼠标滚轮

键盘事件类型:

<KeyPress> 表示按下了键盘

<KeyPress-A> 表示按下键盘A键 A可以设置为其他的按键 这里是区分大小写的 A只会触发大写的A

<Alt-KeyPress-A> 表示同时按下Alt和A键 A可以设置为其他的按键

<Control-KeyPress-A> 表示同时按下Ctrl和A键 A可以设置为其他的按键

<Shift-KeyPress-A> 表示同时按下Shift和A键 A可以设置为其他的按键

<Double-KeyPress-A> 表示双击键盘A键 A可以设置为其他的按键

<Lock-KeyPress-A> 表示开启大写之后键盘A键 A可以设置为其他的按键

<Alt-Control-KeyPress-A> 表示同时按下alt+Ctrl和A键 A可以设置为其他的按键

注意：键盘事件除了entry和text组件其他组件的事件最好绑定在主界面上

事件对象(evt)中包含的信息

x,y 当前触发事件时鼠标相对触发事件的组件的坐标值

x\_root,y\_root 当前触发事件时鼠标相对于屏幕的坐标值

char 获取当前键盘事件时按下的键对应的字符

keycode 获取当前键盘事件时按下的键对应的的ascii码

type 获取事件的类型

num 获取鼠标按键类型 123 左中右

widget 触发事件的组件

width/height 组件改变之后的大小和configure()相关

例：

打印当前触发事件的对象

print(evt.widget)

窗口和组件相关事件类型：

Activate 当中组件由不可以用变为可用时 针对于state的变值

Deactivate 当组件由可用变为不可用时触发

Configure 当组件大小发生变化时触发

Destory 当组件销毁时触发

FocusIn 当组件获取焦点时触发 针对于Entry和Text有效

FocusOut 当组件失去焦点时触发 针对于Entry和Text有效

Map 当组件由隐藏变为显示时触发

UnMap 当组件由显示变为隐藏时触发

Perproty 当窗口属性发生变化时触发

事件绑定函数：

组件.bind(‘事件类型’,自定义事件操作)

为一个组件绑定一个操作

组件.bind\_class(‘哪一类组件 例如Button’,‘事件类型’,自定义事件操作)

为一类组件绑定一个操作

组件.bind\_all(‘事件类型’,自定义事件操作)

为所有组件绑定一个操作 主界面事件绑定

在事件绑定中，为了代码的美观等原因，可以将自定义操作放到一个函数里，以下是函数的声明规则：

格式：

def 函数名(evt) :

操作

讲解：