

Tkinter 8.5 参考 ： Python 的 GUI



摘要

描述了 *Tkinter* 构件为构建图形用户界面 (Gui) 中的 Python 编程语言设置。其中包括覆盖的 *ttk* 主题窗口小部件。

这份出版物是可用在 [Web 窗体](#)，也作为一个 [PDF 文档](#)。请转发到 tcc-doc@nmt.edu 的任何评论。

Tkinter 8.5 参考： Python 的 GUI 1

1. 为 Python 的跨平台图形用户接口生成器.....	5
2. 小的应用程序.....	6
3. 定义	8
4. 布局管理	9
4.1. 方法.grid()	10
4.2. 其他网格管理方法.....	12
4.3. 配置列和行的大小.....	14
4.4. 使根窗口调整大小.....	15
5. 标准属性	17
5.1. 尺寸.....	18
5.2. 坐标系统.....	19
5.3. 颜色.....	20
5.4. 类型的字体.....	21
5.5. 锚.....	23
5.6. 救济样式.....	24
5.7. 位图.....	25
5.8. 游标.....	26
5.9. 图像.....	28
5.10. 几何字符串.....	29
5.11. 窗口名称.....	30
5.12. 帽和加入样式.....	31
5.13. 虚线样式.....	32
5.14. 点画模式匹配.....	33
6. 异常处理	34
7. 构件 Button	35

8. 构件 Canvas	38
8.1. 坐标 Canvas	40
8.2. 显示列表 Canvas.....	41
8.3. 对象 IdCanvas	42
8.4. 标记 Canvas	43
8.5. 参数 Canvas <i>tagOrId</i>	44
8.6. 小部件上的方法 Canvas.....	45
8.7. 弧对象 Canvas.....	54
8.8. 位图对象 Canvas.....	57
8.9. 图像对象 Canvas.....	59
8.10. 线对象 Canvas	60
8.11. oval 对象 Canvas	62
8.12. 多边形对象 Canvas.....	65
8.13. 矩形对象 Canvas.....	68
8.14. 文本对象 Canvas.....	70
8.15. 窗口对象 Canvas.....	72
9. 小部件 Checkbutton	73
10. 小部件 Entry	78
10. 小部件 Entry	84
10.2. 向小部件添加验证 Entry.....	90
11. 小部件 Frame	93
12. 构件 Label	95
13. 小部件 LabelFrame	98
14. 小部件 Listbox	101
14.1. 滚动窗口小部件 Listbox.....	108
15. 小部件 Menu	109
15.1. 项目创建 () 选项 Menuoption.....	114
15.2. 顶级菜单.....	116
16. 小部件 Menubutton	117
17. 小部件 Message	120
18. 小部件 OptionMenu	122
19. 小部件 PanedWindow	123
19.1. 子配置选项 PanedWindow.....	127
20. 小部件 Radiobutton	128
21. 构件 Scale	132
22. 小部件 Scrollbar	136
22.1. 回调 Scrollbar <i>command</i>	140
Tkinter 8.5 参考： Python 的 GUI.....	141
22.2. 连接到另一个小部件 Scrollbar.....	141
23. 小部件 Spinbox	142
24. 小部件 Text	148
24.1. 小部件指数 Text.....	152
24.2. 小部件标记 Text.....	155

24. 4. 窗口小部件 Text.....	156
24. 5. 小部件标签 Text.....	157
24. 6. 在小部件中设置选项卡 Text.....	158
24. 7. 构件撤消/重做堆栈 Text.....	159
24. 8. 小部件上的方法 Text.....	160
25. : 顶级窗口方法 Toplevel	171
26. 普遍小部件的方法.....	175
警告.....	183
27. 外观和选项数据库的标准化.....	188
27. 1. 如何命名窗口小部件类.....	190
27. 2. 如何命名部件实例.....	191
27. 3. 资源规格线.....	192
27. 4. 资源匹配规则.....	194
28. ttk : 主题窗口小部件.....	195
28. 1. 进口 ttk	196
28. 2. ttk 小部件集.....	197
28. 2. ttk 小部件集.....	198
30. ttk . Checkbutton.....	199
31. ttk . Combobox.....	203
32. ttk . Entry.....	205
33. ttk . Frame.....	208
34. ttk . Label.....	210
35. ttk . LabelFrame.....	214
36. ttk . Menubutton.....	217
37. ttk . Notebook.....	220
37. 1. 虚拟事件 ttk 构件. Notebook.....	224
38. ttk . PanedWindow.....	225
39. ttk . Progressbar.....	227
40. ttk . Radiobutton.....	229
41. ttk . Scale.....	232
42. ttk . Scrollbar.....	235
43. ttk . Separator.....	238
44. ttk . Sizegrip.....	239
45. ttk . Treeview.....	240
45. 1. ttk 构件虚拟事件. Treeview.....	251
46. 共同所有 ttk 小部件的方法.....	252
46. 1. 在 ttk 中指定小组件状态.....	254
47. 自定义和创建 ttk 主题和样式.....	255
48. 找到和使用 ttk 主题.....	256
49. 使用和自定义 ttk 样式.....	257
50. ttk 元素层.....	260
50. 1. ttk 布局: 构建一种风格.....	261
50. 2. ttk 样式映射: 动态外观变化.....	264
51. 连接应用程序的逻辑部件.....	267

52. 控制变量： 小部件背后的价值观.....	267
53. 重点： 路由键盘输入.....	271
53.1. 集中在 <i>tk</i> 小部件中.....	273
54. 事件： 对刺激的反应.....	274
54.1. 级别的绑定.....	275
54.2. 事件序列.....	277
54.3. 事件类型.....	278
54.4. 事件修饰符.....	280
54.5. 键名称.....	281
54.6. 写您的处理程序： 类 Event.....	284
54.7. 额外的参数欺骗.....	287
54.8. 虚拟事件.....	289
55. 弹出对话框	290
55.1. 对话框模块 tkMessageBox.....	291
55.2. 模块 tkFileDialog	294
55.3. 模块 tkColorChooser	296

1. 为 Python 的跨平台图形用户接口生成器

Tkinter 是为 Python 设置一个 GUI（图形用户界面）小部件。本文档为 Python 2.7 和 *Tkinter* 8.5 编写运行在 Linux 下的 X 窗口系统。您的版本可能会发生变化。

相关的参考资料：

- 弗雷德里克 Lundh，写 *Tkinter*，有两个版本的他 *Tkinter* 简介：一个 [更完整的 1999 年版](#)和 [2005 年的版本](#)，介绍了几个新的功能。
- [Python 2.7 快速参考](#)：有关 Python 语言的一般信息。
- （大约 1000 行代码），大量工作上所需的应用程序的示例，请参见 [惠：颜色和字体的选择工具](#)。此应用程序的设计演示如何构建您自己的复合部件。

我们先来看 *Tkinter* 的可见部分：创建窗口小部件和安排他们在屏幕上。稍后我们将谈论如何连接的脸——“前面板”——在它背后的逻辑的应用。

2. 小的应用程序

这里有一个微不足道的 *Tkinter* 程序包含只有一个退出按钮：

```
#!/usr/bin/env python

import Tkinter as tk

class Application(tk.Frame):
    def __init__(self, master=None):

        tk.Frame.__init__(self, master)

        self.grid()
        self.createWidgets()

    def createWidgets(self):
        self.quitButton = tk.Button(self, text='Quit',

                                     command=self.quit)

        self.quitButton.grid()

app = Application()

app.master.title('Sample application')

app.mainloop()
```

这条线使脚本自动执行，假定您的系统具有 Python 安装正确。

这条线 *Tkinter* 模块导入程序的命名空间，但将它作为重命名。tk

应用程序类必须从 *Tkinter* 类继承。Frame

调用父类的构造函数。Frame

有必要使应用程序实际上出现在屏幕上。

创建一个标记为“退出”按钮。

将按钮放置在应用程序上。

主要程序从这里开始通过实例化的类。Application

此方法调用设置到“示例应用程序”窗口的标题。

启动应用程序的主循环，等待鼠标和键盘事件。

3. 定义

在我们开始之前，让我们来定义一些常用的术语。

窗口

这个词在不同语境中有不同的含义，但在一般情况下它是指在某个地方在显示屏幕上的矩形区域。

顶级窗口

在屏幕独立存在的窗口。它将装饰与标准帧和控件为您的系统的桌面管理器。您可以在您的桌面上移动它。你可以普遍调整它的大小，虽然您的应用程序可以防止这种情况

小部件

一般术语为任何组成应用程序的图形用户界面中的构建基块。小部件示例： 按钮、 单选按钮、 文本字段、 框架和文本标签。

框架

在 *Tkinter*，小部件是复杂的布局组织的基本单位。框架是可以包含其他控件的矩形区域。Frame

子女、 父母

任何小部件创建时，将创建一个父-子关系。例如，如果您将放到框架内的文本标签，车架是标签的父。

4. 布局管理

稍后我们将讨论的小部件，您的 GUI 应用程序的构建块。如何做小部件被安排在一个窗口中？

虽然有三种不同“几何经理”在 *Tkinter*，作者强烈倾向于几乎一切的几何管理器。此管理器视为一个表的每个窗口或框架 — — 一副强大的行和列。`.grid()`

- 单元格是一行和一系列相交处的区域。
- 每个列的宽度是该列中最宽的单元格的宽度。
- 每行的高度是那一行的最大单元格的高度。
- 对于不填充整个单元格的小部件，您可以指定额外的空间，会发生什么。你可以留下额外的空间构件，外部或伸展要适应它，要么在水平或垂直维度的构件。
- 您可以将多个单元格合并到一个更大的区域，这个过程被称为*跨越*。

当你创建一个小部件时，它不会出现直到你注册它与几何管理器。因此，建设和放置的一个小部件是一个两步过程，是像这样：

```
self.thing = tk.Constructor(parent, ...)
self.thing.grid(...)
```

在哪里是的窗口小部件类像、 等等，而是父部件在哪这孩子小部件正在建造。所有的小部件都有一种方法，你可以使用告诉几何管理器把它放在哪儿。`ConstructorButtonFrameparent.grid()`

4. 1. 方法.grid()

在您的应用程序屏幕上显示一个小部件：*w*

```
w.grid(option=value, ...)
```

此方法注册一个小部件与网格几何管理器——如果你不这样做，小部件将存在内部，但它不会在屏幕上可见。选项，请参阅[表 1，“.grid\(\) 几何管理器参数”](#)。*w*

表 1. .Grid() 几何管理器参数

column	列号，在那里你想要这个小部件网格，从零开始计数。默认值为零。
columnspan	通常一个小部件占据网格中的只有一个单元格。然而，你可以抓住多个单元格的行，并将它们合并到一个更大的单元格，通过将选项设置为单元格数目。例如，将跨列 2、3 和 4 0 行单元格中放置小部件。 <code>columnspanw.grid(row=0, column=2, columnspan=3) w</code>
in_	要注册为一个孩子的一些小部件，请使用。新的父项必须是小部件创建时使用的后裔。 <code>ww₂in_₂=w₂w₂parentw</code>
ipadx	内部填充 x。内其左、右两边小部件添加此维度。
ipady	内部 y 填充。其顶部和底部的边框内构件内添加此维度。
padx	X 填充的外部。向左和右外小部件添加此维度。
pady	外部 y 填充。此维度被添加的上方和下方的小部件。
row	到你想要插入的构件，从 0 开始计数的行号。默认值是下编号较大的未占用的一行。
rowspan	通常一个小部件占据网格中的只有一个单元格。然而，你可以抓住多个相邻单元格的列，将选项设置为单元格抓住数目。此选项可以与选项组合用于抓取的单元格块。例如，会将小部件放在面积由合并 20 的单元格，用 3-6 行号和列号 2-6。 <code>rowspancolumnspanw.grid(row=3, column=2, rowspan=4, columnspan=5) w</code>
sticky	此选项确定如何分配任何额外空间内的单元格不采取处其自然大小的小部件。请参阅下文。

- 如果你不提供一个属性，默认行为是中心部件的单元格中。`sticky`
- 你可以使用 `(右上方)` 在一个角落里的细胞定位构件，`(右下)`，`(左下角)`，或 `(左上角)`。`sticky=tk.NEtk.SETk.SWtk.NW`
- 你可以对一侧的细胞通过使用 `(顶部中心)` 为中心的小部件的位置，`(右侧中心)`，`(底部中心)`，或 `(左中)`。
`sticky=tk.Ntk.Etk.Stk.W`
- 使用垂直拉伸构件，但让它水平居中。`sticky=tk.N+tk.S`
- 使用横向伸展，但让垂直居中。`sticky=tk.E+tk.W`
- 利用横向伸展的小部件，垂直以填充整个单元格。
`sticky=tk.N+tk.E+tk.S+tk.W`
- 其他的组合也将工作。例如，将垂直拉伸小部件并将其放置到墙边西
`(左)`。`sticky=tk.N+tk.S+tk.W`

4. 2. 其他网格管理方法

所有小部件上定义这些网格相关的方法：

`w.grid_bbox(column=None, row=None, col2=None, row2=None)`

返回描述网格系统在小部件中的部分或全部的边界框的 4 元组。返回的前两个数字的区域左上角的 x 和 y 坐标，第二两个数字的宽度和高度。 w

如果你传递的参数，并且参数，返回的边界框描述处的行和列的单元格的区域。如果你还在通过和参数，返回的边界框描述网格从列到包容性，并从区域行到包容性。 `columnrowcol2row2columncol2rowrow2`

例如，返回的边框的四个单元，不是一个。 `w.grid_bbox(0, 0, 1, 1)`

`w.grid_forget()`

此方法使小部件从屏幕上消失。它仍然存在，它只是不可见。您可以使用它，使它再次出现，但它不会记得它的网格选项。 `w.grid()`

`w.grid_info()`

返回其键为 w 选项名称，这些选项的相应值的字典。

`w.grid_location(x, y)`

在给定坐标相对于包含小部件，此方法返回一个元组描述的单元格的网格系统包含的屏幕坐标。 (x, y) (col, row) w

`w.grid_propagate()`

通常情况下，所有的小部件 *传播* 他们的尺寸，意味着他们调整到适合的内容。然而，有时您想要强制一个小部件，以一定的尺寸，不论其内容的大小。若要执行此操作，请调用在哪里是你想要迫使其大小窗口小部件。 `w.grid_propagate(0)` w

`w.grid_remove()`

这种方法是像，但记住其网格选项，所以如果你它再一次，它将使用相同的网格配置选项。 `.grid_forget().grid()`

`w.grid_size()`

返回一个包含的列数和行数的 2 元组分别在的网格系统。 w

`w.grid_slaves(row=None, column=None)`

返回管理小部件的部件的列表。如果不提供任何参数，您将得到一个列表的所有托管的窗口小部件。使用要在一行中选择仅部件的参数或参数用于在一列中选择只的小部件。`wrow=column=`

4. 3. 配置列和行的大小

除非你采取一定的措施，在一个给定的部件内的网格列的宽度将等于其最宽的单元格的宽度和网格行的高度将是其最高的单元格的高度。只有它将放置的位置如果它不能完全满足该单元格的小部件控件上的属性。`sticky`

如果你想要重写此自动调整大小的列和行，父部件包含网格布局上使用这些方法：`w`

`w.columnconfigure(N, option=value, ...)`

在网格布局中内小部件，配置列以便给定了给定。选项，请参阅下表。`wOptionvalue`

`w.rowconfigure(N, option=value, ...)`

在网格布局中内小部件，配置行所以，给定了给定。选项，请参阅下表。`wOptionvalue`

以下是用于配置列和行大小的选项。

表 2. `.Grid()` 几何管理器的列和行配置选项

minsize	列或行的最小大小以像素为单位。如果没有在给定的列或行，它不会出现，即使您使用此选项。
pad	将被添加到给定的行或列以外的列或行中最大的细胞的像素的数目。
weight	<p>若要使列或行的可伸缩，使用此选项，并提供给此列或行的相对权重分配额外的空间时的值。例如，如果一个小部件包含一个网格布局，这些线路将分发四分之三的额外的空间，对第一列和一个第四到第二列： 如果不使用此选项，行或列将不拉伸。<code>w</code></p> <div><code>w.columnconfigure(0, weight=3)</code> <code>w.columnconfigure(1, weight=1)</code></div>

4. 4. 使根窗口调整大小

你想让用户调整你的整个应用程序窗口的大小和分配额外空间之间及其内部的部件？这就需要一些操作并不明显。

它有需要使用技术为行和列的大小管理，所述[第 4.3 节、“配置列和行大小”](#)，让您的小程序件的网格可拉伸。不过，这并不足够。Application

考虑在[第一节 2，“小的应用程序”](#)，其中包含只有一个[退出](#)按钮讨论的琐碎的应用程序。如果您运行此应用程序，并调整窗口的大小，按钮始终保持相同的大小，在窗口中居中。

这里是方法的[小的应用程序](#)中替换版本。在此版本中，[退出](#)按钮总是填满所有可用的空间。`.__createWidgets()`

```
def createWidgets(self):  
  
    top=self.wininfo_toplevel()  
  
    top.rowconfigure(0, weight=1)  
  
    top.columnconfigure(0, weight=1)  
  
    self.rowconfigure(0, weight=1)  
  
    self.columnconfigure(0, weight=1)  
    self.quit = Button(self, text='Quit', command=self.quit)  
  
    self.quit.grid(row=0, column=0,  
                   sticky=tk.N+tk.S+tk.E+tk.W)
```

“顶层窗口”是最外面的窗口在屏幕上。然而，这个窗口不是你的窗口——它是父实例。若要获取的顶级窗口，调用方法在任何部件上在您的应用程序；请参阅[节 26、“通用构件方法”](#)。

`ApplicationApplication.wininfo_toplevel()`

这条线使顶级窗口网格的行 0 可拉伸。

这条线使顶级窗口网格第 0 列可拉伸。

使小程序件的网格的行 0 可拉伸。Application

使小部件的网格的列 0 可拉伸。Application

参数使扩大以填充其单元格的网格的按钮。

`sticky=tk.N+tk.S+tk.E+tk.W`

还有一个更多的改变，必须作出。在构造函数中，更改第二行所示：

```
def __init__(self, master=None):  
    tk.Frame.__init__(self, master)  
    self.grid(sticky=tk.N+tk.S+tk.E+tk.W)  
    self.createWidgets()
```

参数是必要的所以，小部件将扩展以填充其网格单元中的顶级窗口。

`sticky=tk.N+tk.S+tk.E+tk.aWself.grid()`Application

5. 标准属性

我们看看这些小部件之前，让我们看看如何的一些他们共同的属性 —— 例如大小、 颜色和字体 —— 指定。

- 每个插件都有一整套影响其外观和行为的选项 —— 属性，如字体、 颜色、 尺寸、 文本标签和这样。
- 您可以指定选项，当调用小部件的构造函数使用关键字参数如或。
`text='PANIC!' height=20`
- 您已经创建了一个小部件后，您以后可以通过使用的部件的方法更改任何选项。您可以通过使用的部件的方法检索任何选项的当前设置。有关这些方法的更多信息，请参阅[节 26, “通用构件方法”](#)。
`_.config().cget()`

下一个：[5.1. 尺寸](#)

内容：[Tkinter 8.5 参考： GUI 的 Python](#)

以前：[4.4。 制作根窗口调整大小](#)

帮助：[科技计算机中心： 帮助系统](#)

主页：[关于新的墨西哥科技](#)

John W. Shipman

Comments welcome: tcc-doc@nmt.edu

最后更新： 2013 年-12-31 17:59

网址：<http://www.nmt.edu/tcc/help/pubs/tkinter/web/std-attrs.html>

5. 1. 尺寸

在许多不同的单位，可以描述不同长度、 宽度和其他尺寸的窗口小部件。

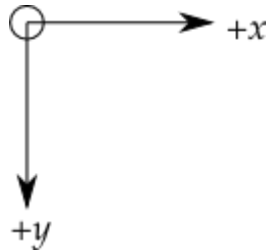
- 如果你将维度设置为一个整数，它被假定为以像素为单位。
- 您可以通过将一个维度设置为一个字符串， 包含数字后面跟指定单位：

表 3。尺寸单位

c	厘米
i	英寸
m	毫米
p	打印机的点 （约 1/72" ）

5.2. 坐标系

最当代的显示系统，每个坐标系统的原点位于其左上角，向右， y 坐标增加底部增加的 x 坐标：



基本单位是像素，且顶端的左边像素的坐标为 $(0, 0)$ 。协调您指定整数则始终表示以像素为单位，但任何坐标可指定为纲的数量；请参阅[第 5.1 条，“维度”](#)。

5. 3. 颜色

有两种方式来 *Tkinter* 中指定的颜色。

- 您可以使用一个字符串，指定红色、 绿色和蓝色在十六进制数字的比例：
：

#rgb	每种颜色的四位
#rrggbb	每种颜色的八位
#rrrrgggbbb	每种颜色的十二位

- 例如，是白色的是黑色，是纯绿色，纯青色（绿色 + 蓝色）。
'#fff' '#000000' '#000fff000' '#00ffff'
- 您还可以使用任何本地定义的标准颜色名称。颜色、 和将始终可用。其他名称可能工作，具体取决于您的本地安装。
'white' 'black' 'red' 'green' 'blue' 'cyan' 'yellow' 'magenta'

5.4. 类型的字体

根据你的平台，可能达三种方式来指定类型样式。

- 作为一个元组的第一个元素是字体的家庭，其次是大小（以磅为单位如果为正数，以像素为单位如果为负数），可以选择后跟一个字符串包含一个或多个样式修饰符、`bold`、`italic`、`underline`、`overstrike`

例子：为 16 磅的 Helvetica 定期；24 点时间为粗斜体。20 像素时代加粗的字体，请使用。`('Helvetica', '16')` `('Times', '24', 'bold italic')` `('Times', -20, 'bold')`

- 您可以通过导入模块，并使用其类构造函数创建一个“字体对象”：`tkFont`

```
import tkFont
...
font = tkFont.Font(option, ...)
```

其选项包括：

family	字体系列名称作为字符串。
size	字体高度，以磅为单位的整数。若要获取字体像素高，请使用。 <i>n</i> - <i>n</i>
weight	'bold' 为一道横杠，为正常体重。'normal'
slant	'italic' 为斜体，为 <code>unslanted</code> 。'roman'
underline	1 为带下划线的文本，为正常。0
overstrike	1 为正常的 <code>overstruck</code> 文本。0

例如，要得到 36 点黑体斜体粗体：

```
helv36 = tkFont.Font(family='Helvetica',
                      size=36, weight='bold')
```

- 如果你在 X 窗口系统下运行，你可以使用任何的 X 字体名称。例如，命名的字体是屏幕上使用好固定宽度字体。使用 `xfontsel` 程序来帮助您选择取悦字体。`'*-lucidatypewriter-medium-r-*-*-*140-*-*-*-*-*'`

若要获得您的平台的所有家庭的可用字体的列表，请调用该函数：

```
tkFont.families()
```

返回值是一个字符串列表。*注：* 您必须在调用此函数之前创建您的根窗口。

在所有对象上定义这些方法：Font

.actual(*option*=None)

如果你不传递任何参数，但你回来字体的实际特性，可能有别于你要求的字典。若要返回的属性的值，将作为参数传递它的名字。

.cget(*option*)

返回的值给定。*option*

.configure(*option*, ...)

使用此方法来更改字体上的一个或多个选项。例如，如果你有一个对象称为，如果您调用，该字体将改为 18pt 倍和任何小部件，使用这种字体也将更改。FonttitleFonttitleFont.configure(family='times', size=18)

.copy()

返回对象的一个副本。Font

.measure(*text*)

一个字符串传递给这个方法，它将返回的字符串将在该字体的宽度的像素数。警告： 有些倾斜的字符可能会扩展在此区域外。

.metrics(*option*)

如果您调用此方法不带任何参数，它返回一个字典的所有 *字体度量*。您可以通过将它的名称作为参数传递检索只是一个度量值。指标包括：

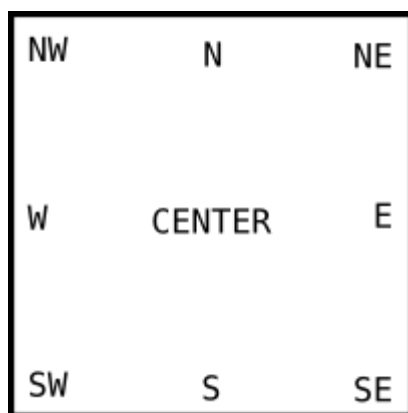
ascent	高度基线和最高的字母上伸部分的顶部之间的像素数。
descent	基线和最低 <i>ascender</i> 底部之间高度的像素数。
fixed	此值是可变宽度字体和等宽字体。01
linespace	总高度的像素数。这是设置固体在给定字体类型的领导。

5.5. 锚

Tkinter 模块定义 锚 常量，你可以使用数控制项目相对于其上下文的放置位置。例如，锚可以指定一个小部件位于什么地方框架内框架时比小部件。

这些常量给出作为指南针点，北起，西部是向左了。我们向我们为此[北半球沙文主义](#)的南半球读者道歉。

锚常数图所示：



例如，如果您创建一个大的框架里面的小部件，并使用选项，小部件将放在框架的右下角。如果您使用相反，小部件将沿顶部边缘为中心。
`anchor=tk. SE``anchor=tk. N`

主槽也用于定义文本相对于参考点的位置。例如，如果您使用作为文本锚点，文本将围绕水平和垂直方向的参考点。锚点将放置的文本，以便参考点吻合（左上角）西北角的文本框中包含文本。锚点将居中文本垂直周围的参照点，通过这一点，在文本框中的左边缘，等等。`tk. CENTER``tk. NW``tk. W`

5.6. 救济样式

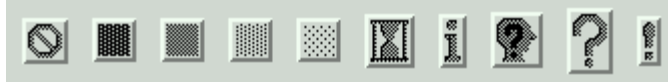
救济风格的一个小部件是指周围的小部件外某些模拟三维效果。下面是一排按钮展示所有可能的救济样式的一个屏幕快照：



这些边框的宽度取决于小部件的选项。上面的图显示了什么他们看起来像 5 像素宽的边框；默认边框宽度为 2。borderwidth

5.7. 位图

在小部件中的选项，这些位图被保证可供使用：bitmap



上面这张图表显示小部件轴承标准位图。从左到右，他们是、 和。
Button'error''gray75''gray50''gray25''gray12''hourglass''info''questhead''question''warning'

您可以使用您自己的位图。(X 位地图) 中的任何文件格式将工作。一个标准的位图名称，使用字符串后跟文件的路径名。`.xbm'` `@'.xbm`

5.8. 游标

有很多不同的鼠标光标。这里显示了它们的名称和图形。根据您的操作系统，精确图形可能会发生变化。

表 4。游标选项值的

 arrow	 man
 based_arrow_down	 middlebutton
 based_arrow_up	 mouse
 boat	 pencil
 bogosity	 pirate
 bottom_left_corner	 plus
 bottom_right_corner	 question_arrow
 bottom_side	 right_ptr
 bottom_tee	 right_side
 box_spiral	 right_tee
 center_ptr	 rightbutton
 circle	 rtl_logo
 clock	 sailboat
 coffee_mug	 sb_down_arrow
 cross	 sb_h_double_arrow
 cross_reverse	 sb_left_arrow
 crosshair	 sb_right_arrow
 diamond_cross	 sb_up_arrow
 dot	 sb_v_double_arrow
 dotbox	 shuttle
 double_arrow	 sizing
 draft_large	 spider
 draft_small	 spraycan

 draped_box	 star
 exchange	 target
 fleur	 tcross
 gobbler	 top_left_arrow
 gumby	 top_left_corner
 hand1	 top_right_corner
 hand2	 top_side
 heart	 top_tee
 icon	 trek
 iron_cross	 ul_angle
 left_ptr	 umbrella
 left_side	 ur_angle
 left_tee	 watch
 leftbutton	 xterm
 ll_angle	 X_cursor
 lr_angle	

5.9. 图像

有三种一般方法用于显示图形图像 *Tkinter* 应用程序中。

- 要在格式中显示位图（双色）图像，请参阅[节 5.9.1， “BitmapImage 类”](#)。 . xbm
- 若要显示全彩色图像中的、或设置格式，请参阅[节 5.9.2， “如果董事类”](#)。 . gif. pgm. ppm
- Python 成像图书馆（PIL）支持图像中更多的格式。它的类是专门设计用于显示图像 *Tkinter* 应用程序内。见太平船务文件的作者的同伴文件：[Python 成像图书馆 \(PIL\) 快速参考](#)。 ImageTk

5.9.1. 类 BitmapImage

若要显示两个彩色图像格式，您将需要此构造函数：. xbm

```
tk.BitmapImage(file=f[, background=b][, foreground=c])
```

在哪里是图像文件的名称。 *f*. xbm

通常情况下，前景（1）位在图像中的将显示为黑色的像素，并将是透明的图像中的背景（0）位。若要更改此行为，请使用可选的选项来设置背景颜色和可选的选项，将前景色设置为颜色。颜色规格，请参阅[5.3 节、“颜色”](#)。

background=*bb*foreground=*cc*

此构造函数返回一个值，可以使用任何地方 *Tkinter* 期望的图像。例如，要将图像显示为一个标签，使用一个小部件（见[第 12、 “标签小部件”](#)）和供应对象作为选项的值：LabelBitmapImageimage

```
logo = tk.BitmapImage('logo.xbm', foreground='red')
Label(image=logo).grid()
```

5.9.2. 类 PhotoImage

若要显示彩色图像、或格式，您将需要此构造函数：. gif. pgm. ppm

```
tk.PhotoImage(file=f)
```

在哪里是图像文件的名称。构造函数将返回一个值，可以使用任何地方 *Tkinter* 期望的图像。 *f*

5. 10. 几何字符串

一个几何字符串是描述的大小和位置的顶级窗口在桌面上的标准方法。

几何字符串具有这一般形式：

$'wxh\pm x\pm y'$

地点：

- 和 wh 部分给窗口宽度和高度，以像素为单位。他们是由字符分隔。 $wh'x'$
- 如果下一个部分有窗体，它指定窗口的左侧应从桌面左侧的像素。如果它具有形式，在窗口的右侧是从桌面右侧的像素。 $+xx-xx$
- 如果下一个部分有窗体，它指定窗口的顶部应低于桌面顶部的像素。如果它具有形式，在窗口的底部将以上桌面的底部边缘的像素。 $+yy-yy$

例如，创建一个窗口会 120 像素宽 50 像素高，和其右上角会沿桌面和下方的上边缘 20 像素的右边缘。 $geometry='120x50-0+20'$

5. 11. 窗口名称

词 *窗口* 在桌面上描述一个矩形区域。

- A *顶级*或*根*窗口是一个独立存在的根据，窗口管理器的窗口。它与窗口管理器的装饰，装饰和能够移动和调整大小独立。您的应用程序可以使用任意数量的顶级窗口。
- “窗口”一词也适用于任何小部件是一个顶级窗口的一部分。

Tkinter 名称使用分层 *窗口路径名称* 的所有这些窗口。

- 根窗口的名称是。’.’
- 子窗口具有名称的形式，其中是一些整数的字符串形式。例如，命名窗口是一个孩子的根窗口（）。’.*n*’*n*’.135932060’’.’
- 内子窗口的子窗口具有名称的形式在哪里是父窗口的名称，是一些整数。例如，命名窗口有父窗口，所以它是根窗口的一个孙子。
’*p*.*n*’*pn*’.135932060.137304468’’.135932060’
- 窗口的*相对名称*是过去中的路径名称的最后一部分。继续前面的示例中，孙子窗口有一个相对的名称。’.’’137304468’

要获得一个小部件的路径名称，请使用。`wstr(w)`

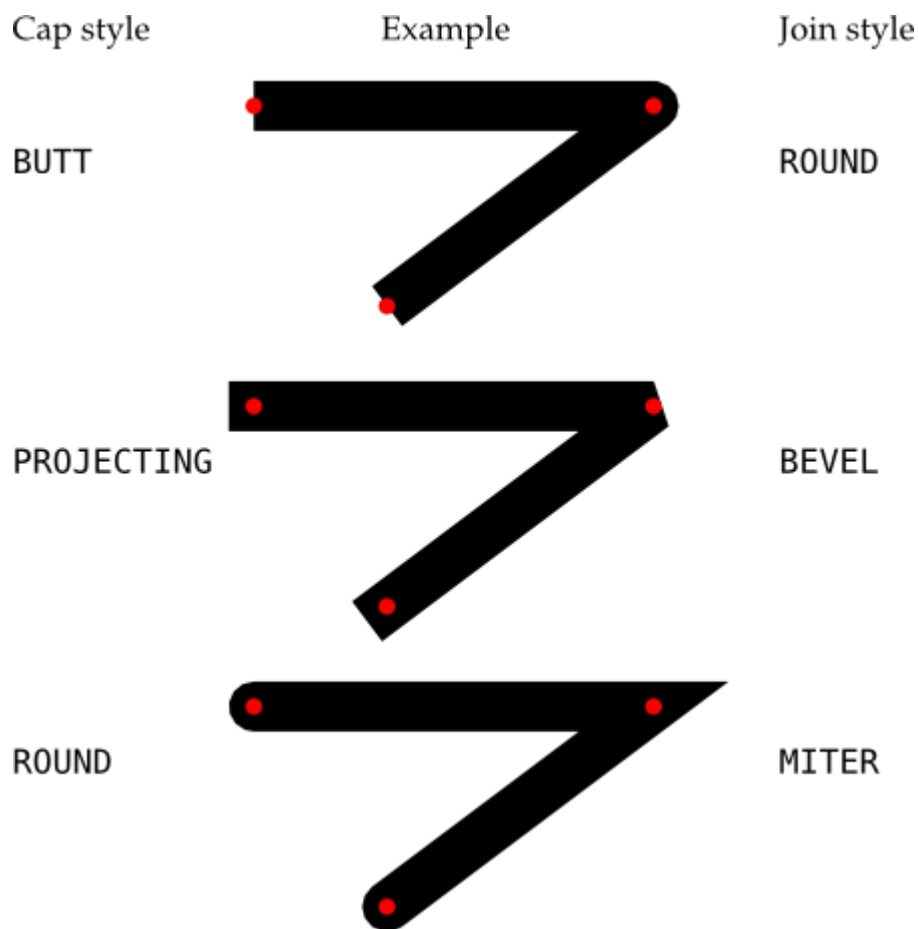
又见[节 26](#)、“[通用构件方法](#)”方法你可以使用来操作窗口名称，特别是[.wininfo name](#)，[.wininfo parent](#)和[.wininfo pathname](#)方法。

5. 12. 帽和加入样式

愉快和有效的渲染图的有时它是一个好的主意，要注意帽和联接类型。

- 线帽样式是结尾的形状的行。样式是：
 - tk.BUTT：切行末尾广场在通过终点线。
 - tk.PROJECTING：行末尾被切断了广场，但过去的终结点的切割的线项目距离等于半线的宽度。
 - tk.ROUND：最后描述了一个半圆形的终结点为中心。
- 联接样式描述两条线段相交角的形状。
 - tk.ROUND：该连接是以相邻的线段相交的点为中心的圆。
 - tk.BEVEL：平面角中间绘制之间相邻行的角度。
 - tk.MITER：相邻的线段边缘继续满足在尖角。

此图显示 *Tkinter* 帽和联接选项如何用一条线由两个连接的线段。小的红色圆圈显示定义此线的点的位置。



5. 13. 虚线样式

大量的小部件允许您指定一个虚线的轮廓。和选项给你短划线的确切模式精细控制。dashdashoffset

dash

此选项指定为整数的元组。第一个整数指定应绘制多少像素。第二个整数指定绘制再一次，在开始之前应跳过多少个像素。当在元组中的所有整数都已都用尽时，被重用相同的顺序中，直至边界是完整。

例如，产生交替 3 像素破折号隔开 5 像素的空白。产生一个值点划线图案，以破折号七次只要点或周围点的差距。值为产生交变的五个像素虚线和五个像素间距。dash=(3, 5) dash=(7, 1, 1, 1) dash=(5,)

dashoff

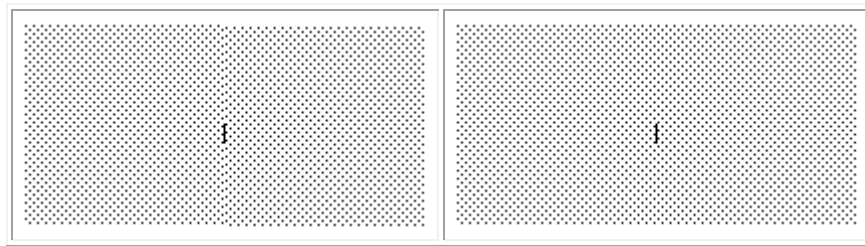
要在不同循环的一个点而不是在开始启动虚线图案，使用选项的哪里的像素数在模式的开头跳过。dashoff=*nn*

示例中的，选项，第一种模式产生将： 上的 2、 关闭 1, 2 和关闭 1。随后的模式将上的 5、 关闭 1, 2 和关闭 1。这里是与此选项组合画一条线的一个屏幕快照：dash=(5, 1, 2, 1) dashoff=3

5.14. 点画模式匹配

这可能看起来像非常挑剔风格点，但如果您绘制图形，有两个对象有斑点图案，真正的专业人员将确保模式沿其边界对齐。

这里是一个例子。左边的屏幕快照显示两个相邻 100×100 方块与斑点状模式，但右侧广场垂直偏移一个像素。短黑图的中心线沿边界的这两个数字。
gray12



第二个屏幕截图是相同的只是两个 100×100 平方他们排队的点画模式。

在实践中，出现这种情况在两种情况。由命名选项控制斑点大面积的对齐方式。对于数字与斑点的轮廓，选项控制其对齐方式。这两个选项具有值的这些形式之一：`offset``outlineoffset`

- `'x, y'`：抵消由这点画模式和价值相对于顶级窗口或到画布的原点。`xy`
- `'#x, y'`：对于在画布上的对象，使用偏移量和相对于顶级窗口。`xy`
- `tk.NE`：对齐点画模式与相应的角落包含对象的一个角落。例如，意味着左上的角的点画图案吻合斑点状的区域的上左角。
`tk.SE``tk.SW``tk.NW``tk.NE`
- `tk.N`：对齐与一侧的包含对象的中心点画模式。例如，意味着中心的点画图案将配合的右侧斑点状的区域中心。`tk.E``tk.Stk``tk.W``tk.E`
- `tk.CENTER`：与包含对象的中心点画图案的居中对齐。

6. 异常处理

是由大多数编程错误引发的异常。tk.TclError

7. 构件 Button

在顶级窗口或命名的框架中创建一个按钮：*parent*

```
w = tk.Button(parent, option=value, ...)
```

构造函数将返回新的小部件。其选项包括：Button

表 5. 按钮小部件选项

activebackground	背景 颜色 按钮时光标下。
activeforeground	前景 颜色 按钮时光标下。
anchor	在按钮上的文本被放置位置。请参见 5.5 条 、“ 锚 ”。例如，会在右上角的按钮放置的文本。 anchor=tk.NE
bd 或 borderwidth	按钮；外周围边框的宽度请参阅 第 5.1 条 ，“ 维度 ”。默认值是两个像素。
bg 或 background	正常的背景 颜色 。
bitmap	其中一个标准 位图 （而不是文本） 在按钮上显示的名称。
command	函数或在单击该按钮时要调用的方法。
cursor	选择要当鼠标位于按钮上方时显示的 光标 。
default	tk.NORMAL 是默认值；如果该按钮是最初被禁用（显示为灰色，响应鼠标单击）的使用。tk.DISABLED
disabledforeground	当按钮处于禁用状态时，使用前景 颜色 。
fg 或 foreground	正常的前景（文本） 颜色 。
font	文本 字体 ，要用于按钮的标签。
height	在文本行（用于文本按钮）或像素（用于图像）按钮的高度。
highlightbackground	重点突出显示小部件不具有 焦点 时的 颜色 。
highlightcolor	重点突出显示小部件具有 焦点 时的 颜色 。

highlightthickness	厚度的 关注 热点。
image	图像 （而不是文本） 按钮上显示。
justify	如何显示多个文本行： 左对齐每行；中心或右对齐。tk.LEFTtk.CENTERtk.RIGHT
overrelief	救济样式用于同时鼠标位于按钮;违约的救济措施是。请参见 第 5.6 节、“救济样式” 。tk.RAISED
padx	额外的左填充和文本的权利。填充的可能值，请参阅 第 5.1 条，“维度” 。
pady	额外的填充文本上方和下方。
relief	指定按钮的救济类型 （见 第 5.6 节，“救济样式” ）。违约的救济措施是。tk.RAISED
repeatdelay	请参阅下面。repeatinterval
repeatinterval	通常情况下，一个按钮触发只一旦当用户释放鼠标按钮。如果您希望该按钮消防在固定的时间间隔，只要按住鼠标按钮，将此选项设置为要之间重复使用和设置的毫秒数到要开始重复之前等待的毫秒数。例如，如果您指定“”按钮将在半秒和每个十分之一秒之后后触发，直到用户释放鼠标按钮。如果用户不按住鼠标按钮至少毫秒，该按钮会着火。 repeatdelayrepeatdelay=500, repeatinterval=100repeatdelay
state	设置此选项可为该按钮显示为灰色，并使其反应迟钝。当鼠标位于它具有价值。默认值为。 tk.DISABLEDtk.ACTIVEtk.NORMAL
takefocus	通常情况下，键盘焦点并访问按钮（见 节 53“焦点：路由键盘输入” ），和一个空格字符作为相同的鼠标单击，“推”按钮。你可以将选项设置为零，以防止将焦点前往该按钮。takefocus
text	按钮上显示的文本。使用内部换行符显示多个文本行。
textvariable	实例是与此按钮上的文本。如果更改了该变量，将按钮上显示的新值。见 节 52“控制变量：小部件背后值” 。StringVar()
underline	默认值为，意味着没有字符按钮上的文本将带有下列线。如果非负的相应的文本字符将带有下列线。例

	如，将下划线的按钮文本的第二个字符。- lunderline=1
width	中信 （如果显示文本） 或像素 （如果显示图像） 按钮的宽度。
wraplength	如果此值设置为一个正数，将包裹文本行以适合此长度范围内。可能的值，请参阅 第 5.1 条，“维度” 。

对象的方法：Button

.flash()

使该按钮之间积极和正常颜色闪烁几次。它原本是在州叶按钮。如果禁用该按钮，忽略。

.invoke()

调用回调的按钮，并返回该函数返回。如果该按钮被禁用或没有回调，任何作用。command

8. 构件 Canvas

画布是矩形的区域，用于绘制图片或其他复杂的布局。它可以放置图形、文本、窗口小部件或帧。请参阅以下部分在画布创建对象的方法：

- `create_arc()`：从椭圆一片。请参阅[节 8.7、“画布弧对象”](#)。
- `create_bitmap()`：为位图图像。请参阅[一节 8.8，“画布位图对象”](#)。
- `create_image()`：图形的图像。请参阅[一节 8.9、“画布图像对象”](#)。
- `create_line()`：一个或多个直线段。请参阅[节 8.10、“画布线对象”](#)。
- `create_oval()`：一个椭圆；使用此也用于绘制圆的椭圆的一个特殊情况。请参阅[节 8.11、“画布 oval 对象”](#)。
- `create_polygon()`：一个多边形。请参阅[8.12 节、“画布多边形对象”](#)。
- `create_rectangle()`：一个矩形。请参阅[一节 8.13、“画布的矩形对象”](#)。
- `create_text()`：文本批注。请参阅[一节 8.14、“画布文本对象”](#)。
- `create_window()`：一个矩形的窗口。请参阅[一节 8.15、“画布窗口对象”](#)。

若要创建一个对象：Canvas

```
w = tk.Canvas(parent, option=value, ...)
```

构造函数将返回新的小部件。支持的选项包括：Canvas

表 6。帆布小部件选项

bd 或 borderwidth	画布；外周围边框的宽度请参阅 第 5.1 条，“维度” 。默认值是两个像素。
bg 或 background	画布上的背景 颜色 。默认值为浅灰色，有关。 '#E4E4E4'
closeenough	，它指定如何关闭鼠标必须到里面考虑的项目。默认值为 1.0。float
confine	如果 true（默认），画布不能滚动外（见下文）。scrollregion
cursor	在画布上使用的光标。请参阅 一节 5.8，“游标” 。
height	在 Y 维度画布的大小。请参阅 第 5.1 条，“维度” 。

highlightbackground	重点突出显示小部件不具有焦点时的颜色。见 节 53“焦点：路由键盘输入” 。
highlightcolor	在 重点 突出显示的 颜色 。
highlightthickness	厚度的 关注 热点。默认值是 1。
relief	救济风格的画布。默认值为。请参见 第 5.6 节、“救济样式” 。tk.FLAT
scrollregion	多大的面积定义画布的元组可以滚动，哪里的左侧、顶部、 右侧和底部。 (w, n, e, s) <i>wnes</i>
selectbackground	使用显示选定的项的背景 颜色 。
selectborderwidth	使用所选项目周围的边框宽度。
selectforeground	使用显示选定的项目的前景 颜色 。
takefocus	通常情况下，集中（见 节 53“焦点：路由键盘输入” ）将循环通过这个部件与 tab 键仅当有键盘绑定设置为它 （请参阅 一节 54、“事件” 的键盘绑定概述）。如果您将此选项设置为 1，焦点总是会参观这个小部件。设置这方面得到的默认行为。’’
width	X 维画布的大小。请参阅 第 5.1 条，“维度” 。
xscrollincrement	通常情况下，可以到任何位置水平滚动画布。你可以得到这种行为通过将设置为零。如果将此选项设置到一些积极的 维度 ，画布可以定位只有在倍数的距离，和的值将用于滚动 滚动单位 ，例如当用户单击滚动条两端的箭头。滚动单位的详细信息，请参阅 节 22、“滚动条部件” 。xscrollincrement
xscrollcommand	画布是可滚动的如果设置此选项的水平滚动条的方法。 <code>.set()</code>
yscrollincrement	工作方式类似，但支配垂直运动。xscrollincrement
yscrollcommand	如果画布是可滚动的此选项应垂直滚动条的方法。 <code>.set()</code>

8. 1. 坐标 Canvas

因为画布可能大于窗口，并配备滚动条在窗口中移动整个画布，有两种坐标系统为每一张画布：

- *窗口*的坐标的点是显示区域的左上角相对于画布上的显示位置。
- *帆布*坐标的点是画布的相对于总的左上角。

8. 2. 显示列表 Canvas

显示列表从背景（“底部”的显示列表）到前景（“顶部”）是指在画布上的所有对象的序列。

如果两个对象重叠时，一个以上其他在显示列表中意味着一个更接近到前台，这将出现在区域中的重叠和掩盖一下面。默认情况下，创建新对象总是在显示列表的顶部（和因此在所有其他对象之前），但你可以重新排序显示列表。

8.3. 对象 IdCanvas

对象 *ID* 的画布上是对象的由该对象的构造函数返回的值。所有对象 ID 值都是简单的整数，并在该画布内唯一对象的对象 ID。

8. 4. 标记 Canvas

标记是一个字符串，你可以将与画布上的对象相关联。

- 标记可以与任意数量的对象在画布上，包括零相关联。
- 一个对象可以有任意数量的标签相关联，包括零。

标签有许多用途。例如，如果你在画布上，画一张地图，河流上的标签的文本对象，您可以将标记附加到所有那些文本对象。这将允许您在该标记，如更改其颜色或删除他们所有对象上执行操作。'riverLabel'

8.5. 参数 Canvas *tagOrId*

参数指定一个或多个对象在画布上。`tagOrId`

- 如果参数是一个整数，它被视为一个对象 ID 和它仅适用于具有该 ID 的唯一对象请参阅[第 8.3 节“画布对象 Id”](#)。`tagOrId`
- 如果这些参数是一个字符串，它被解释为一个标签，并选择有该标记（如果有的话）的所有对象。请参阅[第 8.4 节、“画布标签”](#)。

8. 6. 小部件上的方法 Canvas

所有对象都支持这些方法：Canvas

`. addtag_above(newTag, tagOrId)`

只是上面指定的 [tagOrId](#) 在显示列表中的对象上附加一个新[标记](#)。参数是您想要将附加，作为一个字符串的标记。 *newTag*

`. addtag_all(newTag)`

将给定的[标记](#)附加到画布上的所有对象。 *newTag*

`. addtag_below(newTag, tagOrId)`

向下方指定由 [tagOrId](#) 在显示列表中的一个对象附加一个新的[标记](#)。参数是一个标记字符串。 *newTag*

`. addtag_closest(newTag, x, y, halo=None, start=None)`

将[标记](#)添加到屏幕坐标 (*x*, *y*) 最接近的对象。如果在相同的距离有两个或多个对象，被选择一个更高的[显示列表](#)中。

使用参数来增加有效的点大小。例如，值为 5 将视为内 5 个像素的 (*x*, *y*) 的任何对象重叠。 *halo*

如果在参数中传递的[对象 ID](#)，此方法标记下面是显示列表中的最高限定对象。 *start*

`. addtag_enclosed(newTag, x1, y1, x2, y2)`

将[标记](#)添加到完全发生在矩形的左上角是 (*x1*, *y1*)，其右下角是 (*x2*, *y2*) 内的所有对象。 *newTag*

`. addtag_overlapping(newTag, x1, y1, x2, y2)`

像以前的方法，但影响的所有对象，与给定矩形分享至少一个点。

`. addtag_withtag(newTag, tagOrId)`

将[标签](#)添加到由 [tagOrId](#) 指定的对象。 *newTag*

`. bbox(tagOrId=None)`

返回一个元组 (x_1, y_1, x_2, y_2) 描述包围矩形的由 [tagOrId](#) 指定的所有对象。如果省略该参数，则返回一个矩形封闭在画布上的所有对象。矩形的左上角是 (x_1, y_1) 和右下角是 (x_2, y_2) 。

.canvasx(screenx, gridspacing=None)

翻译一个窗口 x 坐标到画布坐标。如果提供画布坐标是舍入到最接近此值的倍数。 *screenxgridspacing*

.canvay(screeny, gridspacing=None)

转换到画布坐标窗口 y 坐标。如果提供画布坐标是舍入到最接近此值的倍数。 *screenygridspacing*

.coords(tagOrId, x0, y0, x1, y1, ..., xn, yn)

如果您传递只有 [tagOrId](#) 参数，返回一个元组的最低的坐标或只该参数所指定的对象。这些坐标的数目取决于对象的类型。在大多数情况下它将是一个 4 元组 (x_1, y_1, x_2, y_2) 描述对象的定界框。

你可以通过传入新坐标移动对象。

.dchars(tagOrId, first=0, last=first)

从文本项目或项目中删除字符。字符之间和 *包容性* 被删除，在那里这些值可以是整数索引或字符串来表示文本的末尾。例如，对于一个画布和项目，将删除第二个字符。 *firstlast'end' CIC.dchars(I, 1, 1)*

.delete(tagOrId)

删除选定的 [tagOrId](#) 的对象。如果没有项目匹配，它不是错误。
tagOrId

.dtag(tagOrId, tagToDelete)

移除指定的从对象或对象由 [tagOrId](#) 指定的 [标记](#)。 *tagToDelete*

.find_above(tagOrId)

由 [tagOrId](#) 指定的对象的上方对象返回的 ID 号。如果多个对象匹配，你得到最高的一个。如果你通过它最高的对象的对象 ID，返回一个空的元组。

.find_all()

返回 [对象的 ID 号](#) 为所有对象列表的画布上，从最低到最高。

`.find_below(tagOrId)`

返回正下方由 [tagOrId](#) 指定一个对象的[对象 ID](#)。如果多个对象匹配，你得到最低的一个。如果你通过它的最低的对象的对象 ID，返回一个空的元组。

`.find_closest(x, y, halo=None, start=None)`

返回一个包含[对象 ID](#)的对象最接近点的单身人士元组。如果有不符合条件的对象，返回一个空的元组。(*x*, *y*)

使用参数来增加有效的点大小。例如，将视为内 5 个像素的任何对象重叠。*halo**halo*=5(*x*, *y*)

如果对象 ID 作为参数传递，则此方法返回位于下面的[显示列表中的](#)最高限定对象。*start**start*

`.find_enclosed(x1, y1, x2, y2)`

返回的[对象 Id](#)的发生完全的矩形的左上的角和右下角是内的所有对象的列表。(*x1*, *y1*) (*x2*, *y2*)

`.find_overlapping(x1, y1, x2, y2)`

像前面的方法，但返回的对象 Id 与给定矩形分享至少一个点的所有对象的列表。

`.find_withtag(tagOrId)`

返回一个[对象 Id](#)的对象或对象的列表指定的 [tagOrId](#)。

`.focus(tagOrId≠None)`

将焦点移到由 [tagOrId](#) 指定的对象。如果有多个此类对象，则将焦点移到[显示列表](#)中的第一个允许插入光标。如果没有符合条件的项目，或画布上不具有焦点，焦点不会移动。

如果省略该参数，则返回具有焦点的对象的 ID 或如果他们都不做。
,,

`.gettags(tagOrId)`

如果 [tagOrId](#) 是一个对象 ID，返回与该对象关联的所有标记的列表。如果参数是一个[标记](#)，将返回最低的对象包含该标记的所有标记。

`.icursor(tagOrId, index)`

假设所选的项目允许文本插入和具有焦点时，设置将插入光标移动到，这可能是一个整数索引或字符串。否则没有任何影响。

index' end'

`.index(tagOrId, specifier)`

返回的整数索引由 [tagOrId](#) 指定的文本项中给出（最低的一个，如果指定多个对象）。返回值是对应的位置作为一个整数，与通常的 Python 公约》，其中 0 是第一个字符之前的位置。

specifiertagOrId

参数可以是任何的：*specifier*

- tk.INSERT 返回插入光标的当前位置。
- tk.END 返回后该项目的最后一个字符的位置。
- tk.SEL_FIRST 返回当前选定文本的起始位置。如果文本项当前不包含选定文本 *Tkinter* 将引发一个异常。tk.TclError
- tk.SEL_LAST 当前选定的文本，结束后返回位置或提高如果该项目当前不包含所选内容。tk.TclError
- 形式的字符串""，返回的字符的字符，其中包含画布的坐标。如果这些坐标是上方或左侧的文本项，该方法返回 0;如果坐标是向右的或在项目下方，该方法返回项的结束的索引。@*x*, *y*(*x*, *y*)

`.insert(tagOrId, specifier, text)`

插入到 [tagOrId](#)，在参数所给出的位置由指定的对象给出。

stringspecifier

可能的值：*specifier*

- 任何关键字、`INSERT`、`END`、`SEL_FIRST`、`SEL_LAST` 或。请参阅上述这些代码的解释方法的描述。tk.INSERTtk.ENDtk.SEL_FIRSTtk.SEL_LASTindex
- 所需插入，使用正常的 Python 公约为字符串中的位置的位置。

`.itemcget(tagOrId, option)`

返回给定配置的价值中所选的对象（或最低的对象，如果 [tagOrId](#) 指定多个）。这是类似于 *Tkinter* 对象的方法。 *option.cget()*

`.itemconfigure(tagOrId, option, ...)`

如果未不提供任何参数，返回一个字典的键是由 [tagOrId](#) 指定的对象的选项（最小，如果指定多个对象）。 *optiontagOrId*

若要更改指定项的配置选项，提供一个或多个关键字参数的窗体。
option=value

.move(*tagOrId*, *xAmount*, *yAmount*)

将项目添加到他们的 x 坐标和 y 坐标由 [*tagOrId*](#)指定移动。
xAmountyAmount

.postscript(*option*, ...)

生成画布的当前内容封装的 PostScript 表示的形式。这些选项包括：

colormode	使用彩色输出、 灰度或黑白。'color''gray''mono'
file	如果提供，名称将在其中写入 PostScript 文件。如果不指定此选项，则 PostScript 是作为字符串返回。
height	多少的画布打印的 Y 大小。默认值为画布上的整个可见高度。
rotate	如果为 false，将呈现页面，在纵向方向;如果为 true，在景观。
x	最左边的画布区域坐标来打印。
y	最顶层的画布区域坐标来打印。
width	多少的画布打印的 X 大小。默认值为画布上的可见宽度。

.scale(*tagOrId*, *xOffset*, *yOffset*, *xScale*, *yScale*)

他们从点 P 的距离缩放所有对象 = (，)。规模因素和基于值为 1.0，意味着不缩放。在由 [*tagOrId*](#)所选的对象的每一点移动，使得其 x P 距离乘以和其 y 距离乘以。
xOffsetyOffsetxScaleyScalexScaleyScale

此方法不会改变大小的文本项，但可能会移动它。

.scan_dragto (*x*, *y*, ~~获得~~ 10.0)

请参阅下面的方法。*.scan_mark()*

.scan_mark(*x*, *y*)

此方法用于实现快速滚动画布。意图是用户将按住鼠标按钮，然后移动向上扫描（滚动）画布鼠标水平和垂直的方向取决于如何远鼠标的速度已经因为鼠标按钮被按下。

若要实现此功能，请将鼠标的按钮式事件绑定到处理程序，调用在哪里，鼠标当前坐标。将事件绑定到处理程序，假设鼠标按钮仍然是下跌，要求在哪里，是当前的鼠标坐标。`scan_mark(x, y)<Motion>scan_dragto(x, y, gain)xy`

该参数控制扫描的速率。此参数的默认值为 10.0。为更快的扫描使用较大的数字。*gain*

`.select_adjust(oid, specifier)`

调整当前的选定文本，包括由中具有[对象 ID](#)的文本项的参数给出的位置的界限。*specifieroid*

当前选择定位点也将设置为指定的位置。选择定位点的讨论，请参见下面的画布方法。`select_from`

值，请参阅上面的画布方法。*specifierinsert*

`.select_clear()`

移除当前选定的文本，如果它被设置。如果当前没有选定内容，并没有什么。

`.select_from(oid, specifier)`

此方法将选择定位点设置为参数，提出了其[对象 ID](#)的文本项内所给出的位置。*specifieroid*

在给定的画布上当前选定的文本指定的三个位置：起始位置、结束位置 and 选择定位点，可能在任何地方在这两个职位。

若要更改当前选定的文本的位置，使用此方法结合、和帆布方法 (q.v.)。`select_adjustselect_fromselect_to`

`.select_item()`

如果在这个画布上当前选定的文本，返回包含所选内容的文本项的[对象 ID](#)。如果有是当前没有选定内容，此方法返回。None

`.select_to(oid, specifier`

此方法更改当前选定的文本，它包括选择锚，并给出了由内文本项的 [对象 ID](#) 的位置是由。值，请参阅上面的画布方法。

specifieroidsspecifierinsert

.tag_bind(*tagOrId*, *sequence*=None, *function*=None, *add*=None)

将事件绑定到画布上的对象。由 [tagOrId](#) 所选对象的对象，将与事件关联的处理程序。如果参数是一个字符串开头，新的绑定添加到现有的绑定，否则新的绑定替换为给定。

functionsequenceadd + ' sequencesequence

在事件绑定上的一般信息，请参阅[节 54、“事件”](#)。

请注意，绑定应用于在方法调用时都有此标记的项。如果从这些项目后来删除标记，绑定将坚持这些项目。如果您指定的标记是后来应用到你打电话的时候，并没有该标记的项目绑定将不应用于这些新标记的项。tag_bindtag_bind

.tag_lower(*tagOrId*, *belowThis*)

按标记或 ID 移动对象或对象选定的 [tagOrId](#) 内[显示列表](#)下方的第一或唯一对象体的位置。*belowThis*

如果有多个项目与标记，被保留其相对的堆叠顺序。*tagOrId*

此方法不影响画布窗口项目。若要更改窗口项目的堆叠顺序，使用或窗口上的方法。lowerlift

.tag_raise(*tagOrId*, *aboveThis*)

按标记或 ID 移动对象或对象选定的 [tagOrId](#) 内[显示列表](#)上方的第一或唯一对象体的位置。*aboveThis*

如果有多个项目与标记，被保留其相对的堆叠顺序。*tagOrId*

此方法不影响画布窗口项目。若要更改窗口项目的堆叠顺序，使用或窗口上的方法。lowerlift

.tag_unbind(*tagOrId*, *sequence*, *funcId*=None)

从画布对象或由 [tagOrId](#) 指定的对象中移除处理程序和事件绑定。请参阅[第 54、“事件”](#)。*funcIdsequence*

.type(*tagOrId*)

返回的第一或唯一对象的类型指定由 [tagOrId](#)。返回的值将一个字符串,,,,,,, 或。
'arc''bitmap''image''line''oval''polygon''rectangle''text''window'

.xview(tk.MOVETO, *fraction*)

此方法滚动画布相对于其图像，并用于绑定到一个相关的滚动条的选项。画布是水平滚动到一个位置，给出了在哪里 0.0 将画布移动到其最左端的位置和 1.0 到其最右边的位置。commandoffset

.xview(tk.SCROLL, *n*, *what*)

此方法在将向左或向右移动画布： 参数指定到多少移动，可以是或，并告诉多少个单位要移动画布右侧相对于其图像（或左，如果为负数）。*what*tk.UNITStk.PAGES*n*

通过画布的选项；的值给出了移动的大小请参阅[节 22、“滚动条部件”](#)。tk.UNITStk.scrollincrement

对于由运动被乘以画布的宽度的十分之九。tk.PAGES, *n*

.xview_moveto(*fraction*)

此方法在相同的方式滚动画布。.xview(tk.MOVETO, *fraction*)

.xview_scroll(*n*, *what*)

相同。.xview(tk.SCROLL, *n*, *what*)

.yview(tk.MOVETO, *fraction*)

垂直滚动等效。.xview(tk.MOVETO, ...)

.yview(tk.SCROLL, *n*, *what*)

垂直滚动等效。.xview(tk.SCROLL, ...)

.yview_moveto(*fraction*)

垂直滚动等效。.xview()

.yview_scroll(*n*, *what*)

垂直滚动等价物的、
和。.xview().xview_moveto().xview_scroll()

8. 7. 弧对象 Canvas

弧对象在画布上，在其最一般的形式，是一个楔形的切片，带出一个椭圆。这包括整个椭圆形和圆形作为特例。对将要绘制的椭圆的几何形状的更多信息，请参阅[节 8.11，“画布 oval 对象”](#)。

若要在画布上创建弧对象，请使用：*C*

```
id = C.create_arc(x0, y0, x1, y1, option, ...)
```

该构造函数返回的新的弧对象的[对象 ID](#)在画布上。*C*

点 (,) 是顶部左角和 (,)，适合椭圆的矩形的右下角。如果此矩形是正方形的你得到一个圆。*x0y0x1y1*

各种选项包括：

表 7。画布弧选项

activedash	这些选项应用电弧处于状态时，那就是，当鼠标位于弧。例如，此选项指定的内部颜色弧处于活动状态时。对于选项值，看看，和分别。 tk.ACTIVEactivefillldashfilloutlineoutlinestipplestipplewidth
activefill	
activeoutline	
activeoutlinestipple	
activestipple	
activewidth	
dash	轮廓的虚线样式。请参阅 节 5.13、“虚线样式” 。
dashoffset	短跑的大纲模式偏移量。请参阅 节 5.13、“虚线样式” 。
disableddash	这些选项应用时的电弧。statetk.DISABLED
disabledfill	
disabledoutline	
disabledoutlinestipple	
disabledstipple	
disabledwidth	

disabledwidth	
extent	中度的扇区的宽度。切片的角度给出了由选项开始，逆时针延展为度。 <i>startextent</i>
fill	默认情况下，内部的弧是透明的并将选择这种行为。您还可以设置此选项为任何 颜色 和弧的内政将充满那种颜色。 fill=''
offset	点画弧的内政模式偏移量。请参阅 一节 5.14, “匹配点画模式” 。
outline	片外周围边框的颜色。默认颜色为黑色。
outlineoffset	点画轮廓模式偏移量。请参阅 一节 5.14, “匹配点画模式” 。
outlinestipple	如果使用该选项，此选项指定用于点画边界的位图。默认是黑色的并可以通过设置指定的默认值。 outlineoutlinestipple=''
start	切片，以度为单位来衡量从起始角 + x 方向。如果省略，你得到整个椭圆。
state	此选项是默认情况。它可以将设置为使电弧不可见或去变灰色出弧和使它响应事件。 tk.NORMALtk.HIDDENtk.DISABLED
stipple	指示如何将斑点状弧的内部填充的位图。默认值为（固体）。你可能会想要类似。有没有影响，除非已设置为一些颜色。stipple='' stipple='gray25' <i>fill</i>
style	<p>默认设置是绘制整个弧;使用这种风格。若要绘制只圆弧边上的切片，请使用。若要绘制圆弧和和弦（直线连接弧线的端点），请使用。</p> <p>style=tk.PIESLICEstyle=tk.ARCstyle=tk.CHORD</p> <div style="text-align: center;"> <p>The diagram illustrates two different arc styles. On the left, labeled 'PIESLICE', is a sector of a circle, which is a portion of a circle defined by two radii and an arc. On the right, labeled 'CHORD', is a semi-circle with a straight line chord connecting the two endpoints of the arc.</p> </div>
tags	如果一个字符串，该字符串标记弧。用字符串的元组与多个标签标记的弧。请参阅 第 8.4 节、“画布标签” 。

width	外弧周围边框的宽度。默认值为 1 像素。
-------	----------------------

8. 8. 位图对象 Canvas

在画布上的位图对象表现为两种颜色，（为 0 的数据值） 的背景色和前的景色（1 值）。

若要在画布上创建一个位图对象，请使用：*C*

```
id = C.create_bitmap(x, y, *options ...)
```

它返回那画布上的图像对象的整数 ID 号。

值，指定放置位图的参考点。*xy*

选项包括：

表 8。画布上的位图选项

activebackground	这些选项指定、 当位图是积极的就是当鼠标位于位图 和值。backgroundbitmapforeground
activebitmap	
activeforeground	
anchor	位图是相对于点（，） 定位的。默认值是锚 =，意 思位图在（，） 位置上居中。各种选项值，请参阅 节 5.5、“锚” 。例如，如果您指定位图将定位点 （，） 是位于东北（右上角） 角的位图。 <i>xytk. CENTERxyanchoranchor=tk. NExy</i>
background	颜色 将出现在位图中有 0 值。默认值是，意思透明。 <i>background=''</i>
bitmap	位图显示;请参阅 第 5.7 节，“位图” 。
disabledbackground	这些选项指定背景、 位图和前景时，位图使用。 <i>statetk. DISABLED</i>
disabledbitmap	
disabledforeground	
foreground	颜色 将出现在位图中有 1 的值。默认值为。 <i>foreground=' black'</i>

state	默认情况下，与创建项目。使用，使该项目处于灰显状态和响应事件;使用可以使项目不可见。 state=tk.NORMALtk.DISABLEDtk.HIDDEN
tags	如果一个字符串，该字符串标记该位图。用字符串的元组与多个标签标记的位图。请参阅 第 8.4 节、“画布标签” 。

8.9. 图像对象 Canvas

若要在画布上显示图形的图像，请使用：*C*

```
id = C.create_image(x, y, option, ...)
```

此构造函数返回的画布上的图像对象的整数 ID 号。

参照点 (,) 定位的图像。选项包括：*xy*

表 9. 画布上的图像选项

activeimage	当鼠标位于项目上时显示的图像。选项值，请参阅下文。 image
anchor	默认值是，意义的图像在 (,) 位置上居中。此选项的可能值，请参阅 节 5.5、“锚” 。例如，如果指定，以便点 (,) 位于图像的底边 (南方) 中心，将定位图像。 anchor=tk.CENTERxyanchor=tk.Sxy
disabledimage	当该项目处于非活动状态时显示的图像。选项值，请参阅下文。image
image	要显示的图像。信息，请参阅 一节 5.9、“图像” ，以上，有关如何创建可以加载到画布上的图像。
state	通常情况下，在状态下创建图像对象。将此值设置为，使它变灰和响应鼠标。如果你将其设置为，该项目是无形的。 tk.NORMALtk.DISABLEDtk.HIDDEN
tags	如果一个字符串，该字符串标记图像。用字符串的元组多个标签标记的图像。请参阅 第 8.4 节、“画布标签” 。

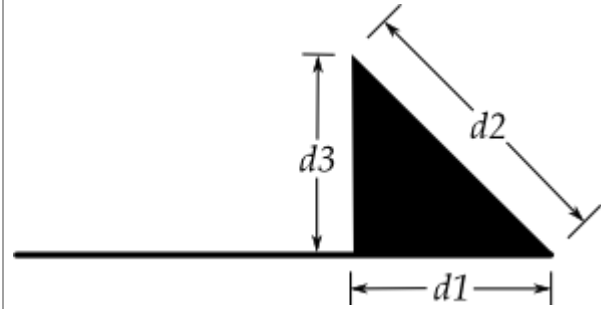
8. 10. 线对象 Canvas

一般情况下，行可以包含任意数量的连接端到端的段，每段可以是直线或曲线。若要在画布上创建一个画布线对象，请使用：*C*

```
id = C.create_line(x0, y0, x1, y1, ..., xn, yn, option, ...)
```

行云透过一系列的点 (,) (,) , (,). 选项包括：*x0y0x1y1xnyn*

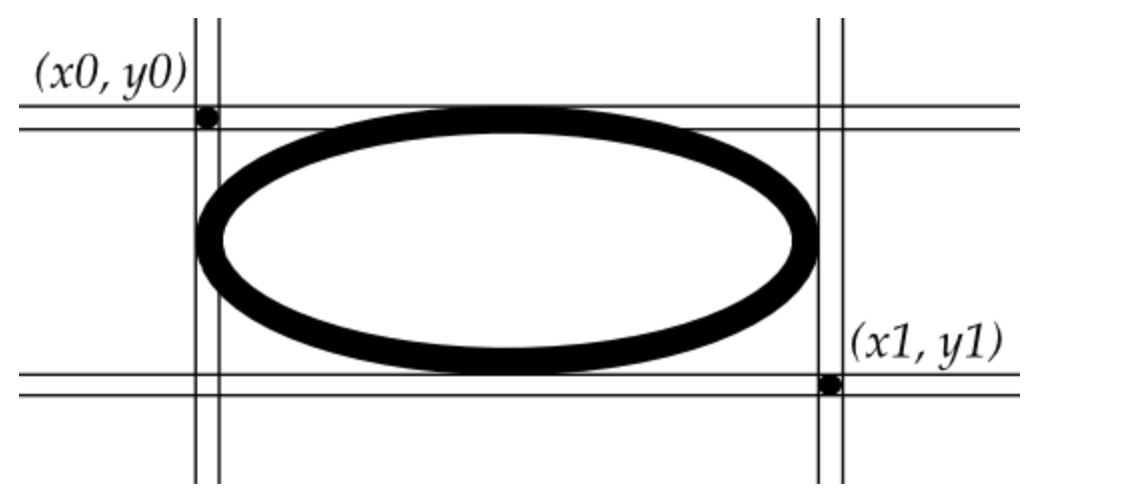
表 10。帆布行选项

activedash	这些选项指定、、和要线处于活动状态，那就是，当鼠标位于它时使用的值。dashfillstipplewidth
activefill	
activestipple	
activewidth	
arrow	默认情况下，行有无箭头。用于获取箭头线 (,) 末尾。使用箭头到达尽头。用于在两端的箭头。 arrow=tk. FIRST <i>x0y0</i> arrow=tk. LASTarrow=tk. BOTH
arrowshape	描述形状的箭头由选项添加一个元组。默认值为。(d1, d2, d3) arrow(8, 10, 3) 
capstyle	您可以使用此选项；指定的线末端的形状请参阅 5. 12 节，“帽和联接样式” 。默认的选项是。tk. BUTT
dash	若要产生一条虚线，请指定此选项;请参阅 节 5. 13、“虚线样式” 。默认外观是一条实线。
dashoffset	如果你指定一种模式，默认设置是启动指定的模式在一行的开头。该选项允许您指定虚线模式的起始位置距为给定

	距离发生后行的起始处。请参阅 节 5.13、“虚线样式” 。 dashdashoffset
disableddash	、 、 、 和在该项目处于状态时要使用的值。 dashfillstipplewidthtk.DISABLED
disabledfill	
disabledstipple	
disabledwidth	
fill	使用的 颜色 在绘制线条。默认值为。fill='black'
joinstyle	对于由多个直线段组成的线条，此选项控制段交界的外观。更多详细信息，请参阅 5.12 节，“帽和联接样式” 。 默认样式是 ROUND
offset	斑点的线路，这个选项的目的是与那些相邻对象的项目点画模式匹配。请参阅 一节 5.14，“匹配点画模式” 。
smooth	如果为 true，线是绘制为一系列的抛物线样条曲线拟合的点集。默认值为 false，它将线呈现为一系列直线段。
splinsteps	如果选项是真实的每个样条被呈现为直线段的数目。选项指定的段数用于近似每一条线;默认值为。 smoothsplinstepssplinsteps=12
state	通常情况下，在状态下创建行项。设置此选项来作线看不见;将其设置到为使鼠标无响应。 tk.NORMALtk.HIDDENtk.DISABLED
stipple	若要绘制斑点的线，此选项设置为指定的点画的图案，如位图。可能的值，请参阅 第 5.7 节，“位图” 。 stipple='gray25'
tags	如果一个字符串，该字符串标记线。用字符串的元组与多个标记标记线。请参阅 第 8.4 节、“画布标签” 。
width	线条的宽度。默认值为 1 像素。可能的值，请参阅 第 5.1 条，“维度” 。

8.11。 oval 对象 Canvas

椭圆形，在数学上，都是椭圆，包括圈作为一种特殊情况。椭圆是融入一个定义按左上角的坐标 (x_0, y_0) 和坐标 (x_1, y_1) 的点只是外面的右下角的矩形。



椭圆形的顶部和左侧行此框，则会配合，但适合只是里面的底部和右侧。

若要在画布上创建一个椭圆，请使用： C

```
id = C.create_oval(x0, y0, x1, y1, option, ...)
```

其中在画布上返回新 oval 对象的[对象 ID](#)。 C

对于椭圆的选项：

表 11。画布椭圆选项

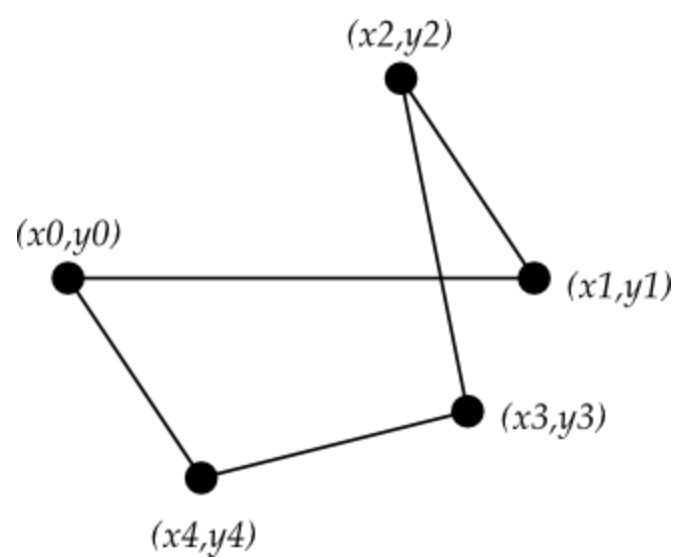
activedash	这些选项指定虚线图案、填充颜色、轮廓颜色、轮廓点画图案，内部点画图案和需要使用椭圆处于状态时，那就是，当鼠标位于椭圆轮廓宽度值。为选项值，请参阅、和。 tk.ACTIVEdashfilloutlineoutlinestipplestipple width
activefill	
activeoutline	
activeoutlinestipple	
activestipple	
activewidth	

dash	产生该椭圆的虚线的边框，请将此选项设置为短划线图案;请参阅 节 5.13、“短跑模式”
dashoffset	使用选项时，选项用于更改与该椭圆的边框虚线图案的对齐方式。请参阅 一节 5.14，“匹配点画模式” 。dashdashoffset
disableddash	这些选项指定椭圆的外观项目时。 statetk.DISABLED
disabledfill	
disabledoutline	
disabledoutlinestipple	
disabledstipple	
disabledwidth	
fill	一个椭圆内部的默认外观是透明的和一个值的会选择这种行为。您还可以设置此选项为任何颜色和椭圆的内部会充满那种颜色;请参阅 5.3 节、“颜色” 。fill=’
offset	点画模式偏移量的内政。请参阅 一节 5.14，“匹配点画模式” 。
outline	椭圆外周围边框的颜色。默认值为。 outline=’ black’
outlineoffset	点画模式偏移量的边界。请参阅 一节 5.14，“匹配点画模式” 。
stipple	指示如何将斑点状椭圆的内部位图。默认值为，这意味着一种纯色。一个典型的值将是。没有任何作用，除非已设置为一些颜色。请参阅 第 5.7 节，“位图” 。stipple=’ stipple=’ gray25’ fill
outlinestipple	要用于边界的点模式。选项值，请参阅下文。 stipple
state	默认情况下，在状态下创建椭圆的项目。将此选项设置为使椭圆响应鼠标操作。将其设置到为使项目不可见。tk.NORMALtk.DISABLEDtk.HIDDEN

tags	如果单个字符串，该字符串标记为椭圆形。用字符串的元组与多个标记标记为椭圆形。请参阅 第 8.4 节、“画布标签” 。
width	椭圆外周围边框的宽度。默认值是 1 个像素;可能的值，请参阅 第 5.1 条，“维度” 。如果你设置为零，将不会显示边框。如果将其设置为零，使填充为透明，可以使整个椭圆消失。

8. 12. 多边形对象 Canvas

一个多边形所显示，有两个部分： 它的轮廓和它的内部。其几何形状指定为一系列的顶点 $[(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)]$ ，但实际周长包括更多的一段从 (x_n, y_n) 回 (x_0, y_0) 。在此示例中，有五个顶点：



在画布上创建一个新的多边形对象：*C*

```
id = C.create_polygon(x0, y0, x1, y1, ..., option, ...)
```

构造函数返回该对象的[对象 ID](#)。选项：

表 12。画布多边形选项

activedash	这些选项指定多边形的外观，当它处于状态，那就是，当鼠标位于它。为选项值，请参阅、和。 tk.ACTIVEDashfilloutlineoutlinestipplestipplewidth
activefill	
activeoutline	
activeoutlinestipple	
activestipple	
activewidth	
dash	使用此选项可以生成多边形的虚线的边框。请参阅 节 5.13、“虚线样式” 。

dashoffset	使用此选项可以在开头之外其周期开始在一些点虚线图案。请参阅 节 5.13、“虚线样式” 。
disableddash	这些选项指定多边形的外观时其是。 statetk.DISABLED
disabledfill	
disabledoutline	
disabledoutlinestipple	
disabledstipple	
disabledwidth	
fill	您可以通过将此选项设置为一种颜色颜色内政。多边形内部的默认外观是透明的，你可以设置为得到这种行为。请参阅 5.3 节、“颜色” 。fill=’
joinstyle	此选项控制外观的相邻的多边形边的交点。请参阅 5.12 节，“帽和联接样式” 。
offset	抵消的多边形内的点模式。请参阅 一节 5.14，“匹配点画模式” 。
outline	颜色的大纲;默认值，这使得轮廓透明。outline=’
outlineoffset	点画为边界的偏移量。请参阅 一节 5.14，“匹配点画模式” 。
outlinestipple	使用此选项可以得到多边形的点刻的边框。该选项的值必须是位图;请参阅 第 5.7 节，“位图” 。
smooth	默认的大纲使用直线连接顶点;使用来获取这种行为。如果你使用，你得到连续样条曲线。此外，如果您设置，您可以任何部分直接通过复制那段每一端的坐标。smooth=0smooth=1smooth=1
splinsteps	如果选项是真实的每个样条被呈现为直线段的数目。选项指定的段数用于近似每一条线;默认值为。smoothsplinsteps=12
state	默认情况下，在状态下创建的多边形。将此选项设置为使多边形不可见，或将其设置为向鼠标无响应。tk.NORMALtk.HIDDENtk.DISABLED
stipple	指示如何将斑点状的多边形内部的位图。默认值为，这意味着一种纯色。一个典型的值将是。没有

	任何作用，除非已设置为一些颜色。请参阅 第 5.7 节，“位图” 。stipple='' stipple='gray25' fill
tags	如果一个字符串，该字符串标记的多边形。用字符串的元组与多个标签标记的多边形。请参阅 第 8.4 节、“画布标签” 。
width	宽度的大纲;默认值为 1。请参阅 第 5.1 条，“维度” 。

8. 13. 矩形对象 Canvas

每个矩形指定为两点：（，） 是左上角，和 （，） 是像素只是外面的右下角的位置。*x0y0x1y1*

例如，由顶部指定的矩形左角（100, 100）和底部右角（102, 102）是由两个像素，包括像素（101, 101）但包括（102, 102）两个正方形像素。

矩形是绘制的两个部分：

- 大纲对其顶部和左侧，但外面的矩形内在于其底部和右侧的矩形。默认外观是一个像素宽的黑色边框。

例如，假设有一个矩形的左上角（10， 10）和右下角（11, 11）。如果您请求没有边界 （） 和绿色填充 （， 你会得到一个绿色像素在（10， 10）。然而，如果您请求与一个黑色边框 （） 相同的选项，你会得到四个黑色像素（10， 10），在（10， 11）、（11, 10），和（11, 11）。width=0fill='green')width=1

- 填充是大纲内的区域。其默认外观是透明的。

在画布上创建一个矩形对象：*C*

```
id = C.create_rectangle(x0, y0, x1, y1, option, ...)
```

此构造函数返回的矩形[对象 ID](#)那画布上。选项包括：

表 13. 画布矩形选项

activedash	这些选项指定的矩形外观当其是，那就是，当鼠标位于该矩形上。选项值，请参阅，及以下。 statetk.ACTIVEDashfilloutlineoutlinestipplestipplewidth
activefill	
activeoutline	
activeoutlinestipple	
activestipple	
activewidth	
dash	若要产生虚线的边框的矩形，使用此选项指定虚线图案。请参阅 节 5.13、“虚线样式” 。

dashoffset	使用此选项可以在周期；开始边界的虚线图案在不同的点请参阅 节 5.13、“虚线样式” 。
disableddash	这些选项指定矩形的外观时其是。statetk.DISABLED
disabledfill	
disabledoutline	
disabledoutlinestipple	
disabledstipple	
disabledwidth	
fill	默认情况下，矩形的内部是空的和你可以得到与此行为。你也可以将选项设置为一种颜色;请参阅 5.3 节、“颜色” 。fill=’
offset	使用此选项可以更改内部点画图案的偏移量。请参阅 一节 5.14，“匹配点画模式” 。
outline	边框的颜色。默认值为。outline=’ black’
outlineoffset	使用此选项来调整偏移量点画图案轮廓;请参阅 一节 5.14，“匹配点画模式” 。
outlinestipple	使用此选项可以产生斑点的大纲。模式指定的位图;请参阅 第 5.7 节，“位图” 。
state	默认情况下，在状态下创建的矩形。鼠标是在矩形时的状态。设置此选项可为矩形显示为灰色，并使其响应鼠标事件。tk.NORMALtk.ACTIVEtk.DISABLED
stipple	指示如何将斑点状的矩形的内部位图。默认值为，这意味着一种纯色。一个典型的值将是。没有任何作用，除非已设置为一些颜色。请参阅 第 5.7 节，“位图” 。stipple=’ stipple=’ gray25’ fill
tags	如果一个字符串，该字符串标记该矩形。使用字符串的元组用多个标记标签的矩形。请参阅 第 8.4 节、“画布标签” 。
width	边框的宽度。默认值为 1 像素。使用使边框不可见。请参阅 第 5.1 条，“维度” 。width=0

8. 14. 文本对象 Canvas

通过创建一个文本对象，可以在画布上显示一行或多行的文本：*C*

```
id = C.create_text(x, y, option, ...)
```

这在画布上返回文本对象的[对象 ID](#)。选项包括：*C*

表 14。画布文本选项

activefill	用于文本处于活动状态，那就是，当鼠标时文本颜色。选项值，请参阅下文。fill
activestipple	要使用文本处于活动状态时的点画模式。选项值，请参阅下文。stipple
anchor	默认值为，也就是说，文本围绕垂直和水平位置（，）。可能的值，请参阅 节 5.5、“锚” 。例如，如果指定，所以其左下的角是在点（，），将放置文本。 anchor=tk.CENTERxyanchor=tk.SWxy
disabledfill	要文本对象时使用的文本颜色。选项值，请参阅下文。statetk.DISABLEDfill
disabledstipple	要禁用文本时使用的点画模式。选项值，请参阅下文。stipple
fill	默认文本颜色是黑色的但你可以将选项设置为那种颜色使它在任何颜色。请参阅 5.3 节、“颜色” 。fill
font	如果你不喜欢默认的字体，设置此选项为任何字体值。请参见 5.4 节，“字体” 。
justify	对于多行文本显示，此选项控制如何行有正当理由：（默认值），或。tk.LEFTtk.CENTERtk.RIGHT
offset	偏移量，用于呈现文本的点画。更多的信息，请参阅 一节 5.14，“匹配点画模式” 。
state	默认情况下，文本项的状态。设置此选项来作在响应鼠标事件，或将其设置到为使它不可见。 tk.NORMALtk.DISABLEDtk.HIDDEN

stipple	指示如何将斑点状的文本的位图。这意味着默认是与固体。一个典型的值将是。请参阅 第 5.7 节，“位图” 。 stipple='' stipple='gray25'
tags	如果一个字符串，该字符串标记的文本对象。用字符串的元组多个标签标记的对象。请参阅 第 8.4 节、“画布标签” 。
text	要在对象中，以字符串形式显示的文本。使用换行符字符（\n）来强制换行。'\n'
width	如果你不指定选项，文本将设置只要是长线的矩形内。然而，你也可以将选项设置为维度和每行文本将分成较短的线路，如果有必要，或甚至破字，以适合指定的宽度以内。请参阅 第 5.1 条，“维度” 。widthwidth

您可以更改文本项中显示的文本。

- 若要检索的文本从一个项目使用[对象 ID](#)在画布上，调用。
`ICC.itemcget(I, 'text')`
- 若要在一个字符串中的文本与画布上，向项目中的文本替换[对象 ID](#)，
请打电话。`ICSC.itemconfigure(I, text=S)`

大量的画布方法允许您操作的文本项。看到[部分 8.6，“上画布窗口小部件的方法”](#)，尤其是、`dcharsfocusicursorindexinsert`

8. 15。 窗口对象 Canvas

您可以通过使用 *画布窗口* 对象放到画布上的任何 *Tkinter* 小部件。一个窗口是可以容纳一个 *Tkinter* 构件的矩形区域。小部件必须作为画布上，相同的顶级窗口的孩子或孩子的一些小部件位于相同的顶级窗口。

如果你想要把复杂多 widget 对象放在画布上，可以使用此方法在画布上，放置一个小部件，然后放置在该框架内的其他窗口小部件。Frame

在画布上创建一个新的画布窗口对象：*C*

```
id = C.create_window(x, y, option, ...)
```

这返回窗口对象的[对象 ID](#)。选项包括：

表 15。画布窗口选项

anchor	默认值为，意思窗口在（，）位置上居中。可能的值，请参阅 节 5.5、“锚” 。例如，如果指定，以便点（，）是在其右（东）边缘的中点，将定位窗口。anchor=tk.CENTERxyanchor=tk.Exy
height	预留的窗口区域的高度。如果省略，则窗口将调整大小以适应包含小部件的高度。可能的值，请参阅 第 5.1 条，“维度” 。
state	默认情况下，窗口项目处于的状态。将此选项设置为使窗口不响应鼠标输入，或使它不可见。tk.NORMALtk.DISABLEDtk.HIDDEN
tags	如果一个字符串，该字符串标记窗口。使用元组的字符串标记窗口与多个标记。请参阅 第 8.4 节、“画布标签” 。
width	预留的窗口区域的宽度。如果省略，则窗口将调整大小以适应所包含的构件的宽度。
window	使用在哪里是您想要放置到画布上的小部件。如果最初省略此属性，您稍后可以调用放置到画布上，小部件在哪里是窗口的对象 id。。window=wwC.itemconfigure (<i>id</i> , window= <i>w</i>)wid

9. 小部件 Checkbutton



Checkbutton 小部件（有时称为“复选框”）的目的是允许用户读取和双向选择。上面的图形显示在关闭的（0）和（1）状态的一种实现 checkbuttons 的外观：这是两个 checkbuttons 使用 24 点时间屏幕截图字体。

该指标是 checkbutton，显示其状态的一部分和标签是在它旁边显示的文本。

- 您将需要创建一个控制变量，类的实例，所以您的程序可以查询和设置的 checkbutton 状态。见[节 52“控制变量：小部件背后值”](#)，下面。
IntVar
- 您还可以使用事件绑定来响应用户操作的 checkbutton；见[第 54、“事件”](#)，下文。
- 您可以禁用 checkbutton。这会更改其外观的“灰色”和使鼠标无响应。
- 你可以摆脱 checkbutton 指标，使整个部件看起来凹进，当它设置，并且看上去凸起，当它清除“推送推送”按钮。

要在现有的父窗口或框架中创建的 `checkbutton:parent`

```
w = tk.Checkbutton(parent, option, ...)
```

构造函数将返回一个新的小部件。选项包括：Checkbutton

表 16。Checkbutton 小部件选项

activebackground	Checkbutton 时光标下的背景色。请参阅 5.3 节、“颜色” 。
activeforeground	Checkbutton 时光标下的前景颜色。
anchor	如果该构件栖息于比它所需要的更大的空间，此选项指定在那个空间 checkbutton 的位置。默认值为。允许的值，请参阅 节 5.5、“锚” 。例如，如果您使用，小部件将放置在左上角的空间。 anchor=tk.CENTERanchor=NW
bg 或 background	后面的标签和指示器显示的正常背景颜色。请参阅 5.3 节、“颜色” 。选项，此选项指定显示为 0 位位图中的颜色。bitmap

bitmap	若要在按钮上显示单色图像，请将此选项设置为位图；请参阅 第 5.7 节，“位图” 。
bd 或 borderwidth	指示器周围的边框的大小。默认值是两个像素。可能的值，请参阅 第 5.1 条，“维度” 。
command	每次用户更改状态的这 checkbutton 调用程序。
compound	使用此选项可以在按钮上显示文本和图形，可一个位图或图像。允许的值描述的位置相对于文本，图形和可以是任何的、、、或。例如，将定位到文本的左侧图形。 tk.BOTTOMtk.TOPtk.LEFTtk.RIGHTtk.CENTERcompound=tk.LEFT
cursor	如果将此选项设置为光标名称（见 节 5.8，“光标” ），鼠标光标将变为这种模式，当它结束的 checkbutton。
disabledforeground	前景 颜色 用于呈现文本的残疾的 checkbutton。默认值是一个斑点的版本的默认前景色。
font	所用的字体。请参见 5.4 节，“字体” 。 <i>text</i>
fg 或 foreground	用于呈现的 颜色 。选项，此选项指定显示为 1 位的位图中的颜色。 <i>textbitmap</i>
height	Checkbutton 上的文本行数。默认值为 1。
highlightbackground	重点突出显示 checkbutton 不具有焦点时的 颜色 。见 节 53“焦点：路由键盘输入” 。
highlightcolor	重点突出显示 checkbutton 获得焦点时的 颜色 。
highlightthickness	厚度的关注热点。默认值为 1。设置为 0 来禁止显示的关注热点。
image	要在按钮上显示一幅图像，对图像对象设置此选项。请参阅 一节 5.9、“图像” 。
indicatoron	通常 checkbutton 作为其指标会显示一个框，显示是否 checkbutton 设置或不。你可以通过设置此行为。然而，如果你设置指示器即会消失，和整个构件成为推按钮时它被清除，并且沉没设置时，它的外观凸起。你可能想要增加值以使它更易于看到这样的控件的状态。 indicatoron=1indicatoron=0borderwidth

justify	如果包含多个行，此选项控制文本的对齐方式:、 或。 <code>texttk.CENTERtk.LEFTtk.RIGHT</code>
offrelief	默认情况下， <code>checkbuttons</code> 使用救济风格时，按钮为 禁用（清除）;使用此选项来指定不同的救济样式，当 按钮处于关闭状态时显示。请参见 第 5.6 节、“救济样 式” 值。 <code>tk.RAISED</code>
offvalue	通常情况下， <code>checkboxbutton</code> 关联的控件变量将设置为 0，当它被清除（关闭）。你可以提供另一个值，通过 设置该值为断开状态。 <code>offvalue</code>
onvalue	通常情况下， <code>checkboxbutton</code> 关联的控件变量将设置为 1，当它被设置（上）。您可以提供通过设置该值为状 态为另一个值。 <code>onvalue</code>
overrelief	使用此选项可以指定要显示鼠标何时位于 <code>checkboxbutton</code> ; 救济样式请参见 第 5.6 节、“救济样式 ” 。
padx	离开左边和右边的 <code>checkboxbutton</code> 和文本多少空间。默 认值为 1 像素。可能的值，请参阅 第 5.1 条，“维度 ” 。
pady	离开的上方和下方的 <code>checkboxbutton</code> 和文本多少空间。 默认值为 1 像素。
relief	使用默认值， <code>checkboxbutton</code> 并不站出来从其背景。你可 以到任何其他样式（请参阅 第 5.6 节、“救济样式” ） 或使用，给你一个固体的黑色框架，它周围设置此选 项。 <code>relief=tk.FLATrelief=tk.SOLID</code>
selectcolor	<code>Checkboxbutton</code> 设置时，它的 颜色 。默认值为。 <code>selectcolor='red'</code>
selectimage	如果你将此选项设置为图像，该图像将出现在 <code>checkboxbutton</code> ，当它被设置。请参阅 一节 5.9、“图像 ” 。
state	默认值是，但您可以使用灰色的控制，使它反应迟钝。 如果光标位于当前上 <code>checkboxbutton</code> ，状态。 <code>state=tk.NORMALstate=tk.DISABLEDtk.ACTIVE</code>
takefocus	默认值是输入的焦点（见 节 53“焦点：路由键盘输入 ” ）将通过 <code>checkboxbutton</code> 。如果你设置焦点不会经过 它。 <code>takefocus=0</code>

text	Checkbox 旁边显示的标签。使用换行符（\n）来显示多行文本。
textvariable	如果你需要在执行期间更改 checkbox 上的标签，创建 StringVar（见 节 52“控制变量：小部件背后值” ）管理的当前值，并将此选项设置为该控制变量。控制变量的值更改时，checkbox 的注释也将会自动更改。
underline	默认值为-1，没有文本标签的字符下划线标出。设置此选项的索引（从零开始计数）的文本中的字符的下划线字符。
variable	控制变量的跟踪的当前状态的 checkbox;见 节 52“控制变量：小部件背后值” 。通常此变量是，手段清除和手段设置，但看到和上面的选项。 IntVar0loffvalueonvalue
width	Checkbox 的默认宽度是由显示的图像或文本的大小确定的。你可以将此选项设置为字符数目和 checkbox 将总是有很多字符的余地。
wrlength	通常情况下，不会换行。你可以将此选项设置为字符数目和所有行都将可以都分解成部分不再比这一数字。

对 checkboxes 的方法包括：

.deselect()

清除（关闭）checkbox。

.flash()

闪烁几次之间其活动和正常的颜色，checkbox，而是让它开始的方式。

.invoke()

你可以调用此方法来获取相同的操作，如果用户点击 checkbox 以更改其状态将会发生。

.select()

集（打开）checkbox。

`.toggle()`

如果清除 `checkboxbutton` 设置，如果清除设置它。

10. 小部件 Entry

小部件的目的是让用户查看和修改单行文本。Entry

- 如果你想要显示多行文本，可以进行编辑，请参阅[节 24、“文本小部件”](#)。
- 如果你想要显示一行或多行的文本，能由用户修改，请参阅[一节 12、“标签小部件”](#)。

一些定义：

- *选择*是突出显示的文本区域中的小部件，如果有的话。Entry

通常由用户用鼠标选定，选定的文本复制到系统剪贴板。然而，**Tkinter** 允许您控制选定的文本获取复制到剪贴板。您还可以选择中的文本在程序控制下。Entry

- *插入光标*显示插入新文本的位置。它被显示仅当用户单击鼠标在某个地方在小部件中。它通常显示为闪烁的竖直线内小部件。您可以自定义其外观在几个方面。
- 作为*指数*给出了在小部件的显示的文本内的位置。有几种方法来指定索引：
 - 作为正常的 Python 指标，从 0 开始。
 - 常数后的现有文本指的位置。tk.END
 - 常数是指插入光标的当前位置。tk.INSERT
 - 常数是指所选内容的第一个字符如果有一个选择。tk.ANCHOR
 - 你可能需要找出哪个字符在小部件中的位置对应于一个给定的鼠标位置。为了简化这一过程，可以使用索引作为字符串的形式，以像素为单位小部件的左边的缘和鼠标之间的水平距离在哪里。这种指数将指定该鼠标水平位置处的字符。'*@n'* nEntry

在根窗口或命名的框架中创建一个新的小部件：Entry*parent*

```
w = tk.Entry(parent, option, ...)
```

此构造函数返回新的小部件。选项包括：Entry

表 17. 输入窗口小部件选项

bg 或 background	里面在输入区域的背景色。默认值为浅灰色。
--------------------	----------------------

bd 或 borderwidth	入口区；周围边框的宽度请参阅 第 5.1 条，“维度” 。默认值是两个像素。
cursor	游标使用鼠标时内条目窗口小部件；请参阅 一节 5.8，“游标” 。
disabledbackgr ound	要显示小部件处于状态时的背景颜色。选项值，请参阅上文。tk.DISABLEDbg
disabledforegr ound	要显示小部件处于状态时的前景颜色。选项值，请参阅下文。tk.DISABLEDfg
exportselectio n	默认情况下，如果您选择文本内的部件，它是自动导出到剪贴板中。若要避免此出口，使用。 Entryexportselection=0
fg 或 foreground	用于呈现文本的颜色。默认颜色为黑色。
font	用于在小部件中由用户输入文本的字体。请参见 5.4 节，“字体” 。
highlightbackg round	重点突出显示小部件不具有焦点时的颜色。见 节 53“焦点：路由键盘输入” 。
highlightcolor	在重点突出显示小部件具有 焦点 时的颜色。
highlightthick ness	厚度的 关注 热点。
insertbackgrou nd	默认情况下，插入光标（它显示在文本内的点插入新的键盘输入的位置）是黑色的。要得到不同颜色的插入光标，将设置为任何颜色；请参阅 5.3 节、“颜色” 。 insertbackground
insertborderwi dth	默认情况下，插入光标是一个简单的矩形。你可以得到与浮雕效果光标（见 第 5.6 节，“救济样式” ）通过将设置为 维度 的 3 d 边界。如果你这样做，确保该值至少两次的选项。tk.RAISEDinsertborderwidthinsertwidth
insertofftime	默认情况下，插入光标闪烁的频率。你可以设置为一个值，以毫秒为单位指定插入光标花费了多少时间。默认值为 300。如果您使用插入光标不会眨眼。 insertofftimeinsertofftime=0
insertontime	类似于，此选项指定光标花费每眨眼的时间。默认值为 600（毫秒）。insertofftime

insertwidth	默认情况下，插入光标是 2 像素宽。你可以调整这通过将设置为任何 维度 。insertwidth
justify	此选项控制当文本不填充构件的宽度的文本的对齐方式。值可以是（默认值），或。tk.LEFTtk.CENTERtk.RIGHT
readonlybackground	要显示小部件的选项时的背景颜色。state'readonly'
relief	选择文本输入周围的三维阴影效果。请参见 第 5.6 节、“救济样式” 。默认值为。relief=tk.SUNKEN
selectbackground	要使用显示所选的文本的背景颜色。请参阅 5.3 节、“颜色” 。
selectborderwidth	使用所选的文本周围的边框宽度。默认值为一个像素。
selectforeground	选定文本的前景（文本） 颜色 。
show	通常情况下，用户类型出现在条目中的字符。若要使回显每个字符为星号“密码”输入，设置。show='*'
state	使用此选项可以禁用部件，这样用户不能键入任何内容进去。使用禁用部件，再允许用户输入。你的程序还可以找出是否光标目前是在构件通过查询此选项；当鼠标位于它时，它将具有价值。您还可以设置此选项，这是象状态，但小部件的内容仍然可以选择或复制。 Entrystate=tk.DISABLEDstate=tk.NORMALtk.ACTIVE'disabled' tk.DISABLED
takefocus	默认情况下，焦点将通过进入小部件选项卡。将此选项设置为 0，采取不按顺序的小部件。讨论的焦点，请参见 节 53“焦点：路由键盘输入” 。
textvariable	为了能从你条目窗口小部件中检索当前的文本，您必须设置此选项到类的实例；见 节 52“控制变量：小部件背后值” 。您可以检索文本的使用，或将其设置在哪里使用，是关联的控件变量。StringVarv.get() v.set() v
validate	此选项用于设置小部件，以便其内容在某些时候检查验证函数。请参阅 章节 10.2、“条目窗口小部件添加验证” 。
validatecommand	验证构件的文本的回调。请参阅 章节 10.2、“条目窗口小部件添加验证” 。

width	中字符的条目的大小。默认值为 20。对于非等宽字体，小部件的物理长度将基于时间的选项值的字符的平均宽度。width
xscrollcommand	如果您希望用户将经常输入更多文本比屏幕大小的小部件，您可以链接您条目窗口小部件到一个滚动条。将此选项设置为滚动条的方法。更多的信息，请参阅 第 10.1 款，“滚动条目窗口小部件” 。 <code>.set</code>

对象上的方法包括：Entry

`.delete(first, last=None)`

从这种 widget，达开始与一在[索引](#)，但包括位置处的字符删除字符。如果省略第二个参数，则删除只位置的单个字符。*firstlastfirst*

`.get()`

以字符串形式返回该条目的当前文本。

`.icursor(index)`

在给定的[索引](#)设置只在该字符前的插入光标。

`.index(index)`

把移动条目的内容，这样，位于给定[索引](#)处的字符是最左侧的可见字符。如果文本与之匹配的条目内完全，没有任何作用。

`.insert(index, s)`

插入前位于给定[索引](#)处的字符的字符串。*s*

`.scan_dragto(x)`

请参阅下面的方法。scan_mark

`.scan_mark(x)`

使用此选项可以设置快速扫描的内容有一个滚动条，支持水平滚动部件。Entry

要实现此功能，将鼠标的按钮式事件绑定到调用的处理程序，在哪里当前鼠标位置。然后将该事件绑定到处理程序调用，当前鼠标位置在

哪里。方法将不断在速率成正比时调用的位置和当前的位置之间的水平距离构件的内容滚动。

```
scan_mark(x) xx<Motion>scan_dragto(x) xxscan_dragtoEntryscan_mark
```

.select_adjust(*index*)

此方法用于确保[选择](#)包括位于指定[索引](#)处的字符。如果所选内容已包含这样的性格，什么也没有发生。如果不是，扩展所选内容是从其当前位置（如果有），包括位置。*index*

.select_clear()

清除所选内容。如果当前没有选择，没有任何影响。

.select_from(*index*)

设置，所选字符的索引位置，并选择该字符。tk.ANCHOR*index*

.select_present()

如果有[选择](#)，返回 true，否则返回 false。

.select_range(*start*, *end*)

设置在程序控制下的[选择](#)。选择达开始在[索引](#)，但包括索引处字符的文本。位置必须在位置之前。*startendstartend*

若要选择条目窗口小部件中的所有文本，请使用。

```
ee.select_range(0, tk.END)
```

.select_to(*index*)

选择从达位置但不是包括位于给定[索引](#)处的字符的所有文本。

```
tk.ANCHOR
```

.xview(*index*)

相同。此方法很有用在链接到一个水平滚动条的小部件。请参阅[第10.1 款，“滚动条目窗口小部件”](#)。`.xview()`Entry

.xview_moveto(*f*)

位置，相对于整个文本的字符位于窗口的左边缘的位置条目中的文本。该参数必须为在范围 [0, 1]，其中 0 表示文本和 1 左端的右端。*ff*

`.xview_scroll(number, what)`

使用水平滚动的条目。该参数必须是要么，滚动的字符宽度，或向滚动，滚动块条目窗口小部件的大小。有积极向左滚动，为右，负以滚动右到左。例如，为条目窗口小部件，将一“页”的文本向右移动，并将向左移动文本四个字符。

```
whattk.UNITStk.PAGESnumberee.xview_scroll(-1,  
tk.PAGES)e.xview_scroll(4, tk.UNITS)
```

10. 小部件 Entry

小部件的目的是让用户查看和修改单行文本。Entry

- 如果你想要显示多行文本，可以进行编辑，请参阅[节 24、“文本小部件”](#)。
- 如果你想要显示一行或多行的文本，能由用户修改，请参阅[一节 12、“标签小部件”](#)。

一些定义：

- *选择*是突出显示的文本区域中的小部件，如果有的话。Entry

通常由用户用鼠标选定，选定的文本复制到系统剪贴板。然而，*Tkinter* 允许您控制选定的文本获取复制到剪贴板。您还可以选择中的文本在程序控制下。Entry

- *插入光标*显示插入新文本的位置。它被显示仅当用户单击鼠标在某个地方在小部件中。它通常显示为闪烁的竖直线内小部件。您可以自定义其外观在几个方面。
- 作为*指数*给出了在小部件的显示的文本内的位置。有几种方法来指定索引：
 - 作为正常的 Python 指标，从 0 开始。
 - 常数后的现有文本指的位置。tk.END
 - 常数是指插入光标的当前位置。tk.INSERT
 - 常数是指所选内容的第一个字符如果有一个选择。tk.ANCHOR
 - 你可能需要找出哪个字符在小部件中的位置对应于一个给定的鼠标位置。为了简化这一过程，可以使用索引作为字符串的形式，以像素为单位小部件的左边的缘和鼠标之间的水平距离在哪里。这种指数将指定该鼠标水平位置处的字符。'*@n'* nEntry

在根窗口或命名的框架中创建一个新的小部件：Entry*parent*

```
w = tk.Entry(parent, option, ...)
```

此构造函数返回新的小部件。选项包括：Entry

表 17. 输入窗口小部件选项

bg 或 background	里面在输入区域的背景色。默认值为浅灰色。
--------------------	----------------------

bd 或 borderwidth	入口区；周围边框的宽度请参阅 第 5.1 条，“维度” 。默认值是两个像素。
cursor	游标使用鼠标时内条目窗口小部件；请参阅 一节 5.8，“游标” 。
disabledbackground	要显示小部件处于状态时的背景颜色。选项值，请参阅上文。tk.DISABLEDbg
disabledforeground	要显示小部件处于状态时的前景颜色。选项值，请参阅下文。tk.DISABLEDfg
exportselection	默认情况下，如果您选择文本内的部件，它是自动导出到剪贴板中。若要避免此出口，使用。 Entryexportselection=0
fg 或 foreground	用于呈现文本的颜色。默认颜色为黑色。
font	用于在小部件中由用户输入文本的字体。请参见 5.4 节，“字体” 。
highlightbackground	重点突出显示小部件不具有焦点时的颜色。见 节 53“焦点：路由键盘输入” 。
highlightcolor	在重点突出显示小部件具有 焦点 时的颜色。
highlightthickness	厚度的 关注 热点。
insertbackground	默认情况下，插入光标（它显示在文本内的点插入新的键盘输入的位置）是黑色的。要得到不同颜色的插入光标，将设置为任何颜色；请参阅 5.3 节、“颜色” 。 insertbackground
insertborderwidth	默认情况下，插入光标是一个简单的矩形。你可以得到与浮雕效果光标（见 第 5.6 节，“救济样式” ）通过将设置为 维度 的 3 d 边界。如果你这样做，确保该值至少两次的选项。tk.RAISEDinsertborderwidthinsertwidth
insertofftime	默认情况下，插入光标闪烁的频率。你可以设置为一个值，以毫秒为单位指定插入光标花费了多少时间。默认值为 300。如果您使用插入光标不会眨眼。 insertofftimeinsertofftime=0
insertontime	类似于，此选项指定光标花费每眨眼的时间。默认值为 600（毫秒）。insertofftime

insertwidth	默认情况下，插入光标是 2 像素宽。你可以调整这通过将设置为任何 维度 。insertwidth
justify	此选项控制当文本不填充构件的宽度的文本的对齐方式。值可以是（默认值），或。tk.LEFTtk.CENTERtk.RIGHT
readonlybackground	要显示小部件的选项时的背景颜色。state'readonly'
relief	选择文本输入周围的三维阴影效果。请参见 第 5.6 节、“救济样式” 。默认值为。relief=tk.SUNKEN
selectbackground	要使用显示所选的文本的背景颜色。请参阅 5.3 节、“颜色” 。
selectborderwidth	使用所选的文本周围的边框宽度。默认值为一个像素。
selectforeground	选定文本的前景（文本） 颜色 。
show	通常情况下，用户类型出现在条目中的字符。若要使回显每个字符为星号“密码”输入，设置。show='*'
state	使用此选项可以禁用部件，这样用户不能键入任何内容进去。使用禁用部件，再允许用户输入。你的程序还可以找出是否光标目前是在构件通过查询此选项；当鼠标位于它时，它将具有价值。您还可以设置此选项，这是象状态，但小部件的内容仍然可以选择或复制。 Entrystate=tk.DISABLEDstate=tk.NORMALtk.ACTIVE'disabled' tk.DISABLED
takefocus	默认情况下，焦点将通过进入小部件选项卡。将此选项设置为 0，采取不按顺序的小部件。讨论的焦点，请参见 节 53“焦点：路由键盘输入” 。
textvariable	为了能从你条目窗口小部件中检索当前的文本，您必须设置此选项到类的实例；见 节 52“控制变量：小部件背后值” 。您可以检索文本的使用，或将其设置在哪里使用，是关联的控件变量。StringVarv.get() v.set() v
validate	此选项用于设置小部件，以便其内容在某些时候检查验证函数。请参阅 章节 10.2、“条目窗口小部件添加验证” 。
validatecommand	验证构件的文本的回调。请参阅 章节 10.2、“条目窗口小部件添加验证” 。

width	中字符的条目的大小。默认值为 20。对于非等宽字体，小部件的物理长度将基于时间的选项值的字符的平均宽度。width
xscrollcommand	如果您希望用户将经常输入更多文本比屏幕大小的小部件，您可以链接您条目窗口小部件到一个滚动条。将此选项设置为滚动条的方法。更多的信息，请参阅 第 10.1 款，“滚动条目窗口小部件” 。 <code>.set</code>

对象上的方法包括：Entry

`.delete(first, last=None)`

从这种 widget，达开始与一在[索引](#)，但包括位置处的字符删除字符。如果省略第二个参数，则删除只位置的单个字符。*firstlastfirst*

`.get()`

以字符串形式返回该条目的当前文本。

`.icursor(index)`

在给定的[索引](#)设置只在该字符前的插入光标。

`.index(index)`

把移动条目的内容，这样，位于给定[索引](#)处的字符是最左侧的可见字符。如果文本与之匹配的条目内完全，没有任何作用。

`.insert(index, s)`

插入前位于给定[索引](#)处的字符的字符串。*s*

`.scan_dragto(x)`

请参阅下面的方法。scan_mark

`.scan_mark(x)`

使用此选项可以设置快速扫描的内容有一个滚动条，支持水平滚动部件。Entry

要实现此功能，将鼠标的按钮式事件绑定到调用的处理程序，在哪里当前鼠标位置。然后将该事件绑定到处理程序调用，当前鼠标位置在

哪里。方法将不断在速率成正比时调用的位置和当前的位置之间的水平距离构件的内容滚动。

```
scan_mark(x) xx<Motion>scan_dragto(x) xxscan_dragtoEntryscan_mark
```

.select_adjust(*index*)

此方法用于确保[选择](#)包括位于指定[索引](#)处的字符。如果所选内容已包含这样的性格，什么也没有发生。如果不是，扩展所选内容是从其当前位置（如果有），包括位置。*index*

.select_clear()

清除所选内容。如果当前没有选择，没有任何影响。

.select_from(*index*)

设置，所选字符的索引位置，并选择该字符。tk.ANCHOR*index*

.select_present()

如果有[选择](#)，返回 true，否则返回 false。

.select_range(*start*, *end*)

设置在程序控制下的[选择](#)。选择达开始在[索引](#)，但包括索引处字符的文本。位置必须在位置之前。*startendstartend*

若要选择条目窗口小部件中的所有文本，请使用。
ee.select_range(0, tk.END)

.select_to(*index*)

选择从达位置但不是包括位于给定[索引](#)处的字符的所有文本。
tk.ANCHOR

.xview(*index*)

相同。此方法很有用在链接到一个水平滚动条的小部件。请参阅[第 10.1 款，“滚动条目窗口小部件”](#)。*.xview()*Entry

.xview_moveto(*f*)

位置，相对于整个文本的字符位于窗口的左边缘的位置条目中的文本。该参数必须为在范围 [0, 1]，其中 0 表示文本和 1 左端的右端。*ff*

`.xview_scroll(number, what)`

使用水平滚动的条目。该参数必须是要么，滚动的字符宽度，或向滚动，滚动块条目窗口小部件的大小。有积极向左滚动，为右，负以滚动右到左。例如，为条目窗口小部件，将一“页”的文本向右移动，并将向左移动文本四个字符。

```
whattk.UNITStk.PAGESnumberee.xview_scroll(-1,  
tk.PAGES)e.xview_scroll(4, tk.UNITS)
```

10.2. 向小部件添加验证 Entry

在一些应用中，要检查一个小部件，以确保它们是有效的根据您的应用程序必须执行一些规则的内容。你定义什么是有效通过编写一个回调函数，检查的内容和信号是否有效。Entry

下面是设置一个小部件上的验证过程。

1. 写一个回调函数，在检查的文本，如果文本是有效的或者如果不返回。如果回调函数返回，将拒绝该用户尝试编辑的文本，与文本将保持不变。EntryTrueFalseFalse
2. 注册回调函数。在此步骤中，会产生一个 Python 函数的 Tcl 包装。

假设你的回调函数是函数命名。若要注册此函数，请使用[通用部件的方法](#)。此方法返回字符串的字符串， *Tkinter* 可以使用来调用您的函数。isOkay.register(isOkay)

3. 当你调用构造函数时，使用选项在构造函数中指定您的回调，并使用选项来指定时将调用的回调来验证回调中的文本。下面更详细地讨论了这些选项的值。EntryvalidatecommandEntryvalidate

这里有值的选项和它们的含义。validate

'focus'

验证该构件获取或失去焦点时（见[节 53“焦点：路由键盘输入”](#)）。Entry

'focusin'

验证时构件获取焦点。

'focusout'

验证时构件失去焦点。

'key'

验证每当任何击键更改部件的内容。

'all'

在所有上述情况中验证。

`'none'`

关闭验证。这是缺省选项值。请注意这是字符串，不是特别的 Python 值。`'none'` `None`

选项的值取决于所用参数你想你的回调以收到。`validatecommand`

- 也许的唯一回调需要知道的是中当前显示的文本。如果是这样，它可以使用的方法相关联的小部件来检索该文本。`Entry.get()` `textvariable`

在这种情况下，你需要的就是选择“”，其中是您的回调函数的名称。
`validatecommand=ff`

- **Tkinter** 还可以提供的信息给回调的项数。如果你想要使用这些项目，当您调用构造函数时，使用选项，哪里是你的回调函数的名称和每增加是替代代码。为您提供每个替换代码，回调将收到包含适当值的位置参数。`Entryvalidatecommand=(f, s1, s2, ...) f si`

这里是替代代码。

表 18。回调替换代码

<code>'%d'</code>	动作代码：未遂的删除，未遂的插入，1 或-1 如果回调呼吁关注的焦点，焦点或改为 0。 <code>textvariable</code>
<code>'%i'</code>	当用户尝试插入或删除文本时，此参数将在插入或删除开始处的索引。如果回调，是关注的焦点，将是焦点或，参数的更改。 <code>textvariable-1</code>
<code>'%P'</code>	文中将有如果变化允许的值。
<code>'%s'</code>	中的项发生更改之前的文本。
<code>'%S'</code>	如果该调用是由于插入或删除操作，此参数将被插入或删除的文本。
<code>'%v'</code>	小部件的选项的当前值。 <code>validate</code>
<code>'%V'</code>	此回调的原因：一、、或如果改变了。 <code>'focusin''focusout''key''forced'</code> <code>textvariable</code>
<code>'%W'</code>	小部件的名称。

这里是一个小例子。假设你想让你的回调以收到要找出为什么它被称为；要找出，哪里会发生在插入或删除；并找出什么是要插入或删除。你的方法可能如下所示：`'%d''%i''%S'`

```
def isOkay(self, why, where, what):  
    ...
```

接下来可以使用通用的方法来包装该函数。我们假设这就是一些小部件。`.register()` self

```
okayCommand = self.register(isOkay)
```

若要设置此回调，您会在构造函数中使用这两个选项：`Entry`

```
self.w = Entry(self, validate='all',  
               validatecommand=(okayCommand, '%d', '%i', '%S'), ...)
```

假设目前包含的字符串，且该用户选择，然后按退格键。这将导致一个电话：
0 为删除，2 之前的位置和要删除的字符串。如果将回报，新的案文;如果它返回时，文本不会更改。`Entry'abcdefg''cde' isOkay(0, 2, 'cde')'c''cde' isOkay() True' abfg' False`

本插件还支持一个选项，指定调用的回调函数时返回。此命令可能通过使用方法修改小部件中的文本小部件的关联。设置此选项的工作方式相同的设置。您必须使用方法来包装你的 Python 函数;此方法返回一个字符串作为包装函数的名称。然后你就会通过作为选项的值或该字符串，或作为包含替换代码的元组的第一个元素。

```
EntryinvalidcommandvalidatecommandFalse.set() textvariablevalidatecommand.register() invalidcommand
```

11. 小部件 Frame

一个框架基本上只是一个容器为其他小部件。

- 您的应用程序的根窗口基本上是一个框架。
- 每一帧有其自己的网格布局，因此，[网格](#)的每个帧内的小部件独立工作。
- 框架窗口小部件是一个宝贵的工具，使您的应用程序的模块化。你可以一组相关部件成一个复合的小部件，把他们放进一个框架。更好的是，您可以声明一个新类继承自，向它添加您自己的界面。这是很好地隐藏内相关部件从外面的世界一组的交互的细节。Frame

要创建一个新的框架构件在根窗口或框架命名：*parent*

```
w = Frame(parent, option, ...)
```

构造函数将返回新的小部件。选项：Frame

表 19. 框架窗口小部件选项

bg 或 background	框架的背景颜色。请参阅 5.3 节、“颜色” 。
bd 或 borderwidth	图文框的边框的宽度。默认值为 0（无边框）。允许的值，请参阅 第 5.1 条，“维度” 。
cursor	光标使用鼠标时内框架构件；请参阅 一节 5.8，“游标” 。
height	新框架的垂直的 维度 中。这将被忽略，除非你也叫在框架上；看到 4.2 节，“其他网格管理方法” 。 <code>.grid_propagate(0)</code>
highlightbackground	重点突出显示在框架不具有焦点时的 颜色 。见 节 53“焦点：路由键盘输入” 。
highlightcolor	在重点突出所示，当框架具有 焦点 时的 颜色 。
highlightthickness	厚度的 关注 热点。
padx	通常情况下，紧紧围绕它的内容相吻合。若要添加像素的水平空间框架内，设置。 <code>FrameNpadx=N</code>
pady	用于添加到框架内的垂直空间。请参见上面。 <code>padx</code>

relief	<p>违约的救济措施的框架是，框架将融合与周围的环境的手段。把框架的边框，请设置其为积极价值和标准救济类型；之一设置及其救济请参见第 5.6 节、“救济样式”。tk.FLATborderwidth</p>
takefocus	<p>通常情况下，框架窗口小部件不参观由输入焦点（见节 53“焦点：路由键盘输入”有关本主题的概述）。但是，您可以设置如果您想要接收键盘输入的帧。若要处理这种投入，你将需要创建绑定为键盘事件；更多关于事件和绑定，请参阅第 54，“事件”。</p> <p>takefocus=1</p>
width	<p>新的框架的水平尺寸。请参阅第 5.1 条，“维度”。此值被忽略，除非你也叫在框架上；看到4.2 节，“其他网格管理方法”。<code>.grid_propagate(0)</code></p>

12. 构件 Label

标签小部件可以在相同的样式或位图图像中显示一行或多行的文本。在根窗口或框架中创建一个标签小部件：*parent*

```
w = tk.Label(parent, option, ...)
```

构造函数将返回新的小部件。选项包括：Label

表 20。标签小部件选项

activebackground	背景 颜色 ，当鼠标位于小部件时显示。
activeforeground	前景 颜色 ，当鼠标位于小部件时显示。
anchor	此选项如果部件有比文本需要更多空间，放置文本的控件。默认值为，其中中心中可用空间的文本。其他值，请参阅 5.5 条“的锚” 。例如，您使用时，会在可用空间的左上角放置文本。anchor=tk.CENTERanchor=tk.NW
bg 或 background	标签区域的背景颜色。请参阅 5.3 节、“颜色” 。
bitmap	将此选项设置为等于位图或图像的对象，该标签将显示该图形。请参阅 第 5.7 节、“位图” 和 科 5.9、“图像” 。
bd 或 borderwidth	标签；周围边框的宽度请参阅 第 5.1 条，“维度” 。默认值为两个像素。
compound	如果你想要显示文本和图形（位图或图像）的构件，此选项指定相对于文本的图形的相对取向。值可以是任何的、、、或。例如，如果指定，图形将显示文本的下面。 Labelcompoundtk.LEFTtk.RIGHTtk.CENTERtk.BOTTOMtk.TOPcompound=BOTTOM
cursor	当鼠标位于此标签时显示的光标。请参阅 一节 5.8，“游标” 。
disabledforeground	要显示当构件的前景 颜色 。statetk.DISABLED

font	如果您要在此标签中显示的文本（与或选项，该选项指定在哪一种字体将显示该文本。请参见 5.4 节，“字体” 。 texttextvariablefont
fg 或 foreground	如果你在此标签中显示的文本或位图，此选项指定文本的颜色。如果您要显示的位图，这是将出现在位置 1 位的位图中的颜色。请参阅 5.3 节、“颜色” 。
height	在行（而不是像素！） 标签的高度。如果未设置此选项，该标签将调整大小以适应其内容。
highlightbackg round	重点突出显示小部件不具有 焦点 时的 颜色 。
highlightcolor	重点突出显示小部件具有 焦点 时的 颜色 。
highlightthick ness	厚度的 关注 热点。
image	要在标签小部件显示静态图像，对图像对象设置此选项。请参阅 一节 5.9、“图像” 。
justify	指定如何多行文本将与彼此对齐方式： 为左，对齐居中（默认值），或为右对齐。tk.LEFTtk.CENTERtk.RIGHT
padx	在小部件内文本左侧和右侧添加额外的空间。默认值为 1。
pady	添加小部件内文本上方和下方的额外空间。默认值为 1。
relief	指定标签的装饰边框的外观。默认值是;其他值，请参见 第 5.6 节、“救济样式” 。tk.FLAT
state	默认情况下，构件处于状态。设置此选项来作响应鼠标事件。国家将鼠标时在构件。 Entrytk.NORMALtk.DISABLEDtk.ACTIVE
takefocus	通常情况下，焦点不循环通过窗口小部件;见 节 53“焦点： 路由键盘输入” 。如果你想要这个小部件访问的焦点，设置。Labeltakefocus=1
text	要在一个标签小部件显示一行或多行的文本，请设置此选项到包含文本的字符串。内部的换行符 （） 将强制换行。'\n'
textvariable	以奴隶对控制变量的类标签小部件中显示的文本，请将此选项设置给该变量。见 节 52“控制变量： 小部件背后值” 。StringVar

underline	您可以显示以下 n th 字母的文本，下划线 () 从 0 开始计数，通过将此选项设置为 n 。默认值为，这意味着没有下划线。 <code>_underline=-1</code>
width	在 字符 (而不是像素 !) 标签的宽度。如果未设置此选项，该标签将调整大小以适应其内容。
wraplength	您可以通过将此选项设置为所需数量限制中每行的字符数。缺省值为 0，意味着线路将会打破只在换行符。

没有特殊的方法来 label 部件不常见的 (见[第 26, “通用构件方法”](#))。

13. 小部件 LabelFrame

小部件，像这种[框架构件](#)，是一个空间的容器 —— 一个可以包含其他控件的矩形区域。然而，与不同的小部件，部件允许您标签显示为边界周围地区的一部分。LabelFrameFrameLabelFrame



这里是一个小部件包含两个小部件示例。请注意标签“重要控制”中断边界。这个小部件说明违约的救济措施（见[第 5.6 节](#)，“[救济样式](#)”），默认标签锚固，在框架的顶部的左侧位置的标签。LabelFrameButtonGROOVE' nw'

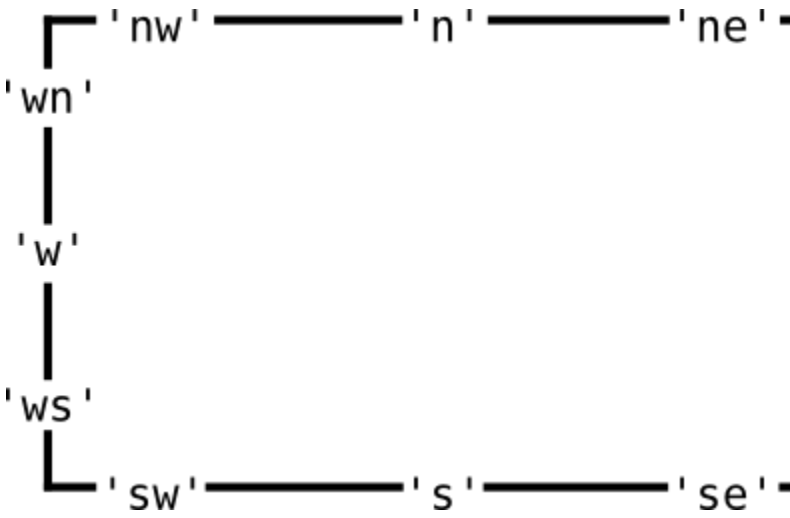
要创建一个新的小部件里面根窗口或框架：LabelFrameparent

```
w = tk.LabelFrame(parent, option, ...)
```

此构造函数返回新的小部件。选项：LabelFrame

表 21。LabelFrame 小部件选项

bg 或 background	里面的部件；显示的背景颜色请参阅 5.3 节 、“ 颜色 ”。
bd 或 borderwidth	小部件；周围绘制的边框宽度请参阅 第 5.1 条 ，“ 维度 ”。默认值为两个像素。
cursor	选择光标显示当鼠标位于小部件；请参阅 一节 5.8 ，“ 游标 ”。
fg 或 foreground	颜色 用于标签文本。
height	新框架的垂直的 维度 中。这将被忽略，除非你也叫在框架上；看到 4.2 节 ，“ 其他网格管理方法 ”。 <code>.grid_propagate(0)</code>

highlightbackground	重点突出显示小部件不具有 焦点 时的 颜色 。
highlightcolor	重点突出显示小部件具有 焦点 时的 颜色 。
highlightthickness	厚度的 关注 热点。
labelanchor	<p>使用此选项来指定标签的位置上的部件的边界。默认位置是，哪些地方该标签在左端的上边框。为九个可能的标签位置，请参阅此关系图：'nw'</p> 
labelwidget	而不是文本标签，您可以使用任何小部件作为标签通过该窗口部件作为此选项的值。如果提供两个选项，将忽略该选项。labelwidgettexttext
padx	使用此选项可以添加其他填充里面的左、 右两边的小部件的框架。值以像素为单位。
pady	使用此选项可以添加其他填充里面的顶部和底部的小部件的框架。值以像素为单位。
relief	<p>此选项控制外面的小部件周围的边框的外观。默认样式是;其他值，请参见第 5.6 节、“救济样式”。</p> <p>tk.GROOVE</p>
takefocus	通常情况下，小部件将不接收焦点;提供一个到此选项可使焦点遍历序列的构件部分的值。有关更多信息，请参见 节 53“焦点： 路由键盘输入” 。True
text	标签的文本。

width

水平[维度](#)的新框架。这将被忽略，除非你也叫在框架上;看到 [4.2 节](#)，[“其他网格管理方法”](#)。`.grid_propagate(0)`

14. 小部件 `Listbox`

一个列表框小部件的目的是显示一组文本行。通常他们被为了允许用户从列表选择一个或多个项目。所有文本行都使用相同的字体。如果你需要什么东西更像是一个文本编辑器，请参阅[第 24， “文本小部件”](#)。

若要创建新的列表框小部件里面根窗口或框架：*parent*

```
w = tk.Listbox(parent, option, ...)
```

此构造函数返回新的小部件。选项：`Listbox`

表 22。列表框小部件选项

<code>activestyle</code>	此选项指定活动行的外观。它可能具有的任何这些值： <code>'underline'</code> 强调了活动线。这是默认选项。 <code>'dotbox'</code> 积极行括在所有四个边的虚线。 <code>'none'</code> 活动线给出了没有特殊的外观。
<code>bg</code> 或 <code>background</code>	列表框中的背景颜色。
<code>bd</code> 或 <code>borderwidth</code>	列表框周围的边框的宽度。默认值是两个像素。可能的值，请参阅 第 5.1 条， “维度” 。
<code>cursor</code>	当鼠标是在列表框中显示的光标。请参阅 一节 5.8， “游标” 。
<code>disabledforeground</code>	颜色 的列表框中的文本时它是。 <code>statetk.DISABLED</code>
<code>exportselection</code>	默认情况下，用户可能会选择用鼠标，文本和选定的文本将被导出到剪贴板中。若要禁用此行为，请使用。 <code>exportselection=0</code>

font	用于在列表框中的文本的字体。请参见 5.4 节，“字体” 。
fg 或 foreground	用于在列表框中的文本的颜色。请参阅 5.3 节、“颜色” 。
height	在列表框中显示的 <i>行</i> （而不是像素！）数。默认值为 10。
highlightbackground	重点突出显示小部件不具有焦点时的颜色。见 节 53“焦点：路由键盘输入” 。
highlightcolor	在重点突出显示小部件具有 焦点 时的颜色。
highlightthickness	厚度的 关注 热点。
listvariable	<p>A 连接到目录的列表框中的值（见 节 52“控制变量：小部件背后值”。StringVar</p> <p>如果调用的方法中，您将重新获得一个 <i>字符串</i> 的形式，每个在哪里行的列表框中的内容。<code>.get()listvariable"('v₀', 'v₁', ...)"v_i</code></p> <p>若要一次更改整个集的列表框中的线条，调用上，那里一个字符串包含行值与它们之间的空间。<code>.set(s)listvariables</code></p> <p>例如，如果是关联与列表框中的选项，此调用将设置列表框包含三行：<code>listConStringVarlistvariable</code></p> <pre>listCon.set('ant bee cicada')</pre> <p>此调用将返回的字符串：<code>"('ant', 'bee', 'cicada')"</code></p> <pre>listCon.get()</pre>
relief	选择三维边框底纹效果。默认值为。其他值，请参见 第 5.6 节、“救济样式” 。tk.SUNKEN
selectbackground	使用显示所选的文本的背景 颜色 。
selectborderwidth	使用所选的文本周围的边框宽度。默认值是在颜色；固体块中显示选定的项如果你增加的条目移动远和所选的条目显示救济（见 第 5.6 节，“救济样式” ）。 <code>selectbackgroundselectborderwidthtk.RAISED</code>

selectforeground	使用显示选定的文本的前景 颜色 。
selectmode	<p>确定可以选择多少项，并拖动鼠标是如何影响所选内容：</p> <ul style="list-style-type: none"> • tk.BROWSE：通常情况下，您只能选择一条线从一个列表框。如果您单击某一项，然后将其拖动到不同的行，所选内容将跟随鼠标。这是默认设置。 • tk.SINGLE：你只能选择一条线上，你不能拖动鼠标 — 无论你单击按钮 1，选择了那条线。 • tk.MULTIPLE：一次你可以选择任意数量的行。点击任何线切换处于选中状态。 • tk.EXTENDED：第一行上单击并拖动到最后一行，你可以同时选择任何相邻的行组。
state	默认情况下，列表框处于状态。若要使列表框响应鼠标事件，请将此选项设置。tk.NORMALtk.DISABLED
takefocus	通常情况下，焦点将通过列表框小部件选项卡。将此选项设置为 0，采取不按顺序的小部件。见 节 53“焦点：路由键盘输入” 。
width	在字符（而不是像素！） 构件的宽度。宽度基于平均的字符，所以这个长度比例字体中的某些字符串可能不适合。默认值为 20。
xscrollcommand	如果你想要允许用户水平滚动列表框中，您可以链接到一个水平滚动条的列表框小部件。将此选项设置为滚动条的方法。在可滚动的列表框小部件的更多信息，请参阅 节 14.1、“滚动列表框小部件” 。 .set
yscrollcommand	如果你想要允许用户垂直滚动列表框中，您可以链接到一个垂直滚动条的列表框小部件。将此选项设置为滚动条的方法。请参阅 一节 14.1，“滚动列表框小部件” 。 .set

一组特殊的指数形式用于许多 listbox 对象上的方法：

- 如果索引指定为一个整数，它是指与该索引从 0 开始计数的列表框中的行。
- 索引到最后一个引用列表框中的线。tk.END

- 索引是指所行。如果列表框允许多重选择，它指的最后选定的行。
tk.ACTIVE
- 形式的索引字符串是指线最接近构件的左上角相对于协调 (,)。
'@x, y' xy

对象上的方法包括：Listbox

. activate(*index*)

选择的行指定由给定。 *index*

. bbox(*index*)

返回作为一个 4 元组，框的左上角的像素的位于指定的代码行的边界框，并给出以像素为单位。返回的值包括只行文本所占用的一部分。

index(xoffset, yoffset, width, height) (xoffset, yoffset) widthheightwidth

如果由参数指定的行不是可见的此方法返回。如果是部分可见，返回边界框可能延长外的可见区域。 *indexNone*

. curselection()

返回一个元组包含所选的元素或元素，从 0 开始计数的行号。如果未选择任何内容，则返回一个空的元组。

. delete(*first*, *last*=None)

删除所在的行的索引范围 [,]，**包容性**（与通常 Python 的成语，在那里删除并未提到的最后一个索引），从 0 开始计数。如果省略第二个参数，则删除一行与索引。 *firstlastfirst*

. get(*first*, *last*=None)

返回一个元组包含文本的行，从指标与包容性。如果省略第二个参数，则返回的文本行的接近。 *firstlastfirst*

. index(*i*)

如果可能的话，职位列表框中的可见部分，包含索引的行是顶部的小部件。 *i*

. insert(*index*, **elements*)

向列表框中指定的代码行之前插入一个或多个新行。如果你想要将新行添加到列表框的末尾，使用作为第一个参数。 *indextk.END*

`.itemcget(index, option)`

检索列表框中的特定行的选项值之一。选项值，请参阅下文。如果给定的选项未设置为给定的行，则返回的值将为空字符串。itemconfig

`.itemconfig(index, option=value, ...)`

更改为指定的代码行的配置选项。选项名称包括：*index*

background

给定行的背景[颜色](#)。

foreground

给定行的文本[颜色](#)。

selectbackground

给定行选中状态时的背景[颜色](#)。

selectforeground

给定行选中状态时的文本[颜色](#)。

`.nearest(y)`

返回索引的可见线条接近 *y* 坐标 *y* 与列表框小部件。

`.scan_dragto(x, y)`

请参阅下文。scan_mark

`.scan_mark(x, y)`

使用此方法来执行扫描 — — 快速稳定滚动 — — 的一个列表框。若要获取此功能，请将一些鼠标按钮事件绑定到处理程序调用与当前鼠标位置。然后将该事件绑定到处理程序调用与当前的鼠标位置，并将所记录的位置和当前的位置之间的距离成正比的速度滚动列表框。

scan_mark<Motion>scan_dragto scan_mark

`.see(index)`

调整列表框中的位置，以便所提及的线是可见。*index*

`.selection_anchor(index)`

在参数所选行上放置“选择锚”。一旦设置了这个锚点，你可以用特殊的指数形式来引用它。 *index*tk.ANCHOR

例如，名为 `listbox`，此序列会选择行 3、4 和 5:`listbox`

```
listbox.selection_anchor(3)
listbox.selection_set(tk.ANCHOR, 5)
```

`.selection_clear(first, last=None)`

取消选择所有行之间指数和包容性。如果省略第二个参数，则取消选择带有索引的行。 *firstlastfirst*

`.selection_includes(index)`

返回 1，如果符合给定处于选中状态，否则返回 0。 *index*

`.selection_set(first, last=None)`

选择所有行之间指数和包容性。如果省略第二个参数，则选择与索引行。 *firstlastfirst*

`.size()`

在列表框中返回行的数。

`.xview()`

若要使列表框水平滚动，对该方法设置关联的水平滚动条的选项。请参阅[一节 14.1, “滚动列表框小部件”](#)。 *command*

`.xview_moveto(fraction)`

滚动列表框中，以便使最左侧的其最长的线的宽度是外左侧的列表框。分数是 [0, 1] 范围内。 *fraction*

`.xview_scroll(number, what)`

水平滚动列表框。对于参数，使用滚动的字符，或将滚动页面，那就是，通过列表框的宽度。该参数会指示多少滚动;负值将文本移到列表框，正值左内右。 *what*tk.UNITStk.PAGES*number*

`.yview()`

若要使列表框垂直滚动，对该方法设置相关联的垂直滚动条的选项。请参阅[一节 14.1, “滚动列表框小部件”](#)。 *command*

`.yview_moveto(fraction)`

滚动列表框中，以便使其长线的宽度的顶部是外左侧的列表框。分数是 $[0, 1]$ 范围内。*fraction*

`.yview_scroll(number, what)`

垂直滚动列表框。对于参数，使用滚动的行，或将滚动页面，那就是，通过列表框的高度。该参数会指示多少滚动；负值移动中的文本向下列表框中，并积极的价值观向上移动文本。

*what*tk.UNITStk.PAGES*number*

14. 1. 滚动窗口小部件 Listbox

这里是一个代码片段说明创建和链接的水平和垂直滚动条的列表框。

```
self.yScroll = tk.Scrollbar(self, orient=tk.VERTICAL)
self.yScroll.grid(row=0, column=1, sticky=tk.N+tk.S)

self.xScroll = tk.Scrollbar(self, orient=tk.HORIZONTAL)
self.xScroll.grid(row=1, column=0, sticky=tk.E+tk.W)

self.listbox = tk.Listbox(self,
    xscrollcommand=self.xScroll.set,
    yscrollcommand=self.yScroll.set)
self.listbox.grid(row=0, column=0, sticky=tk.N+tk.S+tk.E+tk.W)
self.xScroll['command'] = self.listbox.xview
self.yScroll['command'] = self.listbox.yview
```

15. 小部件 Menu

“下拉菜单”是一种流行的方式，向用户呈现大量的选择，但占用上应用程序的最小空间，当用户不作出选择。

- 上方是始终显示在应用程序的部分。
- 菜单是只在用户点击上方后出现的选项列表。
- 若要选择一个选项，用户可以拖动鼠标从上方的选择之一。或者，他们可以单击并释放上方：选择将出现并保持直到用户单击其中之一。
- *Tkinter* 的 Unix 版本（至少）支持“催泪小康菜单”。如果你作为设计者希望它，上方选择将出现一条虚线。用户可以点击这条线要“扯断”菜单：将出现一个新的、单独的、独立的窗口，包含选项。

指节 16、[“上方小部件”](#)，下面，来看看如何创建上方，并将其连接到一个菜单小部件。第一次让我们看看小部件，它显示的选项列表。Menu

显示菜单上的选项可以是任何这些东西：

- 一个简单的命令：一个文本字符串（或图像），用户可以选择来执行一些操作。
- 级联：一个文本字符串或图像，用户可以选择显示另一个整个菜单的选择。
- Checkbutton（见[第 9， “Checkbutton 小部件”](#)）。
- 一组单选按钮（见[第 20、 “单选按钮小部件”](#)）。

若要创建一个菜单小部件，您必须首先创建，我们将称之为：Menubuttonmb

```
w = tk.Menu(mb, option, ...)
```

此构造函数返回新的小部件。选项包括：Menu

表 23. 菜单小部件选项

activebackground	将出现在一个选择，当它是鼠标下的背景色。请参阅 5.3 节、“颜色” 。
activeborderwidth	指定当鼠标下选择周围绘制边框的宽度。默认值为 1 像素。可能的值，请参阅 第 5.1 条，“维度” 。
activeforeground	将出现在一个选择，当它是鼠标下的前景 颜色 。
bg 或 background	不下鼠标选择背景 颜色 。

bd 或 borderwidth	所有的选择；周围边框的宽度请参阅 第 5.1 条，“维度” 。默认值为一个像素。
cursor	显示鼠标时的选择，但只有当菜单已被撕掉的光标。请参阅 一节 5.8，“游标” 。
disabledforeground	是谁的项的文本的 颜色 。statetk.DISABLED
font	用于文本选择的默认字体。请参见 5.4 节，“字体” 。
fg 或 foreground	前景 颜色 用于选择不鼠鼠标。
postcommand	可以将此选项设置为一个程序，并将调用该过程，每次有人提出此菜单。
relief	默认三维效果的菜单是。其他选项，请参见 第 5.6 节、“救济样式” 。relief=tk.RAISED
selectcolor	指定显示在 checkbuttons 和 单选按钮 被选中时的 颜色 。
tearoff	通常情况下，可以撕下菜单： 第一个位置 （位置 0） 列表中选择元素占用的催泪小康，和这些额外的选项添加从位置 1 开始。如果你设置，菜单中不会有眼泪-关闭功能，并选择将添加位置 0 开始。 tearoff=0
tearoffcommand	如果您想您的程序会通知用户单击菜单中的撕条目时，你的程序设置此选项。它将调用具有两个参数：父窗口，该窗口 ID 和新催泪小康菜单的根窗口的窗口 ID。
title	通常情况下，分开的菜单窗口的标题将导致此菜单的上方或级联的文本相同。如果你想要改变，窗口的标题，请将选项设置为该字符串。title

这些方法是在对象上可用。那些在菜单创建的选择有自己特定的选项;请参阅[节 15.1，“菜单项 \(coption\) 创建选项”](#)。Menu

`.add(kind, coption, ...)`

添加的新元素给出了作为在此菜单中的下一个可用选择。参数可以是任何的、`''`、`''`、`''` 或。取决于参数，这种方法是等效的、 等等;请参阅以下这些方法的详细信息。

`kindkind' cascade' ' checkbutton' ' command' ' radiobutton' ' separator' kind.add_cascade().add_checkbutton()`

`.add_cascade(coption, ...)`

作为在此菜单中的下一个可用选择添加一个新的串级元素。在此调用中使用[选项](#)连接级联到下一级菜单，一个类型的对象。`menuMenu`

`.add_checkbutton(coption, ...)`

作为在自我中的下一个可用选择添加新的 `checkbutton`。选项允许您设置 `checkbutton` 一样，您可以将设置一个 [Checkbutton](#) 对象；请参阅[节 15.1, “菜单项 \(coption\) 创建选项”](#)。

`.add_command(coption, ...)`

作为在自我中的下一个可用选择添加一个新的命令。使用、或选项将文本或图像放置菜单；使用选项连接到这种选择采摘时将调用的程序的这种选择。`labelbitmapimagecommand`

`.add_radiobutton(coption, ...)`

作为在自我中的下一个可用选择添加新的单选按钮。选项允许您设置单选按钮中一样，您可以将设置一个对象；请参阅[节 20、 “单选按钮部件”](#)。`Radiobutton`

`.add_separator()`

添加一个分隔符后最后目前定义的选项。这是只是一个统治的水平线你可以使用设置小康群体的选择。分隔符都算作选择，所以如果你已经有三个选择，和你添加一个分隔符，分隔符将占据位置 3 （从 0 开始计数）。

`.delete(index1, index2=None)`

此方法删除编号从通过包容性的选择。若要删除一个选择，请省略参数。您不能使用此方法来删除一个催泪小康的选择，但你可以通过菜单对象拆分选项设置为 0。`index1index2index2`

`.entrycget(index, coption)`

若要检索选择一些 [coption](#) 的当前值，请调用此方法设置为指标的选择与设置为所需的选项的名称。`indexcoption`

`.entryconfigure(index, coption, ...)`

若要更改为选择一些 [coption](#) 的当前值，请调用此方法设置为这种选择和一个或多个参数的索引。`indexcoption=value`

.index(*i*)

返回由索引指定选项的位置。例如，您可以使用查找最后选择的索引（或如果没有选择）。*i.index(tk.END)*None

.insert_cascade(*index*, *coption*, ...)

在给出的从 0 开始计数的位置插入新的级联。任何选择后那个位置下移一个。选项是相同的以上。*index.add_cascade()*

.insert_checkbutton(*index*, *coption*, ...)

在由 *index* 指定的位置插入新的 *checkbutton*。选项是相同的以上。*.add_checkbutton()*

.insert_command(*index*, *coption*, ...)

在位置插入一个新的命令。选项是相同的以上。*index.add_command()*

.insert_radiobutton(*index*, *coption*, ...)

在位置插入新的单选按钮。选项是相同的以上。
index.add_radiobutton()

.insert_separator(*index*)

在所指定的位置插入一个新的分隔符。*index*

.invoke(*index*)

调用与位置选择关联的回调函数。如果 *checkbutton*，其状态是之间进行切换，设置和清除；如果单选按钮，，设置这样的选择。
commandindex

.post(*x*, *y*)

在相对于根窗口的位置显示此菜单。（*x*, *y*）

.type(*index*)

返回的类型由指定的选择：要么，或。
*index*tk.CASCADEtk.CHECKBUTTONtk.COMMANDtk.RADIOBUTTONtk.SEPARATORtk.TEAROFF

.yposition(*n*)

菜单选择，返回以像素为单位与菜单的顶部的垂直偏移量。此方法的目的是使您得以精确相对于当前的鼠标位置放置一个弹出式菜单。*nth*

15.1. 项目创建 () 选项 Menuoption

上文所述的菜单方法允许的地方，你可能不适用于一个值的任何选项名称下面通过使用选项名称作为关键字参数与所需的值。例如，若要使命令文本显示的红色字母，请使用“”作为一个选项的方法调用。

```
coptionforeground=' red' add_command
```

表 24。菜单项 coption 值

accelerator	若要显示“”键击组合在右侧的菜单选项，请使用选项“”哪里一个字符串，包含要显示的字符。例如，若要指示命令已控制 X 作为其加速器，使用选项“”。注意此选项实际上并不实施加速器;使用按键绑定来做到这一点。 acceleratoraccelerator=ssaccelerator=' ^X'
activebackgr ound	用于选择鼠标下的时候的背景颜色。
activeforegr ound	用于选择鼠标下的时候的前景颜色。
background	用于选择它们时的背景颜色不下鼠标。请注意，这不能作为缩写。bg
bitmap	显示一个位图这种选择;请参阅 第 5.7 节，“位图” 。
columnbreak	通常都显示在一长列中选择。如果你设置，这种选择将开始一个新列右侧一包含以前选择。columnbreak=1
columnbreak	使用选项“”选择新栏开始这种选择。columnbreak=True
command	在这种选择激活时要调用的过程。
compound	如果你想要在菜单选择上显示文本和图形（位图或图像），请使用此指定图形相对于文本的位置。值可以是任何的、或。例如，值为“”会在文本的上方图形定位。 coptiontk.LEFTtk.RIGHTtk.TOPtk.BOTTOMtk.CENTERTk.NON Ecompound=tk.TOP
font	用于呈现文本的字体。请参见 5.4 节，“字体” label
foreground	用于选择它们时的前景 颜色 不下鼠标。请注意，这不能作为缩写。fg
hidemargin	默认情况下，小的边距分隔相邻的菜单中的选项。使用“”来抑制此边距。例如，如果您的选择是调色板上的颜色色板，

	此选项将使触摸没有任何其他干预的颜色色板。 coptionhidemargin=True
image	为这种选择；显示一个图像请参阅 一节 5.9、“图像” 。
label	要为这种选择显示的文本字符串。
menu	此选项仅用于级联的选择。将其设置为显示下一级别的选择对象。Menu
offvalue	通常情况下，checkboxbutton 的控制变量设置为 checkboxbutton 时关闭。可以通过将此选项设置为所需的值来更改了值。见 节 52“控制变量：小部件背后值” 。0
onvalue	通常情况下，checkboxbutton 的控制变量设置为 checkboxbutton 时。通过将此选项设置为所需的值，可以更改的值。1
selectcolor	通常情况下，设置的 checkboxbutton 或单选按钮中显示的颜色是红色的。通过将此选项设置为所需；颜色改变那种颜色请参阅 5.3 节、“颜色” 。
selectimage	如果你使用选项而不是文本的图形上显示菜单单选按钮或checkboxbutton，如果你使用，当选定项会显示图像。 imageselectimage=II
state	通常情况下，所有的选择响应鼠标单击，但您可以设置它变灰，使它反应迟钝。这将是当鼠标选择结束。 state=tk.DISABLEDcoptiontk.ACTIVE
underline	通常都不在是字母强调。将此选项设置为索引的一封信，强调那封信。label
value	指定关联的控件变量的值（见 节 52“控制变量：小部件背后值” ）的单选按钮。这可以是一个整数，如果控制变量控制变量是否是一个，或字符串。IntVarStringVar
variable	对于 checkboxbuttons 或单选按钮，此选项应设置为与checkboxbutton 或单选按钮组关联的控制变量。见 节 52“控制变量：小部件背后值” 。

15.2. 顶级菜单

尤其是在苹果 Mac，最好是有时创建菜单，如下所示的顶级窗口的一部分。为此，请执行以下步骤。

1. 使用任何小部件，使用该方法获得的顶级窗口。`WWW.wininfo_toplevel()`
2. 创建一个小部件，作为第一个参数使用的顶级窗口。`Menu`
3. 加入这个小部件的项目将显示在应用程序的顶部。`Menu`

这里是一个简单的例子。假设是应用程序的实例，从继承的类的一个实例。此代码将创建一个名为“**帮助**”的顶级菜单选择一个选择名为“**关于**”调用处理程序命名为：`selfFrameself.__aboutHandler`

```
top = self.wininfo_toplevel()
self.menuBar = tk.Menu(top)
top['menu'] = self.menuBar

self.subMenu = tk.Menu(self.menuBar)
self.menuBar.add_cascade(label='Help', menu=self.subMenu)
self.subMenu.add_command(label='About',
command=self.__aboutHandler)
```

还有一些变化行为取决于您的平台。

- 在 Windows 或 Unix 系统下顶级菜单选择出现在您的应用程序的主窗口的顶部。
- 在 MacOS X 下的顶级菜单中选择屏幕顶部显示当应用程序处于活动状态，走到 Mac 用户期望看到他们。

你必须为你想要在顶部的菜单栏的所有项目使用的方法。调用、 或将被忽略。`.add_cascade().add_checkbutton().add_command().add_radiobutton()`

16. 小部件 Menubutton

上方是一直停留在屏幕上的下拉菜单中的一部分。每个上方是一个小部件（见上文），可以在用户单击时显示那上方的选择在它与相关联。Menu

若要创建在根窗口或框架内的上方：*parent*

```
w = tk.Menubutton(parent, option, ...)
```

构造函数将返回新的小部件。选项：Menubutton

表 25. 上方的小部件选项

activebackgr ound	背景颜色，当鼠标位于上方。请参阅 5.3 节、“颜色” 。
activeforegr ound	前景 颜色 当鼠标位于上方。
anchor	此选项如果部件有比文本需要更多空间，放置文本的控件。默认值为，其中的文本居中。其他选项，请参见 5.5 条、“锚” 。例如，如果您使用文本将居中左舷的小部件。 anchor=tk.CENTERAnchor=tk.W
bg 或 background	背景 颜色 时鼠标还没有结束的上方。
bitmap	要在上方显示位图，请将此选项设置为位图的名称;请参阅 第 5.7 节，“位图” 。
bd 或 borderwidth	上方的边框的宽度。默认值是两个像素。可能的值，请参阅 第 5.1 条，“维度” 。
compound	如果您指定的文本和图形（位图或图像），此选项指定图形相对于文本的显示位置。可能的值为（默认值），、、、、和。例如，将位置右侧的文本图形。如果指定，显示图形，但（如果有）不是。 tk.NONEtk.TOPtk.BOTTOMtk.LEFTtk.RIGHTtk.CENTERcompound=tk.RIGHTcompound=tk.NONEtext
cursor	当鼠标位于此上方时显示的光标。请参阅 一节 5.8，“游标” 。

direction	通常情况下，菜单中将出现下面的上方。设置要显示的菜单左侧的按钮;用来显示菜单右侧的按钮;或使用放置菜单上方的按钮。 direction=tk.LEFTdirection=tk.RIGHTdirection='above'
disabledforeground	前景 颜色 上这上方显示，当它被禁用。
fg 或 foreground	前景 颜色 时鼠标还没有结束的上方。
font	指定用来显示的字体;请参见 5.4 节，“字体” 。text
height	在行的文本（而不是像素！）上方的高度。默认值是以适合其内容上方的大小。
highlightbackground	重点突出显示小部件不具有焦点时的 颜色 。见 节 53“焦点：路由键盘输入” 。
highlightcolor	在重点突出所示，当小部件具有 焦点 时的 颜色 。
highlightthickness	厚度的 关注 热点。
image	若要显示此上方的图像，对图像对象设置此选项。请参阅 二节 5.9、“图像” 。
justify	此选项控制文本所在的位置时文本不填充上方：使用左对齐的文本（这是默认值）;使用中心位置，或右对齐。 justify=tk.LEFTjustify=tk.CENTERjustify=tk.RIGHT
menu	与一组选项相关联的上方，请设置此选项对包含这些选择的对象。该菜单对象必须已通过将相关联的上方传递给作为其第一个参数的构造函数创建。有关详细信息，请参阅下面的示例显示如何将上方及菜单关联。Menu
padx	离开的上方文本左侧和右侧多少空间。默认值为 1。
pady	离开的上方文本上方和下方多少空间。默认值为 1。
relief	通常情况下，menubuttons 将有外观。其他 3 d 效果，请参见 第 5.6 节、“救济样式” 。tk.RAISED
state	通常情况下，menubuttons 来响应鼠标。设置为灰色出上方并作出响应。state=tk.DISABLED

takefocus	通常情况下，menubuttons 不要键盘焦点（见 节 53“焦点：路由键盘输入” ）。用于将上方添加到焦点遍历顺序。 takefocus=True
text	要显示在上方的文本，请设置此选项的包含所需的文本的字符串。在字符串中的换行符（ <code>\n</code> ）会导致换行符。
textvariable	你可以与这上方关联类的控制变量。设置该控制变量会改变显示的文本。见 节 52“控制变量：小部件背后值” 。 StringVar
underline	通常情况下，没有下划线出现在上方的文本的下方。为了强调的人物之一，此选项设置为该字符的索引。
width	在字符（而不是像素！）上方的宽度。如果未设置此选项，该标签将调整大小以适应其内容。
wraplength	通常情况下，不会换行。你可以将此选项设置为字符数目和所有行都将可以都分解成部分不再比这一数字。

这里是一个简单的例子显示的创作上方以及其关联的菜单与两个复选框：

```

self.mb = tk.Menubutton(self, text='condiments',
                        relief=RAISED)
self.mb.grid()

self.mb.menu = tk.Menu(self.mb, tearoff=0)
self.mb['menu'] = self.mb.menu

self.mayoVar = tk.IntVar()
self.ketchVar = tk.IntVar()
self.mb.menu.add_checkbutton(label='mayo',
                             variable=self.mayoVar)
self.mb.menu.add_checkbutton(label='ketchup',
                             variable=self.ketchVar)

```

本示例创建标记的上方。单击时，两个 checkbuttons 标记，并将降下来。
condimentsmayoketchup

17. 小部件 Message

这个小部件是类似于小部件（见[第 12、 “标签小部件”](#)），但它用于显示消息到多个行。所有文本将都显示在相同的字体;如果你需要用多个字体显示文本，请参阅[节 24、 “文本小部件”](#)。Label

若要创建一个新的小部件作为根窗口或框架命名的孩子：Messageparent

```
w = tk.Message(parent, option, ...)
```

此构造函数返回新的小部件。选项可以是任何的这些：Message

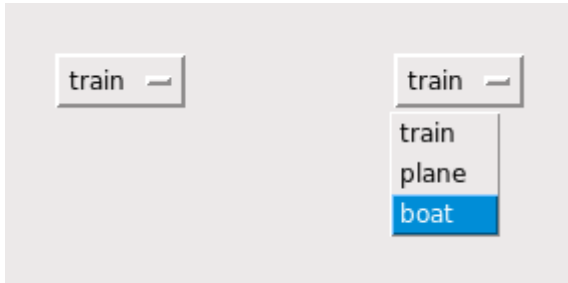
表 26。邮件小部件选项

aspect	使用此选项可以指定一个百分比的宽度与高度的比例。例如，会给你一个文本消息放入一个正方形;随着，文本区域将两次和高一样宽。默认值是 150，那就是，文本将融入更广泛，比它高的框 50%。 aspect=100aspect=200
bg 或 background	文本; 后面的背景颜色请参阅 5.3 节、“颜色” 。
bd 或 borderwidth	小部件; 周围边框的宽度请参阅 第 5.1 条, “维度” 。默认值是两个像素。此选项是可见的只有当选项不是。relieftk.FLAT
cursor	指定显示在构件; 鼠标时的光标请参阅 一节 5.8, “游标” 。
font	指定用于在窗口小部件; 中显示文本的字体请参见 5.4 节, “字体” 。
fg 或 foreground	指定的文本的颜色;请参阅 5.3 节、“颜色” 。
highlightbackground	重点突出显示小部件不具有焦点时的 颜色 。见 节 53 “焦点：路由键盘输入” 。
highlightcolor	在重点突出所示，当小部件具有 焦点 时的 颜色 。
highlightthickness	厚度的 关注 热点。
justify	使用此选项来指定如何多行文本对齐。用于获取笔直的左边的距; 中心每个线;和要笔直的右边距。 justify=tk.LEFTjustify=tk.CENTERjustify=tk.RIGHT

padx	使用此选项可以添加额外的空间内小部件的文本左侧和右侧。值以像素为单位。
pady	使用此选项可以添加额外的空间里面上面这个小程序和文本的下面。值以像素为单位。
relief	此选项指定部件；外边框的外观请参见 第 5.6 节、“救济样式” 。默认样式是。tk.FLAT
takefocus	通常情况下，一个小部件不会获得焦点（见 节 53“焦点：路由键盘输入” ）。使用可以将小部件添加到焦点遍历列表。MessageTakefocus=True
text	此选项的值是在小部件内显示的文本。
textvariable	如果你想要能够改变消息在程序控制下的，此选项将与相关联的实例（见 节 52“控制变量：小部件背后值” ）。此变量的值是要显示的文本。如果您同时指定和选项，将忽略该选项。 StringVarTextTextVariableText
width	使用此选项可以在小部件中，以像素为单位指定文本区域的宽度。默认宽度取决于显示的文本和选项的值。aspect

18. 小部件 `OptionMenu`

这个小部件的目的是向下拉的菜单中的用户提供一组固定的选择。



上面的插图显示在两个国家。左侧示例显示小部件在其最初的形式。右边的示例显示它时鼠标具有点击它，并拖累到选择的外观。 `OptionMenu` 'boat'

若要创建一个新的小部件作为根窗口或框架命名的孩子：`OptionMenu` *parent*

```
w = tk.OptionMenu(parent, variable, choice1, choice2, ...)
```

此构造函数返回新的小部件。是一个实例（见[节 52“控制变量：小部件背后值”](#)）与构件相关联和其余的参数是选择要作为字符串显示在小部件中。

`OptionMenu` *variable* `StringVar`

上面的插图是用这个代码片断创建的：

```
optionList = ('train', 'plane', 'boat')
self.v = tk.StringVar()
self.v.set(optionList[0])
self.om = tk.OptionMenu(self, self.v, *optionList)
```

要找出哪个选择当前选定的小部件，关联的控件变量的方法将作为一个字符串返回的选择。 `OptionMenu.get()`

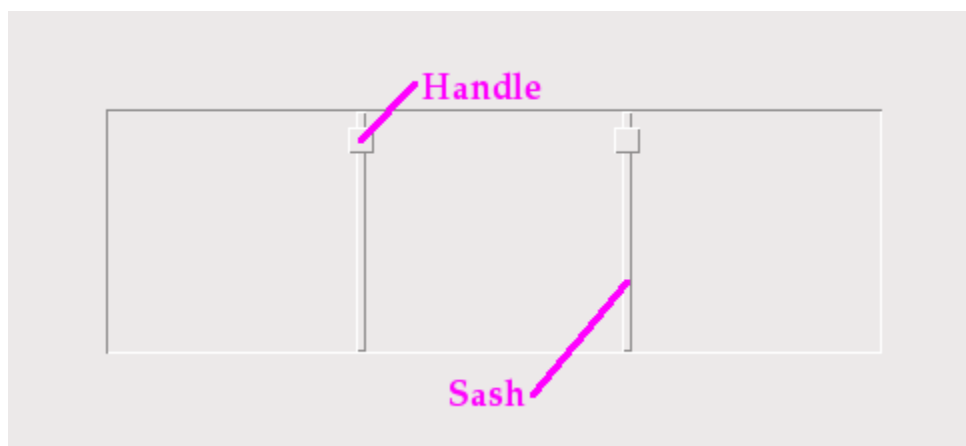
19. 小部件 `PanedWindow`

小部件的目的是让应用程序的用户空间应用程序中如何瓜分一些控制。

`PanedWindow`

A 是有点像：它是孩子的小部件的容器。每个小部件包含水平或垂直堆叠，孩子的小部件。使用鼠标，用户可以拖动的界线之间来来回回的孩子小部件。

`PanedWindowFrame`
`PanedWindow`



- 您可以选择显示在小部件内的句柄。句柄是一个用户可以用鼠标拖动的小广场。
- 您可以选择使窗框可见。腰带是孩子小部件之间放置一个酒吧。
- 一个窗格是一个子部件所占据的区域。

若要创建一个新的小部件作为根窗口或框架命名的孩子：`PanedWindowparent`

```
w = tk.PanedWindow(parent, option, ...)
```

此构造函数返回新的小部件。这里是选项：`PanedWindow`

表 27. `PanedWindow` 小部件选项

bg 或 background	后面的子窗口小部件；显示的背景颜色请参阅 5.3 节 、“颜色”。
bd 或 borderwidth	小部件；外周围边框的宽度请参阅 第 5.1 条 ，“维度”。默认值是两个像素。
cursor	光标将显示当鼠标位于小部件；请参阅 一节 5.8 ，“游标”。

handlepad	使用此选项可以指定句柄和腰带末尾之间的距离。来说，这是腰带的左端和句柄；之间的距离因为，它是腰带的顶部和句柄之间的距离。默认值是 8 个像素;其他值，请参阅 第 5.1 条，“维度” 。 orient=tk.VERTICALorient=tk.HORIZONTAL
handlesize	使用此选项以指定的大小的句柄，这始终是一个正方形;请参阅 第 5.1 条，“维度” 。默认值为 8 个像素。
height	指定窗口小部件；的高度请参阅 第 5.1 条，“维度” 。如果不指定此选项，高度由儿童小部件的高度确定。
opaqueresize	此选项控制如何调整大小的操作工作。默认值为，不断拖着腰带做的大小调整。如果此选项设置为，腰带（和相邻子部件）保持不动的直到用户释放鼠标按钮，然后再它跳转到新的位置。opaqueresize=TrueFalse
orient	堆栈窗口小部件的孩子肩并肩，请使用。要把它们堆顶部到底部，请使用。orient=tk.HORIZONTALorient=tk.VERTICAL
relief	选择小部件；周围的边框的救济样式请参见 第 5.6 节、“救济样式” 。默认值为。tk.FLAT
sashpad	使用此选项可以分配额外的空间，每个窗扇的两边。默认值为零;其他值，请参阅 第 5.1 条，“维度” 。
sashrelief	此选项指定用于呈现腰带；救济样式请参见 第 5.6 节、“救济样式” 。默认样式是。tk.FLAT
sashwidth	指定宽度的腰带;请参阅 第 5.1 条，“维度” 。默认宽度为两个像素。
showhandle	使用显示的柄。默认值，用户仍然可以使用鼠标移动窗扇上的。句柄是只是一个视觉提示。showhandle=TrueFalse
width	宽度的部件;请参阅 第 5.1 条，“维度” 。如果你不指定一个值，将通过儿童小部件的大小来决定宽度。

若要添加子部件到，创建子窗口小部件作为孩子的家长，但是而不是使用方法来注册它们，使用的方法。

PanedWindowPanedWindow.grid().add()PanedWindow

这里是小部件上的方法。PanedWindow

.add(*child*[, *option=value*] ...)

使用此方法将给定的部件添加为下一胎的这。首先创建小部件作为其父窗口小部件，但不是调用方法来注册它。然后调用和孩子会出现在内部中的下一个可用的位置。

`childPanedWindow``childPanedWindow.grid().add(childPanedWindow`

伴随每个孩子的是一组控制其位置和外观的配置选项。请参阅[一节 19.1、“PanedWindow 子配置选项”](#)。您可以提供这些配置选项作为关键字参数的方法。您还可以设置或与方法，随时更改其值或检索任何使用方法；这些选项的当前值下面将介绍这些方法。`.add().paneconfig().panecget()`

`.forget(child)`

删除一个孩子小部件。

`.identify(x, y`

为窗口坐标中的给定位置，此方法返回一个值，用于描述在该位置功能。`(x, y)`

- 如果该功能是一个子窗口，该方法将返回一个空字符串。
- 如果该功能是腰带，该方法返回一个元组第一的腰带，为第二个，依此类推 1 0 在哪里。`(n, 'sash')n`
- 如果该功能是一个句柄，该方法返回一个元组在哪里 0 第一句柄，1，第二次，等等。`(n, 'handle')n`

`.panecget(child, option)`

此方法检索一个孩子小部件配置选项，其中是孩子的小部件，是作为一个字符串选项的名称的值。孩子小部件配置选项的列表，请参阅[二节 19.1、“PanedWindow 子配置选项”](#)。`childoption`

`.paneconfig(child, option=value, ...)`

使用此方法为孩子小部件配置选项。选项所述[节 19.1、“PanedWindow 子配置选项”](#)。

`.panes()`

此方法返回的子列表小部件，从左到右 `()` 或从顶部到底部 `(为)` 的顺序。`orient=tk.HORIZONTAL``orient=tk.VERTICAL`

`.remove(child)`

删除给出;这是相同的操作方法。`child.forget()`

`.sash_coord(index)`

此方法返回窗扇的位置。参数选择腰带： 腰带之间这第一次两个孩子之间的第二和第三个孩子，等等的框格为 1 的 0。其结果是一个元组包含，框格的左上角的坐标。*index*(*x*, *y*)

`.sash_place(index, x, y)`

使用此方法来重新定位腰带评选 (0 表示第一个腰带，等等)。和坐标指定所需的新位置的左上角的腰带。*Tkinter* 忽略对构件的方向正交坐标： 使用值来重新定位为，腰带和使用坐标移动选项腰带。

indexxyxorient=tk.HORIZONTAL,yorient=tk.VERTICAL

19. 1. 子配置选项 PanedWindow

每个孩子都有一组控制其位置和外观的配置选项。当一个孩子是添加的方法，或设置的方法，或查询与上面介绍的方法，可以提供这些选项。

`PanedWindow.add().paneconfig().panecget()`

表 28. PanedWindow 子小部件选项

after	通常情况下，当新的儿童对、新儿童是后添加任何现有的子部件。相反可能使用选项，只是后存在的子部件的位置插入新的小部件。 <code>.add()PanedWindowafter=ww</code>
before	作为选项在对方法的调用中使用，将新的小部件放在之前存在的子部件的位置。 <code>before=w.add()w</code>
height	此选项指定所需的高度的孩子小部件;请参阅 第 5.1 条，“维度” 。
minsize	使用此选项可以指定子构件的最小大小的方向的定位。来说，这是最小的宽度;因为，它是最小高度。允许的值，请参阅 第 5.1 条，“维度” 。 <code>PanedWindoworient=tk.HORIZONTALorient=tk.VERTICAL</code>
padx	大量额外的空间添加到左边和右边的子窗口小部件;请参阅 第 5.1 条，“维度” 。
pady	大量额外的空间，要添加的上方和下方的孩子小部件;请参阅 第 5.1 条，“维度” 。
sticky	此选项功能类似的参数的方法;请参阅 4.1 节、“<code>.grid()</code>法” 。它指定如何定位一个孩子小部件，如果该窗格大于小部件。例如，会在左上角的窗格（“西北”）的位置小部件。 <code>sticky.grid()sticky=tk.NW</code>
width	所需的宽度的孩子小部件;请参阅 第 5.1 条，“维度” 。

20. 小部件 Radiobutton

单选按钮是允许用户只能选择一组选项的相关部件的集合。每个单选按钮包含两个部分，*指标*和*标签*：



- 该标记是变成红色中选定项的钻石形部分。
- 标签是文本，虽然你可以使用 [图像](#)或[位图](#)作为标签。
- 如果你愿意，你可以免除与指示器。这使得单选按钮看起来像“推送推送”按钮，选定的条目出现凹陷与其余出现凸起。
- 形成一官能团的几个单选按钮，创建一个单一的控制变量（见[节 52“控制变量：小部件背后值”](#)，下面），并将每个单选按钮的选项设置为该变量。variable

控制变量可以是或。如果两个或多个单选按钮共享同一个控制变量，将其中的任何设置将清除其它。IntVarStringVar

- 每个单选按钮组中的必须有唯一的期权的控制变量的类型相同。例如，三个单选按钮组可能共享和值为 0、 1 和 99。或者你可以使用一个控制变量并给单选按钮选项像、 和。
valueIntVarStringVarvalue' too hot'' too cold'' just right'

若要创建一个新的单选按钮小部件作为根窗口或框架命名的孩子：*parent*

```
w = tk.Radiobutton(parent, option, ...)
```

此构造函数返回新的小部件。选项：Radiobutton

表 29。单选按钮小部件选项

activebackground	背景颜色，当鼠标位于单选按钮。请参阅 5.3 节、“颜色” 。
activeforeground	前景 颜色 当鼠标位于单选按钮。
anchor	如果该构件栖息于比它所需要的更大的空间，此选项指定在该空间中单选按钮的位置。默认值为。其他定位的选项，请

	参见 5.5 条、“锚” 。例如，如果您设置单选按钮将放在右上角的可用空间。 <code>anchor=tk.CENTERanchor=tk.NE</code>
bg 或 background	后面的指标和标签的正常背景 颜色 。
bitmap	在单选按钮上显示单色图像，请将此选项设置为位图；请参阅 第 5.7 节，“位图” 。
bd 或 borderwidth	该指示器部件本身的边框的大小。默认值是两个像素。可能的值，请参阅 第 5.1 条，“维度” 。
command	每次用户更改此单选按钮的状态调用程序。
compound	如果您指定的文本和图形（位图或图像），此选项指定图形相对于文本的显示位置。可能的值为（默认值）， <code>tk.NONEtk.TOPtk.BOTTOMtk.LEFTtk.RIGHTtk.CENTERcompound=tk.BOTTOMcompound=tk.NONEtext</code> 和。例如，将定位图形文本的下面。如果指定，显示图形，但（如果有）不是。
cursor	如果将此选项设置为游标名称（见 节 5.8，“游标” ），鼠标光标将变为这种模式，当它结束的单选按钮。
disabledforeground	用于呈现禁用的单选按钮的文本的前景颜色。默认值是一个斑点的版本的默认前景色。
font	所用的字体。请参见 5.4 节，“字体” 。 <code>text</code>
fg 或 foreground	用于呈现的 颜色 。 <code>text</code>
height	单选按钮上的文本行（而不是像素）的数量。默认值为 1。
highlightbackground	重点突出显示在单选按钮不具有焦点时的颜色。见 节 53“焦点：路由键盘输入” 。
highlightcolor	重点突出显示单选按钮具有 焦点 时的 颜色 。
highlightthickness	厚度的 关注 热点。默认值为。设置来禁止显示的关注热点。 <code>highlightthickness=0</code>
image	若要显示图像而不是为此单选按钮的文本，请设置此选项的图像对象。请参阅 一节 5.9、“图像” 。单选按钮时，将出现图像未被选中。比较，下面。 <code>selectimage</code>

indicatoron	通常单选按钮显示其指标。如果您将此选项设置为零，指示器即会消失，并且整个构件成为时清除和凹下设置时，它的外观凸起“推送推送”按钮。你可能想要增加值以使它更易于看到这样的控件的状态。borderwidth
justify	如果包含多个行，此选项控制文本的对齐方式：（默认值），或。texttk.CENTERtk.LEFTtk.RIGHT
offrelief	如果你压制指标通过断言，选项指定要当未选中单选按钮时，显示的 救济方式 。默认值是。 indicatoron=Falseoffrelieftk.RAISED
overrelief	指定要当鼠标位于单选按钮时，显示的 救济方式 。
padx	多大的空间去左边和右边的单选按钮和文本。默认值为 1。
pady	离开的上方和下方的单选按钮和文本多少空间。默认值为 1。
relief	默认情况下，单选按钮将有救济，所以它不站出来，从它的背景。更多 3 d 效果选项，请参阅 第 5.6 节、“救济样式” 。您还可以使用，其中固体黑色在周围显示框架单选按钮。tk.FLATrelief=tk.SOLID
selectcolor	单选按钮设置时，它的 颜色 。默认值是红色的。
selectimage	如果你正在使用选项显示图形而不是文本，清除单选按钮时，您可以将选项设置为设置单选按钮时，将显示不同的图像。请参阅 一节 5.9、“图像” 。imageselectimage
state	默认值是，但您可以设置灰色的控制，使它反应迟钝。如果光标位于当前上的单选按钮，状态。 state=tk.NORMALstate=tk.DISABLEDtk.ACTIVE
takefocus	默认情况下，输入焦点（见 节 53“焦点：路由键盘输入” ）将通过单选按钮。如果你设置，重点将不会访问此单选按钮。takefocus=0
text	单选按钮旁边显示的标签。使用换行符（\n）来显示多行文本。'\n'
textvariable	如果你需要在执行期间更改单选按钮上的标签，创建（见 节 52“控制变量：小部件背后值” ）来管理的当前值，并将此选项设置为该控件变量。只要控制变量值发生变化，单选按钮的注释将自动更改为该文本以及。StringVar
underline	默认值为-1，没有文本标签的字符下划线标出。设置此选项的索引（从零开始计数）的文本中的字符的下划线字符。

value	单选按钮打开时由用户，其控制变量设置为其当前的选项。如果控制变量是，给每个单选按钮组中的不同的整数选项。如果控制变量是，给每个单选按钮不同的字符串选项。 valueIntVarvalueStringVarvalue
variable	此单选按钮组；同其他单选按钮控制变量见 节 52“控制变量：小部件背后值” 。这可以是或。IntVarStringVar
width	单选按钮的默认宽度是由显示的图像或文本的大小决定的。你可以将此选项设置为字符（而不是像素）的数目和单选按钮将总是有很多字符的余地。
wraplength	通常情况下，不会换行。你可以将此选项设置为字符数目和所有行都将可以都分解成部分不再比这一数字。

单选按钮对象上的方法包括：

.deselect()

清除（关闭） 单选按钮。

.flash()

闪烁几次之间其活动和正常的颜色，单选按钮，而是让它开始的方式。

.invoke()

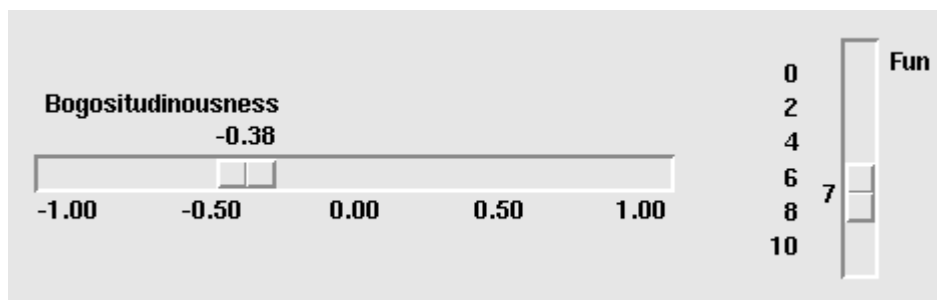
你可以调用此方法来获取相同的操作，如果用户点击单选按钮以更改其状态将会发生。

.select()

集（打开） 单选按钮。

21. 构件 `Scale`

规模小部件的目的是允许用户设置一些值或指定范围内。下面是两个规模小部件，一横一竖：`intfloat`



每个规模显示一个滑块，用户可以通过拖动沿槽来更改的值。在图中，第一个滑块目前在-0.38，第二次 7。

- 可以将滑块拖动到一个新的值，使用鼠标按钮 1。
- 如果您单击按钮 1 在槽中的，滑块会将每次点击那个方向移动一个增量。按住按钮 1 海槽将延迟后，开始自动重复它的功能。
- 如果规模具有[键盘焦点](#)，向左箭头键、向上箭头键击将移动滑块向上（垂直尺度）或左（水平尺度）。向右箭头键和向下箭头键击将移动滑块向下或向右。

若要创建一个新的规模小部件作为根窗口或框架命名的孩子：`parent`

```
w = tk.Scale(parent, option, ...)
```

构造函数将返回新的小部件。选项：`Scale`

表 30. 规模小部件选项

<code>activebackground</code>	当鼠标在滑块的颜色。请参阅 5.3 节 、“颜色”。
<code>bg</code> 或 <code>background</code>	背景 颜色 的外槽的小部件的部件。
<code>bd</code> 或 <code>borderwidth</code>	槽和滑块的 3 d 边框的宽度。默认值是两个像素。可接受的值，请参阅 第 5.1 条 ，“维度”。
<code>command</code>	要调用每次在滑块移动过程。此过程将会传递一个参数，新的缩放值。如果在滑块移动迅速，你可能不会回调，为每一个可能的位置，但你一定会回调时它安定。

cursor	显示鼠标时规模过大的光标。请参阅 一节 5.8, “游标” 。
digits	<p>您的程序读取的当前值显示在一个规模小部件的方式是通过控制变量;见节 52“控制变量：小部件背后值”。规模的控制变量可以是，（用于类型），或。如果它是一个字符串变量，该选项控制多少位数的数字刻度值转换为字符串时使用。</p> <p>IntVarDoubleVarfloatStringVardigits</p>
font	用于标签和注释的字体。请参见 5.4 节, “字体” 。
fg 或 foreground	用于标签和注释的文本 颜色 。
from_	<p>值，该值定义范围的一端。对于垂直尺度，这是的顶端;为水平的尺度，左端。底线（）不是一个错字：因为是一个保留的字在 Python 中，此选项的拼写。默认值为 0.0。看到选项，下面，对另一端的范围。</p> <p>float_fromfrom_to</p>
highlightbackground	重点突出显示时规模没有焦点的 颜色 。见 节 53“焦点：路由键盘输入” 。
highlightcolor	重点突出规模具有 焦点 时的 颜色 。
highlightthickness	重点突出 的厚度。默认值为 1。设置来禁止显示的 关注热点 。highlightthickness=0
label	您可以通过将此选项设置为标签的文本显示内规模小部件的标签。如果规模水平或如果垂直的右上角，标签会出现在左上角。默认值是没有标签。
length	规模小部件的长度。这是 x 尺寸如果规模是水平的或如果垂直 y 维度。默认值为 100 个像素。允许的值，请参阅 第 5.1 条, “维度” 。
orient	<p>如果你想要的规模来运行沿 x 尺寸，或运行平行于 y 设置-轴。默认值是垂直的。</p> <p>orient=tk.HORIZONTALorient=tk.VERTICAL</p>
relief	<p>默认情况下，与规模不站出来从它的背景。您还可以使用要围绕规模，或任何其他救济中描述的类型第 5.6 节、“救济样式”固体黑色框架。</p> <p>relief=tk.FLATrelief=tk.SOLID</p>

repeatdelay	此选项控制按钮 1 有多久会被按在料槽滑块一再开始朝这个方向努力之前。默认值是，和单位是毫秒。 repeatdelay=300
repeatinterval	此选项控制如何经常滑块跳一旦按钮 1 被压低了在槽为至少毫秒。例如，会跳滑块每 100 毫秒。 repeatdelayrepeatinterval=100
resolution	通常情况下，用户将只能更改在整个单位中的比例。将此选项设置为其他值，以更改缩放值的最小增量。例如，如果，你设置，规模将有 5 个可能的值：-1.0，-0.5，0.0，0.5，1.0。将忽略所有的小运动。使用禁用任何舍入的值。from_=-1.0to=1.0resolution=0.5resolution=-1
showvalue	通常情况下，规模的当前值被显示在文本形式中滑块（上面的水平尺度，左侧的垂直尺度）。将此选项设置为 0 来禁止显示该标签。
sliderlength	通常的滑块是规模的 30 像素沿长度。您可以更改该长度由将选项设置为您所需的长度;请参阅 第 5.1 条，“维度” 。sliderlength
sliderrelief	默认情况下，滑块是以救济样式显示。对于其他救济样式，任何描述在 第 5.6 节、“救济样式” 的值设置此选项。tk.RAISED
state	通常情况下，规模小部件响应鼠标事件，和当他们有 焦点 ，也键盘事件。设置，使构件反应迟钝。 state=tk.DISABLED
takefocus	通常情况下，焦点将循环规模小部件。将此选项设置为 0，如果你不想这种行为。见 节 53“焦点：路由键盘输入” 。
tickinterval	通常情况下，没有“刻度”被显示沿尺度。若要显示定期规模值，此选项设置为一个数字，和爬虫将显示在此值的倍数。例如，如果匹配和标签将显示沿值 0.0、0.25、0.50、0.75、1.00 规模。这些标签显示以下规模如果水平，如果垂直的左侧。默认值为 0，取消显示的刻度。 from_=0.0to=1.0tickinterval=0.25
to	值，该值定义一端的范围;另一端是选项，上文所定义的。值可以大于或小于值。为垂直尺度的值定义底

	部的规模;为水平的尺度，右端。默认值为 100.0。 floatfrom_tofrom_to
troughcolor	颜色 的槽。
variable	控制变量为这种规模，如果有的话；的见 节 52”控制变量：小部件背后值” 。控制变量可能从类（用于类型），或。在后者的情况下，数值值将转换为字符串。请参阅选项，如上所示，这种转换的详细信息。 IntVarDoubleVarfloatStringVardigits
width	槽的构件部分的宽度。如果规模，这是 x 尺寸为垂直尺度和 y 维度。默认值为 15 像素。 orient=tk.HORIZONTAL

缩放对象有这些方法：

.coords(value=None)

返回的坐标，相对于左上角的小部件，对应于一个给定的规模。对，你得到滑块的中心的坐标在其当前的位置。要找到会在哪里滑块如果刻度值被设置为某个值，使用。value=Nonexvalue=x

.get()

此方法返回表的当前的值。

.identify(x, y)

给出一对坐标相对于左上角的小部件，此方法返回一个字符串，标识哪些功能的构件部分的在该位置。返回值可以是任何的这些：(x, y)

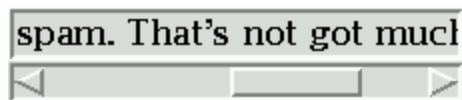
'slider'	滑块。
'trough1'	为水平的尺度，左侧的滑块;为滑块上方的垂直尺度。
'trough2'	为水平的尺度，右边的滑块;垂直的秤，滑块下方。
''	位置不是在任何上述部分。(x, y)

.set(value)

设置刻度值。

22. 小部件 Scrollbar

大量的小部件，如列表框和画布，可以像进入一个更大的虚拟区域滑动窗口。您可以连接到它们给用户一种方式为幻灯片视图周围相关内容滚动条小部件。这里是与相关联的滚动条部件条目窗口小部件的一个屏幕快照：



- 滚动条可以是水平的如所示的上方，或垂直。一个小部件，有两个可滚动的维度，例如在画布或列表框，可以有水平和垂直滚动条。
- *滑块或滚动滑块*，是看起来提出的矩形，该矩形显示当前的滚动位置。
- 两个三角形的 *箭头* 在每个末端用于移动位置的小步骤。在左侧或顶端一调用，并且右侧或底部一个叫做。arrow1 arrow2
- *槽* 是沉没看地区可见背后的箭头和滑块。海槽分为两个区域命名（上方或左侧的滑块）和（下面或右边的滑块）。trough1 trough2
- 滑块的大小和位置，相对于整个构件的长度显示的大小和位置的视图相对于它的总大小。例如，如果一个垂直滚动条是一个列表框，与相关联，其滑块延伸从 50% 到 75% 的滚动条的高度，这意味着的可见部分的列表框中显示整个列表在半路标记开始和结束在四分之三的标记的那部分。
- 在水平滚动条，单击 B1（按钮 1）上的左箭头移动视图由少量向左。B1 点击右箭头移动视图由这一数额向右。对于垂直滚动条，单击向上和向下-指箭头移动视图少量向上或向下。请参阅相关联的小部件，以找出确切金额这些操作移动视图的讨论。
- 用户可以拖动滑块与 B1 或 B2（中间的按钮）来移动视图。
- 水平滚动条，单击左侧的滑块槽中的 B1 移动视图留下的一个页面，并单击右边的滑块槽中的 B1 向右移动视图页面。对于垂直滚动条，相应操作移动视图页面向上或向下。
- 单击 B2 任意位置沿槽移动滑块，其左边或上边的结束是在鼠标，或尽可能接近它尽可能。

*归一化的位置*的滚动条是指闭区间 $[0.0, 1.0]$ 定义滑块的位置的数字。对于垂直滚动条位置 0.0 是在顶部和底部；1.0 为水平滚动条位置 0.0 位于左端和 1.0 在右边。

若要创建一个新的小部件作为根窗口或框架的孩子：`Scrollbarparent`

```
w = tk.Scrollbar(parent, option, ...)
```

构造函数将返回新的小部件。滚动条的选项包括：`Scrollbar`

表 31。滚动条小部件选项

activebackground	滑块和箭头鼠标何时位于它们的颜色。请参阅 5.3 节、“颜色” 。
activerelief	默认情况下，滑块是救济式所示。当鼠标位于滑块显示滑块与不同的 救济方式 。tk.RAISED
bg 或 background	颜色 滑块和箭头鼠标时不超过他们。
bd 或 borderwidth	槽，整个周边 3 d 边框的宽度和 3 d 效果上的箭头和滑块的宽度。默认值是在低谷，周围没有边框和两个像素的边框环绕箭头和滑块。可能的值，请参阅 第 5.1 条，“维度” 。
command	移动滚动条时要调用过程。调用序列的讨论，请参见 第 22.1 条，“滚动条命令回调” 。
cursor	在滚动条上的鼠标时显示的光标。请参阅 一节 5.8，“游标” 。
elementborderwidth	箭头和滑块周围边框的宽度。默认为，这意味着要使用的选项的值。elementborderwidth=-1borderwidth
highlightbackground	颜色 的重点突出显示时，滚动条不具有焦点。见 节 53“焦点：路由键盘输入” 。
highlightcolor	重点突出显示滚动条具有 焦点 时的 颜色 。
highlightthickness	厚度的 关注 热点。默认值为。设置为以禁止显示的关注热点。10
jump	此选项控制当用户拖动滑块时，会发生什么。通常（），每个小的滑块拖导致回调被调用。如果您启用了此选项，直到用户释放鼠标按钮，不会调用回调。 jump=0command1
orient	为一个垂直的水平滚动条，设置一个（缺省方向）。orient=tk.HORIZONTALorient=tk.VERTICAL
relief	可控制的 救济方式 的构件;默认样式是。此选项在 Windows 中没有任何影响。tk.SUNKEN
repeatdelay	此选项控制按钮 1 有多久会被按在料槽滑块一再开始朝这个方向努力之前。默认值是，和单位是毫秒。 repeatdelay=300

repeatinterval	此选项控制如何经常滑块运动将重复时在料槽按住按钮 1。默认值是，和单位是毫秒。 repeatinterval=100
takefocus	通常情况下，你可以选项卡通过滚动条部件；焦点见 节 53“焦点：路由键盘输入” 。如果你不想这种行为，设定。滚动条的默认键绑定允许用户使用 ←、→ 箭头键来移动水平滚动条，和他们可以使用 ↑ 和 ↓ 键移动垂直滚动条。takefocus=0
troughcolor	颜色 的槽。
width	滚动条（如果水平，其 y 维度和其 x 尺寸如果垂直）的宽度。默认值为 16。可能的值，请参阅 第 5.1 条，“维度” 。

滚动条对象上的方法包括：

.activate(element=None)

如果不提供任何参数，则此方法返回一个字符串、、、，或根据鼠标在哪里。例如，该方法返回鼠标是否在滑块上。返回空字符串如果鼠标当前未在任何这三个控件。'arrow1' 'arrow2' 'slider' ''' slider'

以突出显示一个控件（使用其救济风格和它的颜色），调用此方法并传递一个字符串，标识的控制你想要突出显示，之一、或。
activereliefactivebackground'arrow1' 'arrow2' 'slider'

.delta(dx, dy)

给定的鼠标移动的像素数，此方法返回的值应该添加到当前的滑块位置，来实现，同样的动作。值必须是在闭区间 [-1.0, 1.0]。(dx, dy) float

.fraction(x, y)

给出一个像素位置，此方法返回相应的归一化在该位置最接近的间隔 [0.0, 1.0] 滑块位置。(x, y)

.get()

返回两个数字（，）描述的滑块当前位置。值分别；给出了滑块，为水平和垂直滚动条的左边或上边边缘位置值的位置的右边或底部。

每个值是在区间 $[0.0, 1.0]$ 左侧或顶端的立场是 0.0 和 1.0 是最右边或底部的位置。例如，如果滑块从半路延伸到四分之三的沿槽的方式，你可能会回来元组 $(0.5, 0.75)$ 。*abab*

.identify(*x*, *y*)

此方法返回一个字符串，指示的（如果有）的给定坐标下的滚动条组件。返回值是空字符串或之一，如果该位置不在任何滚动条组件。
 (x, y) 'arrow1' 'trough1' 'slider' 'trough2' 'arrow2' ''

.set(*first*, *last*)

要连接到另一个小部件，滚动条设置的或滚动条上的方法。参数具有相同的含义由该方法返回的值。请注意，移动滚动条上的滑块并不移动相应的部件。*wxscrollcommand**scrollcommand.set.get()*

22. 1. 回调 Scrollbar *command*

当用户通过操作一个滚动条时，滚动条会调用其回调。此调用的参数取决于用户所执行的操作：`command`

- 当用户请求一个“单位”运动离开之类的例如通过单击上箭头的左边或上边，按钮 B1 回调看的参数：

- `command(tk.SCROLL, -1, tk.UNITS)`

- 当用户请求一个单位正确的或向下运动时，的论据如下：

- `command(tk.SCROLL, 1, tk.UNITS)`

- 当用户请求一个页面左边的或向上的运动：

- `command(tk.SCROLL, -1, tk.PAGES)`

- 当用户请求一个向右翻页或向下的运动：

- `command(tk.SCROLL, 1, tk.PAGES)`

- 当用户将滑块拖动到一个值在范围 $[0, 1]$ ，其中 0 表示一路左或向上和 1 手段一路的右或下，电话是：*f*

- `command(tk.MOVETO, f)`

这些调用序列参数匹配的预期和画布、列表框和文本小部件的方法。小部件并没有一种方法。请参阅第 10.1 款，[“滚动条目窗口小部件”](#)。`.xview().yview()``Entry.xview()`

下一个：[22. 2. 滚动条连接到另一个小部件](#)

内容：[_Tkinter 8.5](#) 参考：[GUI 的 Python](#)

以前：[22. 滚动条部件](#)

帮助：[科技计算机中心：帮助系统](#)

主页：[关于新的墨西哥科技](#)

John W. Shipman

Comments welcome: tcc-doc@nmt.edu

最后更新：2013 年-12-31 17:59

网址：<http://www.nmt.edu/tcc/help/pubs/tkinter/web/scrollbar-callback.html>

Tkinter 8.5 参考 ： Python 的 GUI



22.2. 连接到另一个小部件 Scrollbar

这里是显示创建画布与水平和垂直滚动条的代码片段。在这个片段中，被假定为一个部件。selfFrame

```
self.canv = tk.Canvas(self, width=600, height=400,
    scrollregion=(0, 0, 1200, 800))
self.canv.grid(row=0, column=0)

self.scrollY = tk.Scrollbar(self, orient=tk.VERTICAL,
    command=self.canv.yview)
self.scrollY.grid(row=0, column=1, sticky=tk.N+tk.S)

self.scrollX = tk.Scrollbar(self, orient=tk.HORIZONTAL,
    command=self.canv.xview)
self.scrollX.grid(row=1, column=0, sticky=tk.E+tk.W)

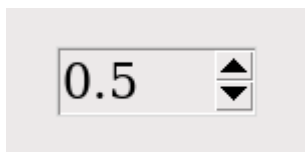
self.canv['xscrollcommand'] = self.scrollX.set
self.canv['yscrollcommand'] = self.scrollY.set
```

备注：

- 连接是双向的。画布上的选项必须连接到水平滚动条上的方法，并滚动条上的选项以连接到画布上的方法。垂直滚动条和帆布必须具有相同的相互连接。xscrollcommand.setcommand.xview
- [.Grid\(\)](#) 方法调用的滚动条上的选项强迫他们伸展足以适合的对应维度的画布。sticky

23. 小部件 Spinbox

小部件允许用户从给定的一组中选择值。这些值可能范围的数字或一组固定的字符串。Spinbox



在屏幕上，面积为显示当前值和一双箭头。Spinbox

- 用户可以单击向上箭头价值推进到下一个较高的价值序列。如果已经是最大值，你可以设置小部件，如果你愿意，所以，新值将环绕在周围的最低值。
- 用户可以单击向下箭头来推进到下一个较低值的值序列中。这支箭可能也配置环绕在周围，因此，如果当前值是最低的点击向下箭头将显示最高值。
- 用户还可以输入值直接，处理小部件，就好像它。用户可以将焦点移到小部件（见[节 53“焦点：路由键盘输入”](#)），或者通过单击它或通过使用 tab 或 shift tab 键，然后编辑显示的值。Entry

若要创建一个新的小部件作为根窗口或框架的孩子：Spinboxparent

```
w = tk.Spinbox(parent, option, ...)
```

构造函数将返回新的小部件。选项包括：Spinbox

表 32。滚轮指定小部件选项

activebackground	背景颜色，当光标位于窗口小部件;请参阅 5.3 节、“颜色” 。
bg 或 background	构件的背景 颜色 。
bd 或 borderwidth	小部件；周围边框的宽度请参阅 第 5.1 条，“维度” 。默认值为一个像素。
buttonbackground	上箭头显示的背景颜色。默认值为灰色。
buttoncursor	光标将显示当鼠标箭头；结束请参阅 一节 5.8，“游标” 。

buttondownrelief	救济式的向下指的箭头;请参见 第 5.6 节、“救济样式” 。默认样式是。tk.RAISED
buttonup	救济风格的向上的箭头;请参见 第 5.6 节、“救济样式” 。默认样式是。tk.RAISED
command	使用此选项来指定一个函数或方法被称为每当用户单击箭头之一。请注意，该回调是当用户直接编辑值，就好像它不会调用。Entry
cursor	选择当鼠标输入部分的部件；上时显示的光标请参见 一节 5.8，“游标” 。
disabledbackground	这些选项选择显示小部件时的前景色和背景颜色。
disabledforeground	statetk.DISABLED
exportselection	通常情况下，可以输入部分中的文本剪切和粘贴。若要禁止这种行为，请将选项设置为。 SpinboxexportselectionTrue
font	使用此选项可以选择不同的字体为输入文本;请参见 5.4 节，“字体” 。
fg 或 foreground	此选项选择用于输入部分的部件，以显示文本的颜色和颜色的箭头。
format	使用此选项来控制结合的数字值的格式设置和选项。例如，将作为十字字符字段，与四个数字的小数点后显示的值。from_toformat=' %10.4f'
from_	使用此选项结合使用选项 （如下所述） 可以限制到一个数值范围内的值。例如，将只允许值介于 1 和 9 包容性之间。请参阅下面的选项。 tofrom_=1to=9increment
highlightbackground	颜色 的重点突出显示时不具有焦点。见 节 53“焦点：路由键盘输入” 。Spinbox
highlightcolor	颜色 的重点突出显示具有 焦点 。Spinbox
highlightthickness	厚度的 关注 热点。默认值为。设置为以禁止显示的关注热点。10
increment	当你约束的值和选项，您可以使用选项来指定多少价值增加或减少当用户点击一个箭头。对于示例中的，选项、 和 ，向上箭头将单步执行值 0.0、 0.5、

	1.0、 1.5、 2.0。 from_toincrementfrom_=0.0to=2.0increment=0.5
insertbackground	选择插入光标在小部件的输入部分中显示的 颜色 。
insertborderwidth	此选项控制插入光标周围边框的宽度。通常情况下，插入光标将有没有边框。如果此选项设置为非零值，则插入光标将显示在 救济方式 。tk.RAISED
insertofftime	这两个选项控制的插入光标闪烁周期： 它的时间花费断断续续地，分别以毫秒为单位。对于示例中的，选项，光标会闪烁了 0.2 秒，然后上的 0.4 秒。 insertofftime=200insertontime=400
insertontime	
insertwidth	使用此选项可以指定的插入光标；宽度可能的值，请参阅 第 5.1 条，“维度” 。默认宽度为两个像素。
justify	此选项控制在小部件的输入部分文本的位置。值可能是左对齐的文本；中心或右对齐的文本。 tk.LEFTtk.CENTERTRIGHT
readonlybackground	此选项指定的背景色将显示小部件时；请参阅 5.3 节、“颜色” 。state' readonly'
relief	使用此选项来选择该构件；救济样式请参见 第 5.6 节、“救济样式” 。默认样式是。tk.SUNKEN
repeatdelay	这些选项指定自动重复行为的鼠标单击上箭头；值以毫秒为单位。值指定多久鼠标按钮必须按住它重复，然后指定函数的重复频率。默认值分别为 400 和 100 毫秒。repeatdelayrepeatinterval
repeatinterval	
selectbackground	使用显示选定的项的背景 颜色 。
selectborderwidth	显示选定项周围的边框宽度。
selectforeground	使用显示选定的项目的前景 颜色 。
state	通常情况下，一个小部件是在状态下创建。设置此选项来作构件响应鼠标或键盘操作。如果你将其设置为，小部件的输入部分中的值不能修改击键次数，但价值仍可被复制到剪贴板上，和小部件仍然响应单击上箭头。Spinboxtk.NORMALtk.DISABLED' readonly'
takefocus	通常情况下，输入部分的一个小部件可以具有焦点（见 节 53“焦点：路由键盘输入” ）。若要从焦点遍历序列中删除小部件，请设置。 Spinboxtakefocus=False

textvariable	如果你想要检索的当前值的小部件，您可以使用下面的方法或你可以关联部件的控制变量通过该控制变量作为此选项的值。见 节 52“控制变量：小部件背后值” 。 <code>.get()</code>
to	此选项指定的值范围的上限。看到该选项，在上面，和也的选项。 <code>from_increment</code>
values	有两种方式来指定小部件的可能值。一种方法是提供字符串的元组作为选项的值。例如，将允许只有那些三个字符串作为值。若要配置小部件来接受范围的数值，请参阅上面的选项。 <code>valuesvalues=('red', 'blue', 'green')</code> <code>from_</code>
width	使用此选项可以指定在小部件的输入部分中允许的字符数。默认值为 20。
wrap	通常情况下，当小部件是在其最高值，向上箭头不执行任何操作，和小部件在其最低值时，向下箭头不执行任何操作。如果你选择向上箭头将提前从最高值到最低，和向下箭头将推进从最低值到最高。 <code>wrap=True</code>
xscrollcommand	使用此选项可以连接到该构件的输入部分的滚动条。有关详细信息，请参阅 节 22.2、“连接到另一个小部件的滚动条” 。

这些方法都可用小部件上:Spinbox

.bbox(*index*)

此方法返回在输入部分的小部件位置处的字符的边界框。结果是一个数组，这些值在哪里和坐标的左上的角，和字符的宽度和高度，以像素为单位，按此顺序。*index(x, y, w, h)xy*

.delete(first, last=None)

这种方法删除字符的输入部分。值的和被解释在 Python 切片的标准方式。`Spinboxfirstlast`

.get()

此方法返回的值。返回的值始终为一个字符串，即使小部件设置为包含数字。`Spinbox`

`.icursor(index)`

使用此方法将插入光标定位在指定，使用标准的 Python 公约为阵地的位置。*index*

`.identify(x, y)`

鉴于该构件内的一个位置，此方法返回一个字符串，描述什么在该位置。值可以是任何的：*(x, y)*

- 'entry' 为在输入区域。
- 'buttonup' 为向上指的箭头。
- 'buttondown' 为向下指的箭头。
- '' (空字符串) 如果这些坐标不在小部件内。

`.index(i)`

此方法返回的数值位置的索引。参数可以是任何的：*i*

- tk.END 要获取该条目的最后一个字符后的位置。
- tk.INSERT 要插入光标的位置。
- tk.ANCHOR 要选择锚点的位置。
- tk.SEL_FIRST' 要获取的所选内容的开始位置。如果所选内容不在小部件内，此方法将引发异常。tk.TclError
- tk.SEL_LAST 让刚刚过去的所选内容的结束位置。如果所选内容不在小部件内，此方法将引发异常。tk.TclError
- 形式的字符串""表示在小部件内的坐标。返回值是字符的包含该坐标的位置。如果坐标一共在构件外部，返回值将接近那个位置的字符的位置。@xx

`.insert(index, text)`

此方法从字符串在所指定的位置插入字符。可能的索引值，请参阅上面的方法。*textindex.index()*

`.invoke(element)`

调用此方法来获取用户单击箭头相同的效果。该参数是向上箭头和向下箭头。*element* 'buttonup' 'buttondown'

`.scan_dragto(x)`

此方法有效[节 10、"条目窗口小部件"](#)中描述的方法相同。`.scan_dragto()`

`.scan_mark(x)`

此方法有效[节 10、“条目窗口小部件”](#)中描述的方法相同。`.scan_mark()`

`.selection('from', index)`

在小部件中所指定的位置设置选择定位点。可能的值，请参阅上面的方法。选择定位点的初始值为 0。`index.index()`

`.selection('to', index)`

选择之间选择锚文本和给定。`index`

`.selection('range', start, end)`

选择之间的文本和指数。允许的索引值，请参阅上面的方法。`startend.index()`

`.selection_clear()`

清除所选内容。

`.selection_get()`

返回选定的文本。如果当前没有选定内容，则此方法将引发异常。`tk.TclError`

24. 小部件 Text

文本小部件是更广义的方法处理多行文本比小部件。文本小部件是非常完整的文本编辑器窗口中：Label

- 你可以用不同的字体、 颜色和背景混合文本。
- 你可以点缀文本嵌入的图像。图像被视为单个字符。请参阅[一节 24.3, “文本小部件图像”](#)。
- [索引](#)是一种描述两个字符之间的文本小部件的特定位置。请参阅[一节 24.1, “文本小部件指数”](#)。
- 一个文本小部件可能包含字符的位置之间的不可见的 *标记*对象。请参阅[一节 24.2, “文本小部件标记”](#)。
- 文本小部件允许您定义的文本称为 *标记*区域的名称。您可以更改标记的区域的外观更改其字体、 前景色和背景色和其他选项。请参阅[一节 24.5, “文本小部件标签”](#)。
- 您可以将事件绑定到某个标记的区域。请参阅[第 54, “事件”](#)。
- 你甚至可以嵌入文本小部件中包含任何 *Tkinter* 小部件“窗口”—— 甚至包含其他小部件框架构件。一个窗口也被视为单个字符。请参阅[24.4, “文本小部件窗口”](#)。

若要创建一个文本小部件作为根窗口或框架命名的孩子：*parent*

```
w = tk.Text(parent, option, ...)
```

构造函数将返回新的小部件。选项包括：Text

表 33. 文本小部件选项

autoseparators	如果设置了此选项，该选项控制还是分隔符会自动添加到撤消堆栈后每个插入或删除 （如果） 不 （如果）。撤消机制的概述，请参阅 一节 24.7, “文本小部件撤消/重做堆栈” 。 undoautoseparatorsautoseparators=Trueautoseparators=False
bg 或 background	默认背景颜色的文本小部件。请参阅 5.3 节, “颜色” 。
bd 或 borderwidth	文本小部件；周围边框的宽度请参阅 第 5.1 条, “维度” 。默认值是两个像素。
cursor	当鼠标在文本小部件会出现光标。请参阅 一节 5.8, “游标” 。

exportselection	通常情况下，在文本控件内的选定文本被出口为窗口管理器中的选定内容。如果你不想要那种行为，设置。 exportselection=0
font	小部件中插入文本的默认字体。请注意，您可以在小部件中的多个字体通过使用标签来改变一些文字的属性。请参见 5.4 节，“字体” 。
fg 或 foreground	用于文本（和位图）在小部件内的 颜色 。您可以更改的颜色标记的区域;此选项是只是默认值。
height	在行（而不是像素！），构件的高度测量根据电流大小。font
highlightbackground	重点突出显示文本小部件不具有焦点时的 颜色 。见 节 53“焦点：路由键盘输入” 。
highlightcolor	重点突出显示文本小部件具有 焦点 时的 颜色 。
highlightthickness	厚度的 关注 热点。默认值为。设置来禁止显示的关注热点。highlightthickness=0
insertbackground	插入光标的 颜色 。默认颜色为黑色。
insertborderwidth	插入光标周围的三维边框的大小。默认值为。0
insertofftime	插入光标时其闪烁周期是关闭的毫秒数。将此选项设置为零，以抑制闪烁。默认值为 300。
insertontime	插入光标是在其闪烁周期中的毫秒数。默认值为 600。
insertwidth	（它的高度由它的线中的最高项）插入光标的宽度。默认值为 2 个像素。
maxundo	此选项设置保留撤消堆栈上的操作的最大数目。撤消机制的概述，请参阅 一节 24.7、“文本小部件撤消/重做堆栈” 。将此选项设置为-1，在撤消堆栈中指定任意的数量的条目。
padx	内部的填充添加到左侧和右侧的文本区域的大小。默认值为一个像素。可能的值，请参阅 第 5.1 条，“维度” 。
pady	添加上面和下面的文本区域的内部边距的大小。默认值为一个像素。
relief	三维外观的文本小部件。默认值是;其他值，请参见 第 5.6 节、“救济样式” 。relief=tk.SUNKEN

selectbackground	使用显示所选的文本的背景 颜色 。
selectborderwidth	使用所选的文本周围的边框宽度。
selectforeground	使用显示选定的文本的前景 颜色 。
spacing1	此选项指定多少额外的垂直空间放上面每行文本。如果一个行换行，这前面添加空格；只有它占有显示的第一行。默认值为。0
spacing2	此选项指定要显示时一个逻辑行换行的文本行之间添加多少额外的垂直空间。默认值为。0
spacing3	此选项指定下面文本的每一行添加多少额外的垂直空间。如果一个行换行，这后面添加空格；只有它占有显示的最后一行。默认值为。0
state	通常情况下，文本小部件响应键盘和鼠标事件；设置要得到这种行为。如果你设置文本小部件不会响应，和你不能以编程方式修改其内容要么。 state=tk.NORMALstate=tk.DISABLED
tabs	此选项控制制表符字符文本的位置。请参阅 节 24.6, “设置选项卡中的文本控件” 。
takefocus	通常情况下，焦点将访问文本小部件（见 节 53“焦点：路由键盘输入” ）。如果你不想在小部件中的焦点，设定。takefocus=0
undo	设置此选项，以启用撤消机制，或禁用它。请参阅 一节 24.7、 “文本小部件撤消/重做堆栈” 。TrueFalse
width	在字符（而不是像素 ！），构件的宽度测量根据当前的字体大小。
wrap	此选项控制太宽的线的显示。 <ul style="list-style-type: none"> • 使用默认的行为，，获取太长会受到破坏，任何字符的任何行。wrap=tk.CHAR • 集和它将断行后将适合的最后一句话。 wrap=tk.WORD • 如果你想要能够创建行太长，放在窗口中，设置和提供一个水平滚动条。wrap=tk.NONE
xscrollcommand	若要使文本小部件水平滚动，水平滚动条的方法设置此选项。.set

`yscrollcommand`

若要使文本小部件垂直滚动，垂直滚动条的方法设置此选项。`.set`

24.1。 小部件指数 Text

索引是指定一个文本小部件中的内容的位置的一般方法。索引是一个与这些形式之一的字符串：

' line. column'

位置就是在给定（计数从零）之前（从一秒数）。例子：
是开始的位置的文本；是第二行第四个字符之前的位置。

columnline' 1. 0' ' 2. 3'

' line. end'

只是之前在给定的行（从一开始计数）末尾换行符的位置。所以，
例如，索引是在第十行的结尾位置。*' 10. end'*

tk. INSERT

在小部件中的文本插入光标的位置。此常数等于字符串。*' insert'*

tk. CURRENT

字符最靠近鼠标指针的位置。此常数等于字符串。*' current'*

tk. END

后文本的最后一个字符位置。此常数等于字符串。*' end'*

tk. SEL_FIRST

如果一些小部件中的文本是当前选择（通过拖动鼠标在它），这是选择开始之前的位置。如果您尝试使用此索引和未选择任何内容，则将引发一个异常。此常数等于字符串。*tk.TclError' sel. first'*

tk. SEL_LAST

在以后的所选内容，如果任何结尾位置。作为与，你会得到一个异常如果你使用这种索引且未选择任何内容。此常数等于字符串。

SEL_FIRSTtk.TclError' sel. last'

' markname'

你可以使用一个标记作为索引；只是将其名称传递指数预计将在那里。
请参阅[一节 24.2, “文本小部件标记”](#)。

`' tag.first'`

标记名称；该区域的第一个字符之前的位置请参阅[一节 24.5, “文本小部件标签”](#)。 *tag*

`' tag.last'`

后一个标记的区域的最后一个字符位置。

`' @x, y'`

接近的坐标 (,) 字符之前的位置。 *xy*

embedded-object

如果你有一个图像或嵌入在文本小部件中的窗口，您可以使用、或嵌入小部件作为索引。请参阅[节 24.3, “文本小部件图像”](#)和[24.4, “文本小部件窗口”](#)。 `PhotoImageBitmapImage`

除了上面的基本索引选项，您可以将任何这些后缀添加到基本索引或索引表达式来生成任意复杂的表达式：

`+ n chars`

从给定索引，移动前的字符。此操作将交叉线。 *n*

例如，假设第一行看起来像这样：

abcdef

索引表达式“”指之间的位置和。你可以省略空白和缩写在这些表达式中的关键字，如果结果是毫不含糊。本示例可以缩写“”。 `1.0 + 5 charsef1.0+5c`

`- n chars`

类似于以前的形式，但位置移动 *向后* 字符。 *n*

`+ n lines`

移动过去的给定索引行。 *Tkinter* 试图离开同一列中的新位置，正如它在行上它离开了，但如果在新位置的线是短的新的位置将在行的结尾。 *n*

`- n lines`

移动前给定索引行。*n*

linestart

移动到给定索引的第一个字符之前的位置。例如，定位””指的是距离鼠标指针最近的行的开头。`current linestart`

lineend

移动到给定索引的最后一个字符后的位置。例如，定位””指的是包含当前所选内容的结束的行结束。`sel.last lineend`

wordstart

开始之前的位置包含给定的索引的单词。例如，索引””指的位置之前在第 11 行上包含的单词定位 44。`11.44 wordstart`

为这次行动的目的，一个字是一串连续的字母、 数字或底线（） 字符，或者是没有这些类型的单个字符。`_`

24.2。 小部件标记 Text

标记表示浮动位置在某个地方在内容的文本小部件。

- 你给它一个名称来处理每个标记。此名称可以是任何字符串不包含空格或句点。
- 有两个特殊标记。 `tk.INSERT` 是插入光标的当前位置，位置靠近鼠标光标。
`tk.CURRENT`
- 浮法标记以及相邻的内容。如果您修改某个地方离标记文本，马克呆在相对于其近邻的同一位置。
- 标记具有属性称为重力控制在一个标记插入文本时，会发生什么。默认重力是，这意味着，当新的文本插入到马克，马克停留后新文本的末尾。如果你设置的一种标志（使用文本小部件的方法） `tk.RIGHT` 重力，马克将住在一个只是之前在那标记新插入的文本的位置。
`tk.LEFT.mark_gravity()`
- 删除所有周围的标记的文本不会删除该标记。如果你想要删除的标记，对文本小部件使用方法。`.mark_unset()`

指节 [24.8](#)， “[文本小部件上的方法](#)”，下面，来看看如何使用标记。

24. 4. 窗口小部件 Text

你可以把任何 *Tkinter* 小部件 —— 甚至包含其他小部件的框架 —— 成一个文本小部件。例如，你可以把一个完全功能的按钮或一组单选按钮成一个文本小部件。

方法用于在文本小部件添加嵌入式的部件。调用顺序和相关的方法，请参阅[二节 24.8， “文本小部件上的方法”](#)。`.window_create()`

24.5。 小部件标签 Text

有很多方法来改变的外观和功能的文本控件中的项。对于文本，您可以更改字体、大小和颜色。此外，你可以使文本，响应键盘或鼠标操作的窗口小部件或嵌入的图像。

要控制这些外观和功能特性，你将每个功能与 *标记* 相关联。然后，您可以将标记关联与任意数量的小部件中的文本块。

- 标记的名称可以是任何字符串不包含空格或句点。
- 还有一个特殊的预定义的标记，称为。这是当前选定的区域，如果有的话。SEL
- 既然任何字符可能是多个标记的一部分，就命令所有的标记 *标记堆栈*。条目的结尾处加上的标记列表中，而且后来的条目优先于以前的项。

因此，例如，如果有的是的两个标记的区域 t_1 和 t_2 ，一部分字符 c t_1 较深，在标记堆栈比 t_2 ， t_1 想要绿色的文本和 t_2 想要蓝色， c 将在呈现蓝色，因为 t_2 t_1 优先。

- 您可以更改标签标记堆栈中的顺序。

标签是使用创建的方法对文本小部件。信息，请参阅[节 24.8， “文本小部件上的方法”](#)，下文对此和相关方法。`.tag_add()`

24.6。 在小部件中设置选项卡 Text

小部件选项使您如何设置制表位在小部件内的数目。tabsText

- 默认值是放置标签每八个字符。
- 要设置特定选项卡位，将此选项设置为序列的一个或多个的距离。例如，设置将从左侧放选项卡停止 3、5 和 12 厘米。过去您设置的最后一个选项卡上，选项卡的最后两个现有的制表位之间的距离作为具有相同的宽度。所以，继续我们的示例中，因为是 7 厘米，如果用户保持按下 Tab 键，光标会在 19 厘米，26 厘米、33 厘米，依此类推。tabs=(' 3c', ' 5c', ' 12c')12c-5c
- 通常情况下后一个制表符，的文本对齐制表位，在其左边缘但你可以包括任何关键字，或在列表中后距离，和这将改变定位文本后的每个选项卡。tk.LEFTtk.RIGHTtk.CENTERTk.NUMERIC
 - 制表位具有默认的行为。tk.LEFT
 - 制表位将放置的文本，因此其右边缘的站。tk.RIGHT
 - 选项卡将居中制表位以下文本。tk.CENTER
 - 制表位将以下文本左侧的文本中的第一期 () 直到停止 — — 在那之后，期间将会围绕停止，和其余文本将位于其右侧。tk.NUMERIC'.'

例如，将设置四个制表位：左对齐的制表位从左侧半英寸、右对齐制表位从左侧 0.8"、居中对齐的制表位从左边，1.2" 和数字对齐制表位从左边 2"。tabs=(' 0.5i', ' 0.8i', tk.RIGHT, ' 1.2i', tk.CENTER, ' 2i', tk.NUMERIC)

24. 7. 构件撤消/重做堆栈 Text

小部件具有一个内置的机制，允许您执行撤消和重做操作，可以取消或恢复对小部件内的文本更改。Text

这里是撤消/重做堆栈是如何工作：

- 每次更改的内容被记录通过推到堆栈上描述的变化，是否插入或删除的条目。这些条目记录内容的旧状态以及新的状态： 如果删除已删除的文本记录;如果插入，插入的文字记录，以及描述的位置和是否插入或删除。
- 您的程序也可能会推动称为 *分离器* 推送到堆栈上的特别记录。
- *撤消* 操作更改为他们在一些以前点小部件的内容。它是通过扭转推送到撤消/重做堆栈上，直到它到达一个分隔符或直到它耗尽堆栈的所有更改。

然而，请注意 *Tkinter* 还记得多少堆栈被推翻在撤消操作，直到一些其他编辑操作将更改小部件的内容。

- *重做* 操作的工作方式只当没有编辑操作发生自上次撤消的操作。它将重新应用所有的撤消的操作。

用于实现撤消/重做堆栈的方法，请参阅、[、](#) 和方法在[部分 24. 8, “文本小部件上的方法”](#)。撤消机制未启用默认情况下;你必须在小部件中设置此选项。`.edit_redo`.`edit_reset`.`edit_separator`.`edit_undoundo`

24. 8. 小部件上的方法 Text

这些方法都可用所有文本小部件上：

.bbox(*index*)

返回给定索引，一个 4 元组处的字符的边界框。如果看不到的字符，则返回。请注意此方法可能不返回精确的值，除非您调用该方法（见[第 26，“通用构件方法”](#)）。(*x*, *y*, *width*, *height*)None.update_idletasks()

.compare(*index1*, *op*, *index2*)

比较两个指数的位置在文本小部件，并返回 true 如果之间的关系持有和。指定要使用，其中一个什么比较:、或。

*op*index1index2op'<'<='=='!='>'>'

例如，为一个文本小部件，返回 true，如果第二行开始之前或在文本末尾。tt.compare(' 2.0', '<=', END)t

.delete(*index1*, *index2*=None)

删除文本只是之后开始。如果省略第二个参数，则只有一个字符被删除。如果给出第二个索引，删除收益达，但不是包括之后，的字符。记得指数坐之间的字符。*index1index2*

.dlineinfo(*index*)

返回为包含的行的边框给定。形式的定界框和谨慎空闲任务更新，请参阅上述方法的定义。*index*.bbox

.edit_modified(*arg*=None)

查询、设置，或清除 *修改国旗*。使用此标志来跟踪是否发生了小部件的内容。例如，如果您要在一个小部件实现文本编辑器，您可能使用修改后的标志来确定是否内容发生了变化，因为你上次将内容保存到一个文件。Text

当不带参数调用，此方法返回是否已设置修改后的标志，如果它是明确。您还明确可以通过将值传递给此方法，设置修改后的标志或删除它通过传递一个值。TrueFalseTrueFalse

插入或删除文本，是否是由程序操作或用户操作的任何操作或撤消或重做操作，将修改后的标志。

`.edit_redo()`

执行恢复操作。有关详细信息，请参阅[节 24.7、“文本小部件撤消/重做堆栈”](#)。

`.edit_reset()`

将清除撤消堆栈。

`.edit_separator()`

撤消堆栈上推一个分隔符。这种分离器限制了未来的撤消操作，包括只推因为分离器被推的变化范围。有关详细信息，请参阅[节 24.7、“文本小部件撤消/重做堆栈”](#)。

`.edit_undo()`

如果堆栈包含没有分隔符取得自从最后的分隔符被迫撤消堆栈上的小部件的内容或到堆栈的底部反转的所有更改。有关详细信息，请参阅[节 24.7、“文本小部件撤消/重做堆栈”](#)。如果在撤消堆栈为空，它是个错误。

`.image_create(index[, option=value, ...])`

此方法将一个图像插入部件。图像被视为只是另一个字符，其大小是图像的自然大小。

这种方法的选项显示在下表中。您可以通过一系列的参数或选项名称和值的字典。*option=value*

align	此选项指定图像是如何垂直对齐，如果它的高度小于其包含的行的高度。值可能使它在顶部的空间；中心 将其放置在底部；或线对齐文本基线与图像的底部。topcenterbottombaseline
image	要使用的图像。请参阅 一节 5.9、“图像” 。
name	你可以将名称分配给此实例的图像。如果您省略此选项， Tkinter 将生成一个唯一的名称。如果你在同一小部件中创建图像的多个实例， Tkinter 将生成唯一的名称，通过附加“”跟着一个数字。Text#
padx	如果提供，此选项是多余的空间图像的两侧添加的像素的数目。
pady	如果提供，此选项是多余的空间，要添加上面和下面的图像的像素的数目。

`.get(index1, index2=None)`

使用此方法从该构件检索当前文本。检索索引处开始。如果省略第二个参数，你得到的字符。如果你提供第二个索引，你得到那些两个指标之间的文本。嵌入的图像和 windows（小部件）将被忽略。如果区域中包含多行，它们是用换行符（`\n`）字符分隔。

index1index1' \n'

`.image_cget(index, option)`

若要检索嵌入式图像上设置的选项的当前值，指向图像和选项的名称索引调用此方法。

`.image_configure(index, option, ...)`

若要设置一个或多个选项对嵌入的图像，与指向图像作为第一个参数和一个或更多对索引调用此方法。*option=value*

如果您不指定任何选项，您将重新获得一本字典定义上的图像和相应的值的所有选项。

`.image_names()`

此方法返回一个元组的名称的所有文本小部件的嵌入的图像。

`.index(i)`

对于索引，此方法返回在窗体中的等效位置。*i' line.char'*

`.insert(index, text, tags=None)`

插入给定在给定。*textindex*

如果省略该参数，新插入的文本将与适用于字符 *都* 之前和之后插入点的任何标记标记。*tags*

如果你想要将一个或多个标记应用于您要插入的文本，提供作为第三个参数标记字符串的元组。适用于现有字符的插入点附近的任何标记将被忽略。*注：* 第三个参数必须是一个元组。如果您提供一个列表变量，*Tkinter* 默默地将无法应用标签。如果你提供一个字符串，每个字符将被视为一个标记。

`.mark_gravity(mark, gravity=None)`

更改或查询的现有标记；重力 [节 24.2, “文本小部件标记”](#)，以上说明，请参阅的重力。

设置重力，传入的名称标记，要么跟着或。找到现有标记的重力，忽略第二个参数和方法返回或。tk.LEFTtk.RIGHTtk.LEFTtk.RIGHT

.mark_names()

在窗口中，返回一个序列的所有标记的名称包括和。

tk.INSERTtk.CURRENT

.mark_next(*index*)

返回的名称标记以下给出;如果有没有以下的标记，该方法将返回一个空字符串。*index*

如果是数值形式，该方法返回的第一个标记的该位置。如果是一个标记，该方法将返回该标记，可能是在相同的数值位置之后的下一个标记。*indexindex*

.mark_previous(*index*)

返回标记前面的名称给定。如果没有前面的标记，该方法将返回一个空字符串。*index*

如果是数值的形式，该方法将返回返回最后一个标记的该位置。如果是一个标记，该方法将返回前面的标记，可能是在相同的数值位置。*indexindex*

.mark_set(*mark*, *index*)

如果没有标记的名称存在，一与重力创建并放置在哪里点。如果标记已经存在，它被移动到新位置。*mark*tk.RIGHT *index*

此方法可能会更改的位置或索引。tk.INSERTtk.CURRENT

.mark_unset(*mark*)

删除已命名的标记。此方法不能用于删除或标记。

tk.INSERTtk.CURRENT

.scan_dragto(*x*, *y*)

请参阅下面。*.scan_mark*

.scan_mark(*x*, *y*)

此方法用于实现快速滚动的小部件。通常情况下，用户按下和鼠标按钮举行一些位置的小部件，并在所需方向上，然后移动鼠标和小部件

移动，鼠标的距离成正比的速度方向已经因为按钮被按下。这项议案可能垂直或水平滚动的任意组合。Text

要实现此功能，将鼠标按下事件绑定到处理程序调用，在哪里和是当前鼠标位置。然后将该事件绑定到处理程序调用，在哪里，都是新的鼠标位置。`.scan_mark(x, y)xy<Motion>.scan_dragto(x, y)xy`

`.search(pattern, index, option, ...)`

搜索为（可以是一个字符串或正则表达式）起价缓冲区中给定。如果成功，它返回一个索引的形式;如果它失败了，它返回一个空字符串。`patternindex' line.char'`

此方法允许选项有：

backwards	将此选项设置为向后搜索从索引。默认值为批转。True
count	如果一场比赛时到控制变量，设置此选项您可以检索的文本的长度匹配法对该变量在方法返回后。IntVar.get()
exact	设置此选项来搜索完全匹配的文本。这是默认选项。比较下面的选项。Truepatternregexp
forwards	设置此选项来搜索从索引批转。这是默认选项。True
regexp	设置此选项去解释作为 Tc1 样式的正则表达式。默认设置是寻找到精确匹配。Tc1 的正则表达式是 Python 的正则表达式，支持这些功能的子集：Truepatternpattern. ^ [c ₁ ...] (...) * + ? e ₁ e ₂
nocase	将此选项设置为 1 来忽略大小写。默认值是区分大小写的搜索。
stopindex	将搜索限制到索引之外的搜索不应该设置此选项。

`.see(index)`

如果包含给定的索引的文本是不可见的直到那文字是可见的滚动文本。

`.tag_add(tagName, index1, index2=None)`

此方法将关联以区域的内容索引之后只是开始，并延伸到索引命名的标记。如果省略，仅随其后的字符标记。

`tagNameindex1index2index2index1`

`.tag_bind(tagName, sequence, func, add=None)`

此方法将事件绑定到用标记的所有文本。信息，请参阅[第 54、“事件”](#)，下面，更多关于事件绑定。*tagName*

若要创建新绑定的标记文本，使用三个参数： 标识的事件，和是你想要它，当事件发生时调用的函数。*sequencefunc*

若要添加另一个绑定到现有的标记，通过相同的前三个参数和第四个参数。*'+'*

要找出哪些绑定存在标记上的给定序列，通过只有两个参数;该方法返回相关联的函数。

若要查找所有的绑定为一个给定的标记，通过只有第一个参数;该方法返回所有标记的参数的列表。*sequence*

.tag_cget(*tagName*, *option*)

使用此方法来检索值的给定为给定。*option**tagName*

.tag_config(*tagName*, *option*, ...)

若要更改的选项为命名的标记值，传递一个或多个成对。*tagNameoption=value*

如果你通过只使用一个参数，您将重新获得一本字典，目前在力命名标记中定义的所有选项和它们的值。

这里是为标记配置选项：

background	与此标记文本的背景颜色。请注意，您不能使用的简称。bg
bgstipple	要使背景显得灰暗，请将此选项设置为标准的位图名称（见 第 5.7 节，“位图” ）之一。这没有任何影响，除非您还指定。background
borderwidth	与此标记文本周围边框的宽度。默认值为。请注意，您不能使用的简称。0bd
fgstipple	若要使文本显示浅灰色，请设置此选项的 位图名称 。
font	用来显示与此标记文本的字体。请参见 5.4 节，“字体” 。
foreground	用于与此标记文本的颜色。请注意这里不能使用缩写。fg

justify	在每一行的第一个字符上设置的选项可确定这条线如何有道理：（默认值），或。 justifylefttk.LEFTtk.CENTERtk.RIGHT
lmargin1	多少有此标记的文本块的第一行的缩进。默认值为。允许的值，请参阅 第 5.1 条，“维度” 。0
lmargin2	多少行连续的有此标记的文本块的缩进。默认值为。0
offset	多少对提升（正值）或降低（负值）文本与此标记相对于基线。使用它来得到上标或下标，例如。允许的值，请参阅 第 5.1 条，“维度” 。
overstrike	设置为通过带有此标记的文本中心画一条水平线。 overstrike=1
relief	用于与此标记文本的三维效果。默认值是；其他可能的值请参见 第 5.6 节、“救济样式” 。relief=tk.FLAT
rmargin	带有此标记的文本块的右边距的 大小 。默认值为。0
spacing1	此选项指定多少额外的垂直 空间 放上面带有此标记文本的每一行。如果一个行换行，这前面添加空格；只有它占有显示的第一行。默认值为。0
spacing2	此选项指定当一个逻辑行换行之间添加多少额外的垂直空间显示带有此标记的文本行。默认值为。0
spacing3	此选项指定下面带有此标记文本的每一行添加多少额外的垂直空间。如果一个行换行，这后面添加空格；只有它占有显示的最后一行。默认值为。0
tabs	如何与此标记线上展开选项卡。请参阅 节 24.6，“设置选项卡中的文本控件” 。
underline	设置为与此标记的文本添加下划线。underline=1
wrap	多长时间在此标记的文本换行。见上文描述的选项的文本小部件。wrap

.tag_delete(*tagName*, ...)

若要删除一个或多个标签，将他们的名字传递给此方法。他们的选择和绑定远去，而从各区域的文本中删除标签。

.tag_lower(*tagName*, *belowThis*=None)

使用此方法以更改标记的标记中的顺序堆叠（[节 24.5，“文本小部件标签”](#)，以上说明，请参阅标记堆栈的）。如果你传递两个参数，具有

名称的标记移动到位置下方的标记名称。如果你通过只使用一个参数，该标记移到标记堆栈的底部。 *tagNamebelowThis*

.tag_names(*index=None*)

如果你通过 *index* 参数，则此方法返回后该索引是与字符关联的所有标记名称的序列。如果你不传递任何参数，你得到一个序列在文本小部件中定义的所有标记名称。

.tag_nextrange(*tagName, index1, index2=None*)

此方法搜索某一区域的地方标记名为开始的地方。区域搜索的索引处开始和结束索引处。如果省略该参数，则搜索去一路文本的末尾。

tagNameindex1index2index2

如果有一个地方在给定的区域，该标记的开始位置，该方法返回一个序列，其中是第一次标记的字符的索引，是位置的索引，只是后最后一个标记字符。 [*i0, i1*] *i0i1*

若没有标记开始被发现在该地区，该方法返回一个空字符串。

.tag_prevrange(*tagName, index1, index2=None*)

此方法搜索某一区域的地方标记名为开始的地方。搜索该区域开始之前索引和结束索引处。如果省略该参数，则搜索去一路文本的末尾。

tagNameindex1index2index2

返回值都是一样。 *.tag_nextrange()*

.tag_raise(*tagName, aboveThis=None*)

使用此方法以更改标记的标记中的顺序堆叠（[节 24.5, “文本小部件标签”](#)，以上说明，请参阅标记堆栈的）。如果你传递两个参数，具有名称的标记移动到位置上方的标记名称。如果你通过只使用一个参数，该标记移到标记堆栈的顶部。 *tagNameaboveThis*

.tag_ranges(*tagName*)

此方法在小部件中，标记名称，发现所有的文本的范围，并返回一个序列，每个只是之前的范围的第一个字符的索引，只是后范围的最后一个字符是索引。 *tagName[s₀, e₀, s₁, e₁, ...] s_ie_i*

.tag_remove(*tagName, index1, index2=None*)

删除标签之间的所有字符从命名为和。如果省略，则将从单个字符后删除该标记。 *tagNameindex1index2index2index1*

`.tag_unbind(tagName, sequence, funcid=None)`

移除的事件绑定给出了从命名的标记。如果有多个处理程序，此序列和标记，您可以通过将它作为第三个参数传递删除只有一个处理程序。*sequence**tagName*

`.window_cget(index, option)`

返回给定的选项为嵌入式构件在给定 *index*.

`.window_configure(index, option)`

若要更改嵌入式构件在给定，传递选项的值中的一个或更多对。
*index**option*=*value*

如果你通过只使用一个参数，您将重新获得一本字典，目前在给定部件的力量中定义的所有选项和它们的值。

`.window_create(index, option, ...)`

此方法创建一个窗口，在那里一个小部件可以嵌入文本小部件。有两种方式来提供嵌入式的构件：

- a. 您可以使用传递给在此方法中，选项的小部件或 window
- b. 你可以定义一个过程创建小部件并将这一程序作为回调传递到选项。*create*

选项有：`.window_create()`

align	指定如何定位嵌入式的构件垂直在其行，如果它不是一样高的行上的文本。值包括：（默认值），其中心线；内垂直构件哪些地方图像的顶部顶部的线;哪些地方的图像底部底部的线;，， 对齐文本基线与图像的底部。 align=tk.CENTERalign=tk.TOPalign=tk.BOTTOMalign=tk.BAS SELINE
create	将在需求创建嵌入式小部件的过程。此过程不带参数必须作为一个孩子的文本小部件创建窗口小部件和作为结果返回小部件。
padx	添加到左边和右边的构件内的文本行的额外 空间 。默认值为。0
pady	添加的上方和下方的文本行内构件的额外空间。默认值为。0

stretch	此选项控制高于嵌入式构件线时，会发生什么。通常此选项，则意味着该嵌入式的构件留在其自然大小。如果你设置这个小部件垂直拉伸以填充行的高度和忽略该选项。 0stretch=1align
window	要嵌入的构件。这个小部件必须是子文本小部件。

.window_names()

返回包含所有嵌入的小部件名称序列。

.xview(tk.MOVETO, *fraction*)

此方法水平，滚动文本小部件，并用于绑定到相关的水平滚动条的命令选项。

可以以两种不同的方式调用此方法。第一次调用中的文本放置在给定的在哪里 0.0 将文本移动到其最左端的位置和 1.0 到其最右边的位置值。*fraction*

.xview(tk.SCROLL, *n*, *what*)

第二次调用将文本移左或向右： 参数指定到多少移动，可以是或，并告诉多少个字符或页来将文本移到右相对于其图像（或左，如果为负数）。*what*tk.UNITStk.PAGES*n*

.xview_moveto(*fraction*)

此方法将文本滚动方式相同。**.xview(tk.MOVETO, *fraction*)**

.xview_scroll(*n*, *what*)

相同。**.xview(tk.SCROLL, *n*, *what*)**

.yview(tk.MOVETO, *fraction*)

垂直滚动等效。**.xview(tk.MOVETO, ...)**

.yview(tk.SCROLL, *n*, *what*)

垂直滚动等效。当垂直滚动的单位为行。**.xview(tk.SCROLL, ...)**tk.UNITS

.yview_moveto(*fraction*)

垂直滚动等效。**.xview_moveto()**

```
.yview_scroll(n, what)
```

垂直滚动等效。`.xview_scroll()`

25. : 顶级窗口方法 Toplevel

一个 *顶级窗口* 是一个独立存在的根据，窗口管理器的窗口。它与窗口管理器的装饰，装饰和能够移动和调整大小独立。您的应用程序可以使用任意数量的顶级窗口。

任何小部件，您可以向其顶级小部件使用。 `www.wininfo_toplevel()`

若要创建新的顶级窗口：

```
w = tk.Toplevel(option, ...)
```

选项包括：

表 34。顶级窗口方法

bg 或 background	窗口的背景颜色。请参阅 5.3 节、“颜色” 。
bd 或 borderwidth	边框的宽度，以像素为单位;默认值为。可能的值，请参阅 第 5.1 条，“维度” 。请参见下面的选项。 0relief
class_	你可以给一个窗口“类”的名称。这种名称对照选项数据库中，所以您的应用程序可以接用户的配置首选项（如颜色）按类名。例如，您可能会设计一系列的弹出窗口称为“尖叫，”并设置大势已去。然后你可以把一条线在这样的数据库选项:，然后，如果你使用该方法读取选项数据库，所有的部件，用那类名称将默认为红色背景。此选项命名是因为是在 Python 中的保留的字。Toplevelclass_='Screamer'
	*Screamer*background: red
	.option_readfile()class_class
cursor	当鼠标在该窗口中显示的光标。请参阅 一节 5.8, “游标” 。
height	窗口的高度;请参阅 第 5.1 条，“维度” 。
highlightbackground	重点突出显示在窗口不具有焦点时的 颜色 。见 节 53”焦点：路由键盘输入” 。
highlightcolor	重点突出显示窗口具有 焦点 时的 颜色 。

highlightthickness	厚度的 关注 热点。默认值为。设置来禁止显示的关注热点。lhighlightthickness=0
menu	此窗口提供顶级菜单栏，供应小部件作为此选项的值。根据 MacOS，此菜单将显示在屏幕的顶部，当窗口处于活动状态。在 Windows 或 Unix 下，它将出现在应用程序的顶部。Menu
padx	使用此选项可以提供额外的空间上的左、 右两边的窗口。值的像素数。
pady	使用此选项可以在窗口的顶部和底部两侧提供额外的空间。值的像素数。
relief	通常情况下，一个顶级窗口会在它附近没有 3 d 边界。若要获取带阴影的边框，请将选项设置为较大，其默认值为零，并集选项常量之一在下讨论 第 5.6 节、“救济样式” 。bdrelief
takefocus	通常情况下，一个顶级窗口不能集中精力。如果你希望它能采取重点;，使用见 节 53“焦点： 路由键盘输入” 。takefocus=True
width	所需的窗口的宽度;请参阅 第 5.1 条，“维度” 。

这些方法都可用于顶级窗口：

`.aspect(nmin, dmin, nmax, dmax)`

约束范围根窗口宽度： 长度比率 $[/, /]$ 。 $n_{min}d_{min}n_{max}d_{max}$

`.deiconify()`

如果此窗口图标化，以展开它。

`.geometry(newGeometry=None)`

设置窗口几何。论点的形式，请参阅[一节 5.10，“几何字符串”](#)。如果省略该参数，则返回当前几何字符串。

`.iconify()`

图标化的窗口。

`.lift(aboveThis=None)`

为了提高此窗口到堆叠顺序中的窗口管理器的顶部，请调用此方法不带任何参数。可以也将提高它到另一个窗口上方的堆叠顺序中的位置将作为参数传递该窗口。Toplevel

`.lower(belowThis=None)`

如果省略该参数，则移动到堆叠顺序中的窗口管理器的底部的窗口。也可以通过该窗口部件作为参数将窗口移动到一个位置只是在一些其他的顶级窗口之下。Toplevel

`.maxsize(width=None, height=None)`

设置的最大窗口大小。如果省略参数，返回当前。(width, height)

`.minsize(width=None, height=None)`

设置的最小窗口大小。如果省略参数，返回当前的极小值为 2 元。

`.overrideredirect(flag=None)`

如果用一个参数调用，此方法设置重写重定向旗子，去除所有的窗口管理器装饰从窗口中，因此它不能被移动、调整大小、图标化，或关闭。如果用一个参数调用，则还原窗口管理器装饰和清除该重写重定向标记。如果不带参数调用，它返回重写重定向标志的当前状态。
TrueFalse

一定要调用的方法（见[第 26, “通用构件方法”](#)）之前设置此标志。如果你叫它进入主循环之前，您的窗口将被禁用，它曾经出现之前。`.update_idletasks()`

这种方法可能无法工作在某些 Unix 和 MacOS 的平台。

`.resizable(width=None, height=None)`

如果是 true，允许水平调整大小。如果是 true，允许垂直调整大小。如果省略参数，作为一个 2 元组返回的当前大小。height

`.state(newstate=None)`

返回窗口的当前状态之一：

- 'normal'：正常显示。
- 'iconic'：图标化的方法。`.iconify()`
- 'withdrawn'：隐藏;请参阅下面的方法。`.withdraw()`

若要更改窗口的状态，上面字符串中的一个作为参数传递给该方法。
例如，若要图标化一个实例，请使用“”。
`Toplevel.TT.state('iconify')`

`.title(text=None)`

设置窗口标题。如果省略该参数，则返回当前标题。

`.transient(parent=None)`

使此窗口瞬态的窗口，对一些窗口；默认父窗口是此窗口的父。*parent*

此方法很有用短命的弹出对话框窗口。在其父面前总是出现一个瞬变的窗口。如果图标化的父窗口，瞬态是以及图标化。

`.withdraw()`

隐藏窗口。将其与还原或。`.deiconify().iconify()`

26. 普遍小部件的方法

方法对所有小部件定义如下。在说明中，可以是任何类型的任何部件。*w*

w.after(*delay_ms*, callback=None, *args)

请 *Tkinter* 至少延迟后调用带有参数的函数毫秒。还有多久它实际上会，没有上限，但您的回调不会比你的请求，更早叫和它将调用只有一次。callbackargsdelay_ms

此方法返回一个整数“之后的标识符”，如果你想要取消回调可以传递给该方法。*.after_cancel()*

如果不传递的参数，该方法等待的毫秒，在[标准的 Python 时间模块](#)的功能。callbackdelay_ms.sleep()

w.after_cancel(*id*)

取消对回调成立较早的请求。该参数是由原始的*.after()*调用返回的结果。*.after()* *id*

w.after_idle(func, *args)

Tkinter 调用函数参数与下一次系统的请求是闲置的那就是下，一次没有要处理的事件。将只有一次调用回调。如果您希望您的回调再次调用，您必须再次调用方法。*funcargs.after_idle*

w.bell()

发出的噪音，通常发出蜂鸣音。

w.bind(sequence=None, func=None, add=None)

此方法用于将事件绑定附加到一个小部件。事件绑定的概述，请参阅[一节 54、“事件”](#)。

参数描述了我们所期望的哪些事件和参数是一个函数时向该构件发生该事件时调用。如果有的话，已经为这个小部件为该事件绑定通常替换旧的回调，但是您可以通过保留两个回调。

sequencefuncfuncadd=' +'

w.bind_all(sequence=None, func=None, add=None)

像，但适用于整个应用程序中的所有小部件。*.bind()*

`w.bind_class(className, sequence=None, func=None, add=None)`

像，但适用于所有小部件命名（例如，`w.bind('Button')`）。

`w.bindtags(tagList=None)`

如果调用此方法时，它将作为一个字符串序列构件返回“绑定标签”。一个绑定标记是窗口的名称（入手`'.'`）或类的名称（例如，`'Listbox'`）。

您可以更改调用的顺序结合水平是通过作为参数的绑定标签序列您想要使用的构件。

绑定到标签的水平和他们的关系的论述，请参见[第 54、“事件”](#)。

`w.cget(option)`

作为一个字符串返回的当前值。你也可以为小部件作为选项的值。`w[option]`

`w.clipboard_append(text)`

将给定的字符串追加到显示剪贴板，剪切和粘贴字符串为所有显示的应用程序的存储位置。`text`

`w.clipboard_clear()`

清除显示剪贴板（见上文）。`w.clipboard_append()`

`w.column_configure()`

看到[4.2 节，“其他网格管理方法”](#)。

`w.config(option=value, ...)`

相同。`w.configure()`

`w.configure(option=value, ...)`

设置一个或多个选项的值。他们的名字是选项为 Python 的保留字（`in`，`is`），使用尾随的底线：`__from__`，`__in__`，`__is__`，`__in__`，`__is__`。

您还可以设置构件的发言的选项的值 `w`

`w[option] = value`

如果你对一个小程序不带任何参数调用该方法，你会得到一本字典的所有部件的当前选项。键是选项名称（包括像的别名）。每个键的值是：`.config()bdbborderwidth`

- 对于大多数项目，5 元组：（选项名称，选择数据库密钥，选项数据库类，默认值，当前值）；或，
- 别名名称（如），两个元组：（别名名称，等效的标准名称）。'fg'

`w.destroy()`

一个小程序呼吁破坏及其所有子级。`w.destroy()` *ww*

`w.event_add(virtual, *sequences)`

此方法创建其名称由字符串参数给出的虚拟事件。每个额外的参数描述了一个序列，即物理事件的说明。当该事件发生时，新的虚拟事件触发。*virtual*

一般的虚拟事件说明，请参阅[节 54、“事件”](#)。

`w.event_delete(virtual, *sequences)`

从其名称由字符串给出的虚拟事件中删除物理事件。如果从某一特定的虚拟事件中移除所有的物理事件，那虚拟的事件不会再发生。*virtual*

`w.event_generate(sequence, **kw)`

此方法导致没有任何外部的刺激触发事件。仿佛它已经被触发受到外界的刺激，将事件的处理都是相同的。参数描述了事件被触发。可以提供参数，在对象中设置选定字段的值在对象中指定字段的名称。*sequence*`Eventkeyword=valuekeywordEvent`

对事件的全面讨论，请参阅[一节 54、“事件”](#)。

`w.event_info(virtual=None)`

如果您调用此方法没有参数，你会回来所有当前定义的虚拟事件名称的序列。

要检索与虚拟的事件关联的物理事件，传递给这个方法的虚拟的事件的名称，您将重新获得一个序列的物理名称，或者如果给定虚拟事件永远不会被定义。*sequence*`None`

`w.focus_displayof()`

返回当前已经有输入焦点同一屏幕上显示小部件作为窗口的名称。如果没有这种窗口已经有输入焦点，返回。None

见[节 53“焦点：路由键盘输入”](#)输入焦点的一般说明。

`w.focus_force()`

部队向该构件的输入焦点。这是不礼貌的。它是更好地等待，窗口管理器，让你的焦点。请参阅下文。`.grab_set_global()`

`w.focus_get()`

返回此应用程序中具有焦点的小部件，如果有的话 — — 否则将返回。None

`w.focus_lastfor()`

此方法检索，最后将输入的焦点在包含的顶级窗口小部件的名称。如果没有这顶级的小部件曾经有输入的焦点，它将返回顶级小部件的名称。如果此应用程序不具有输入的焦点，将返回它回来到此应用程序的下次将获得焦点的小部件的名称。`w.focus_lastfor()`

`w.focus_set()`

如果应用程序具有输入的焦点，焦点将跳转到。如果应用程序不具有焦点，Tk 会记得要给它下一个应用程序获得焦点。`www`

`w.grab_current()`

如果有一场争夺战中的力量的显示，返回其标识符，否则为返回。有关详细信息，请参阅[一节 54](#)、“事件”抓斗的讨论。`wNone`

`w.grab_release()`

如果有一场争夺战中的力，释放它。`w`

`w.grab_set()`

小部件抓住所有事件的应用程序。如果有另一个抓现行，它就消失了。抓斗的讨论，见[第 54、“事件”](#)。`ww`

`w.grab_set_global()`

小部件抓住整个屏幕的所有事件。这被认为是不礼貌的行为，应该只用于急需。任何其他抓取的力就会消失。尝试使用这种令人敬畏的力量只为好，力量和从未为邪恶势力的好吗？`w`

`w.grab_status()`

如果有本地抓力（由设置）中的，此方法返回的字符串。如果有是全局的抓力（从）中，它将返回。如果没有抓斗是在部队，它将返回。`.grab_set()'local'.grab_set_global()'global'None`

`w.grid_forget()`

看到 [4.2 节，“其他网格管理方法”](#)。

`w.grid_propagate()`

看到 [4.2 节，“其他网格管理方法”](#)。

`w.grid_remove()`

看到 [4.2 节，“其他网格管理方法”](#)。

`w.image_names()`

返回的名称的所有图像中的应用程序作为一个序列的字符串。`w`

`w.keys()`

返回一个字符串序列作为小部件的选项名称。

`w.lift(aboveThis=None)`

如果参数是，窗口包含移动到窗口堆叠顺序的顶部。只是一些窗口上方移动窗口，将作为参数传递。`None wToplevel ww`

`w.lower(belowThis=None)`

如果参数是，窗口包含移动到窗口堆叠顺序的底部。只是一些窗口下方移动窗口，将作为参数传递。`None wToplevel ww`

`w.mainloop()`

一般的静态的小部件创建，开始处理事件之后，必须调用此方法。你可以离开主回路的方法（见下文）。你也可以调用此方法来恢复主循环所需的事件处理程序内。`.quit()`

`w.nametowidget(name)`

此方法返回其路径名称是实际构件。请参阅[一节 5.11、“窗口名称”](#)。如果是未知的此方法将引发。`namenameKeyError`

`w.option_add(pattern, value, priority=None)`

此方法添加到 *Tkinter* 选项数据库选项的默认值。是一个字符串，指定的一个或多个部件的选项的默认值。人的价值是：
patternvaluepriority

20	对全局默认属性的部件。
40	对特定应用程序的默认属性。
60	对于来自用户文件如其文件的选项。 <code>.Xdefaults</code>
80	后应用程序设置的选项启动。这是默认优先级别。

更高级别的优先次序优先于较低级别的。选择数据库的概述，请参阅 [第 27 款“规范外观”](#)。参数的语法是线的相同的资源规格的一部分。
pattern.option_add() option-pattern

例如，要得到这个资源规格的线的效果：

```
*Button*font: times 24 bold
```

您的应用程序（在本例中）可能会包含这些行：`self`

```
self.bigFont = tkFont.Font(family='times', size=24,
                           weight='bold')
self.option_add('*Button*font', self.bigFont)
```

任何小部件创建后执行这些线会默认为次 24 粗体（除非由选项给构造函数 `ButtonfontButton` 的重写）。

`w.option_clear()`

此方法从 *Tkinter* 选项数据库中删除所有选项。这就相当于回到了所有的默认值。

`w.option_get(name, classname)`

使用此方法从 *Tkinter* 选项数据库检索选项的当前值。第一个参数是实例键，第二个参数是类键。如果有任何匹配项，则返回最佳匹配的选项的值。如果没有匹配项，它将返回。''

请参阅[“规范外观”部分 27](#)更多关于如何将键匹配的选项。

`w.option_readfile(fileName, priority=None)`

为方便用户配置，您可以指定一个命名的文件，用户可以把他们首选的选项，使用相同的格式的文件。然后，当您的应用程序正在初始化，可以将该文件的名称传递给此方法，并从该文件选项将添加到数据库。如果该文件不存在，或其格式无效，此方法将引发。`.Xdefaultstk.TclError`

有关详细信息，请参阅对[第 27 款“规范外观”](#)导论选项数据库文件和格式选项。

`w.register(function)`

此方法创建一个 Python 函数，Tcl 的包装和 Tcl 包装名称作为字符串返回。使用此方法的示例，请参见[章节 10.2、“条目窗口小部件添加验证”](#)。

`w.quit()`

这种方法退出主循环。请参阅上文，主回路的讨论。`.mainloop()`

`w.rowconfigure()`

看到[4.2 节，“其他网格管理方法”](#)。

`w.selection_clear()`

如果选择（如条目窗口小部件中的文本突出显示部分），目前已清除此选择。`w`

`w.selection_get()`

如果目前有的选择，此方法返回选定的文本。如果没有选择，它将引发 `wtk.TclError`

`w.selection_own()`

使所选内容的所有者在已显示，偷它以前的主人，如果有的话。`ww`

`w.selection_own_get()`

返回窗口小部件，目前拥有中的选定内容的显示。如果没有这种选择，则引发。`wtk.TclError`

`w.tk_focusFollowsMouse()`

通常情况下，输入的焦点循环通过一系列的小部件由其层次结构和创作的秩序；见[节 53“焦点：路由键盘输入”](#)。相反，你可以告诉

Tkinter 强制将焦点放到鼠标在哪里;只是调用此方法。没有简单的方法,以撤消它,然而。

`w.tk_focusNext()`

返回如下部件焦点遍历序列中。请参阅[节 53“焦点：路由键盘输入”](#)讨论焦点遍历。*w*

`w.tk_focusPrev()`

在焦点遍历序列返回前面的小部件。*w*

`w.unbind(sequence, func!=None)`

此方法删除绑定上所描述的事件。如果第二个参数是一个回调绑定到该序列,删除该回调和休息,如果有,将保留在原处。如果省略第二个参数,则删除所有绑定。*wsequence*

请参阅[第 54、“事件”](#),下面,事件绑定了一般性讨论。

`w.unbind_all(sequence)`

删除整个应用程序所描述的事件的所有事件绑定给定。*sequence*

`w.unbind_class(className, sequence)`

像,但适用于所有小部件命名(例如,或)。`.unbind(className'Entry' 'Listbox'`

`w.update()`

此方法强制更新显示。应该使用的是只有当你知道你正在做,因为它会导致不可预知的行为或循环。它永远不应该从事件回调或从一个事件回调函数调用的函数调用。

`w.update_idletasks()`

在更新显示,调整大小和重绘窗口小部件,比如一些任务被称为*空闲任务*,因为它们通常会延期,直到应用程序已经完成处理事件,并已经走回主回路,等待新的事件。

如果你想要强制显示之前应用程序接下来懈怠,必须更新,在任何部件上调用方法。`w.update_idletasks()`

`w.wait_variable(v)`

一直等待，直到设置了变量的值，即使该值不会更改。这种方法进入一个本地等待循环，所以它不会阻止应用程序的其余部分。 *v*

w.wait_visibility(w)

等到小部件（通常）是可见的。 *wToplevel*

w.wait_window(w)

等待，直到窗口销毁。 *w*

w.winfo_children()

返回列表中的所有儿童，从最低（底部）到最高的（顶部）其堆叠顺序中。 *w*

w.winfo_class()

返回的类名称（例如，）。 *w* 'Button'

w.winfo_containing(rootX, rootY, displayof=0)

此方法用于查找包含点（，）的窗口。如果选项是虚假的坐标是相对于应用程序的根窗口；如果为 true，坐标为相对于包含的顶级窗口处理。如果指定的点是在应用程序的顶级窗口之一，此方法返回该窗口；否则它将返回。 *rootXrootYdisplayofwNone*

w.winfo_depth()

返回数的每个像素的位数的显示。 *w*

w.winfo_fpixels(number)

对于任何维度（见[第 5.1 条](#)，“[维度](#)”），此方法返回距离以像素为单位上的显示，作为类的数目。 *numberwfloat*

w.winfo_geometry()

返回描述大小的几何字符串和屏幕上的位置。请参阅[节 5.10](#)，“[几何弦](#)”。 *w*

警告

几何不是准确的直到应用程序已更新其空闲任务。尤其是，所有的几何图形最初是直到小部件和几何经理谈判达成其大小和位置。请参阅方法，以上，在这一节中看到如何保证是最新的小部件的几何形状。`'1x1+0+0'.update_idletasks()`

`w.winfo_height()`

返回以像素为单位的当前高度。请参见备注上几何更新下，以上。你可能更愿意使用，如下所述，这是始终与时俱进。

`w.winfo_geometry().winfo_reqheight()`

`w.winfo_id()`

返回一个整数，用于唯一标识在其顶级窗口内。你会需要这个方法，下面。`w.winfo_pathname()`

`w.winfo_ismapped()`

此方法返回 `true` 如果否则是映射、虚假。如果已经过了网格映射一个小部件（或放置或包装，如果您使用其他几何经理之一）到其父，如果其父映射，等等到顶级窗口。`w`

`w.winfo_manager()`

如果没有网格（或通过其他几何经理之一放置），此方法返回一个空字符串。如果网格或已另有放置，它返回一个字符串命名的几何管理器：此值将是、、、或。

`www'grid''pack''place''canvas''text'`

`w.winfo_name()`

此方法返回的名称相对于其父。请参阅[一节 5.11、“窗口名称”](#)。也请参阅，下面，找出如何获取一个部件的路径名称。

`w.winfo_pathname()`

`w.winfo_parent()`

返回的父路径名称或空字符串如果是一个顶级窗口。更多关于小部件路径名称，请参阅[“窗口名称”一节 5.11](#)以上。`ww`

`w.winfo_pathname(id, displayof=0)`

如果该参数为 `false`，应用程序的主窗口中返回窗口路径名称的小工具来使用的唯一标识符。如果是 `true`，数量在同一顶级窗口中以指定一个小部件。小部件路径名称的讨论，见[第 5.11、“窗口名称”](#)。

`displayofiddisplayofidw`

`w.wininfo_pixels(number)`

对于任何维度（见上文的尺寸，），此方法返回距离以像素为单位上的显示，作为一个整数。*numberw*

`w.wininfo_pointerx()`

返回由返回的坐标相同的值。*x.wininfo_pointerxy()*

`w.wininfo_pointerxy()`

返回一个元组包含相对于鼠标指针的坐标的根窗口。如果鼠标指针不在同一个屏幕上，返回。*(x, y)w(-1, -1)*

`w.wininfo_pointery()`

返回由返回的坐标相同的值。*y.wininfo_pointerxy()*

`w.wininfo_reqheight()`

这些方法返回小部件所需的高度。这是最起码的必要，所有的内容的高度有他们需要的房间。实际高度可能不同与几何经理进行谈判。*ww*

`w.wininfo_reqwidth()`

返回请求的宽度的小部件，包含所需的最小宽度。
像[.wininfo_reqheight\(\)](#)，实际宽度可能有不同与几何经理进行谈判。
ww

`w.wininfo_rgb(color)`

对于任何给定的颜色，此方法返回等效的红-绿-蓝颜色规格作为一个 3 元组，其中每个数字是整数范围 [0, 65536)。例如，如果是，此方法返回 3 元组。*(r, g, b)color'green'(0, 65535, 0)*

有关指定颜色的详细信息，请参阅 [5.3 节、“颜色”](#)。

`w.wininfo_rootx()`

返回的左手边的坐标是相对于根窗口的父。*xww*

如果有一个边框，这是外缘的边界。*w*

`w.wininfo_rooty()`

返回的顶边的坐标是相对于根窗口的父。*yww*

如果有一个边框，这是边框的上边缘。*w*

`w.wininfo_screenheight()`

以像素为单位返回屏幕的高度。

`w.wininfo_screenmmheight()`

以毫米为单位返回屏幕的高度。

`w.wininfo_screenmmwidth()`

以毫米为单位返回屏幕的宽度。

`w.wininfo_screenvisual()`

返回一个字符串，描述的色彩显示的方法。这通常是 16 位或 24 位显示，256 色的显示器。'truecolor' 'pseudocolor'

`w.wininfo_screenwidth()`

返回屏幕的宽度，以像素为单位。

`w.wininfo_toplevel()`

返回含有的顶级窗口。该窗口小部件；上支持的所有方法见[节 25“顶层： 顶级窗口方法”](#)。*wToplevel*

`w.wininfo_viewable()`

返回一个值，如果谓词是可视的即，如果它和它的所有祖先在同一映射。*True wToplevel*

`w.wininfo_width()`

返回以像素为单位的当前宽度。请参见备注上几何更新下，以上。您可能更愿意使用上述方法，；它始终是最新的。

`w.wininfo_geometry().wininfo_reqwidth()`

`w.wininfo_x()`

返回相对于其父的左侧的坐标。如果有一个边框，这是外缘的边界。

XWW

`w.wininfo_y()`

返回相对于其父的顶边的坐标。如果有一个边框，这是外缘的边界。

yww

27. 外观和选项数据库的标准化

它很容易创建它们时适用小部件的颜色、字体和其他选项。然而，

- 如果你想要很多部件都有相同的背景色或字体，它是单调乏味，指定每个选项，每次和
- 它是好，让用户重写你的选择与他们最喜欢的配色方案、字体和其他的选择。

因此，我们使用这个*选项数据库*设置默认选项值。

- 您的应用程序可以指定一个包含用户的首选项（如使用 X 窗口系统的标准文件）文件。您可以设置您的应用程序读取该文件并告诉 *Tkinter* 使用这些默认设置。请参阅[.option readfile\(\)](#)方法中，一节以上，部分[节 26、“通用构件方法”](#)，这个文件的结构中。`.Xdefaults`
- 您的应用程序可以直接指定一个或很多类型的小部件的默认值，通过使用[.option add\(\)](#)方法;请参阅[节 26、“通用构件方法”](#)这个方法。

我们讨论如何设置选项之前，请考虑自定义图形用户界面的外观一般的问题。我们可以给每个应用程序中的构件名称，然后问用户指定每个属性的每个名称。但这是很麻烦，也会使应用程序难以重新配置——如果设计器中添加新的小部件，用户要说明每个属性的每个新的小部件。

所以，选择数据库允许程序员和用户指定*的一般模式*描述哪些部件配置。

这些模式操作的名称的窗口小部件，但部件则使用*两个并行命名模式*命名：

- a. 每个小部件具有一个类名。默认情况下，类名是类的构造函数相同：对于按钮，为一个框架，等等。不过，你可以创建新类的小部件，通常继承类，并给他们你自己创造的新名称。见[节 27.1“如何命名窗口小部件类”](#)的详细信息。`'Button' 'Frame' Frame`
- b. 你也可以给任何小部件的实例名称。小部件的默认名称通常是没有意义的数字（见[节 5.11，“窗口名称”](#)）。然而，随着窗口小部件类，可以将名称分配给任何部件。请参阅一节[节 27.2“如何小部件实例的名字”](#)的详细信息。

在每个应用程序中每个构件，因此具有两个层次结构的名称——的类名称层次结构和实例名称层次结构。例如，是本身嵌入到框架中的文本控件中嵌入的按钮会有类层次结构。它也可能有实例层次结构类似如果你如此命名的所有实例。初始点代表根窗口;在窗口路径名称的更多信息，请参见[节 5.11、“窗口名称”](#)。`Frame.Text.Button.mainFrame.messageText.panicButton`

选项数据库机制可使用任一类名或实例中定义的选项的名称，这样你就可以选择适用于整个类（例如，所有按钮都具有蓝色背景）或特定实例（例如，恐慌按钮有红色字母上它）。我们看看如何命名类和实例，在[节 27.3、“资源规格线”](#)之后，我们将讨论如何选择数据库真的工作。

27.1。 如何命名窗口小部件类

例如，假设这就是您已经创建了新窗口小部件类。它可能是最好要有新的窗口小部件类从类继承的所以到 *Tkinter*，它像一个框架，您可以安排其他小部件，如标签、条目和在它里面的按钮。JukeboxFrame

您通过将名称作为选项传递给父构造函数在您新的类构造函数中设置新的小部件类名称。这里是代码的定义新类片段：class_

```
class Jukebox(tk.Frame):
    def __init__(self, master):
        '''Constructor for the Jukebox class
        ...

        tk.Frame.__init__(self, master, class_='Jukebox')
        self.__createWidgets()
        ...
```

27. 2. 如何命名部件实例

给应用程序中的特定部件实例名称，请将该窗口部件的选项设置为一个包含名称的字符串。name

这里是一个例子的实例名称。假设您要在应用程序中，创建几个按钮，你想要一个按钮有的实例名称。你对构造函数的调用可能如下所示：panicButton

```
self.panic = tk.Button(self, name='panicButton',  
text='Panic', ...)
```

27.3. 资源规格线

每个选项文件中的行在一个或多个应用程序中指定的值的一个或多个选项和这些格式之一：

```
app option-pattern: value
option-pattern: value
```

第一个窗体设置的选项仅当应用程序的名称匹配；第二种形式设置所有应用程序的选项。*app*

例如，如果您的应用程序被称为 *xparrot*，窗体的线

```
xparrot*background: LimeGreen
```

在 *xparrot* 应用程序中以石灰绿色设置的所有选项。（使用选项在命令行上的启动您的应用程序时设置的名称。）`background-name'xparrot'`

一部分具有这种语法：*option-pattern*

```
{{*|.} name}... option
```

也就是说，每个是零个或多个名称列表，每一种前面带有星号或时期。系列中的最后一个名称是选项的您要设置的名称。每个其余的名称可以是：*option-pattern*

- （大写），窗口小部件类的名称或
- （小写）实例的名称。

选择模式的工作的方式是有点复杂。让我们从一个简单的例子开始：

```
*font: times 24
```

这条线说：所有选项应都默认为 24 点时间。称为松散绑定符号，并且意味着此选项模式适用于任何地方在任何应用程序中的任何选项。比较此示例：

```
font*font
```

```
*Listbox.font: lucidatypewriter 14
```

期间并称为紧束缚的象征，这也意味着，此规则只适用于在类中的小部件的选项。`ListboxfontfontListbox`

另一个例子，假设您的 *xparrot* 应用程序有的窗口小部件类的实例。为了设置该类的所有小部件的默认背景色，你可以放一条线你选项文件中像这样：

JukeboxJukebox

```
xparrot*Jukebox*background: PapayaWhip
```

松散绑定 `()` 符号之间，使这一规则适用于任何选项内任何地方的任何部件。比较此选项行：`*JukeboxbackgroundbackgroundJukebox`

```
xparrot*Jukebox.background: NavajoWhite
```

此规则将适用于框架构成部件本身，但由于紧束缚象征它将不适用于在里面小部件的部件。JukeboxJukebox

在下一节我们会谈论 *Tkinter* 出确切地选择要使用的值如果有多个资源适用的规范行的数字。

27. 4. 资源匹配规则

当你正在创建一个小部件，您不要指定一些选项的值和两个或更多的资源规格适用于选项，是最具体适用。

例如，假设您选项文件具有以下两行：

```
*background: LimeGreen
*Listbox*background: FloralWhite
```

两个规范适用于选项的小部件，但第二个是更具体，所以它会赢。

backgroundListbox

一般情况下，资源规格中的名称是序列 n_1, n_2, n_3, \dots ， o 每个 n_i 哪里类或实例的名称。类名称进行排序从最高到最低的水平 and o 是选项的名称。

然而，当 *Tkinter* 创建一个小部件，它的一切是类名和实例名该窗口部件。

资源规格的优先级规则是这样的：

1. 选项的名称必须匹配的 o 部分。例如，如果规则是 *option-pattern*

2. xparrot*indicatoron: 0

这将匹配只有命名的选项。indicatoron

3. 紧束缚运算符 $()$ 是比松散绑定运算符 $()$ 更具体。例如，一条线为是比一条线为更具体。
.*Button.font*Button.font
4. 对实例的引用更具体比对类的引用。例如，如果您有其实例名称是一个按钮，规则是比规则更加具体。
panicButton*panicButton*font*Button*font
5. 更多级别的规则是更具体。例如，规则是比规则更加具体。
*Button*font*font
6. 如果两个规则具有相同的级别数，前面列表中的名称是比后来的名字更具体。例如，规则是比规则更加具体。
xparrot*font*Button*font

28. *ttk*：主题窗口小部件

开始与 Tk 8.5，*ttk* 模块变得可用。本模块替换多（但不是全部）的原始 *Tkinter* 机械。使用本模块可以获得这些优势：

- *特定于平台的外观*。在 Tk 8.5 之前的版本中，Tk 应用程序的最常见的抱怨之一就是它们不符合各种平台的样式。

Ttk 模块使您可以编写您的应用程序以常规的方式，但您的应用程序可以看起来像一个 Windows 应用程序根据 Windows，像苹果 Mac app 下 MacOS，等等，没有对您的程序的任何更改。

每个可能的不同外观是由一个命名的 *ttk* 主题表示的。例如，主题给你上一节中所述的原始 *Tkinter* 小部件的外观。classic

- *简化和国家特定构件行为的泛化*。在基本的 *Tkinter* 世界中，有大量的小部件选项指定小部件应该如何看或根据各种条件的行为。

例如，小部件具有几个不同的选项控制前景（文本）颜色。

`tk.Button`

- 当光标在该按钮上时应用颜色的选项。`activeforeground`
- 当插件被禁用时，将使用颜色。`disabledforeground`
- 小部件会有的颜色，当其他条件不适用。`foreground`

Ttk 模块折叠成一个简单的两个部分系统的大量的这些特殊情况：

- 每个小部件具有不同的国家，数目和每个国家可以打开或关闭独立于其他。国家的例子有：`:`、`:` 和 `:`。`disabledactivefocus`
- 您可以设置指定的 *样式地图* 某些选项将设置为特定值取决于一些国家或一些小部件的状态的组合。

若要使用 *ttk*，你需要知道这些事情。

- [节 28.1，"导入 *ttk*"](#)：设置您的程序使用 *ttk*。
- [节 28.2、"*ttk* 小部件集"](#)：新和替换 *ttk* 小部件。
- [节 47，"自定义和创建 *ttk* 主题和样式"](#)。

28.1. 进口 *ttk*

有不同的方法来导入 *ttk* 模块。

- 如果你喜欢所有的小部件和其他特点 *Tkinter* *ttk* 处于你的全局命名空间，使用这种形式的导入：

- `from Tkinter import *`
- `from ttk import *`

重要的是做这两个进口按这顺序，所以从 *ttk* 的所有小部件类型替换从 *Tkinter* 等效的小部件。例如，您的小部件将来自 *ttk* 和不 *Tkinter*。
`Button`

- 在更复杂的应用程序，在您使用多个导入的模块，它可以大大提高代码的可读性如果你练习 *安全命名空间卫生*：导入所有您使用的模块””语法。这就需要有更多的打字，但它有巨大的优点，你可以看着提到某件事，告诉它来自何处。`import modulename`

我们建议这种形式的导入：

```
import ttk
```

所以在此导入之后，是小部件构造函数，是一个，和等等。
`ttk.LabelLabelttk.ButtonButton`

如果你需要从 *Tkinter* 模块引用的项，它是作为可用。例如，“东北”的锚点代码是。`ttk.Tkinter``ttk.Tkinter.NE`

你可能会改为导入 *Tkinter* 分别以这种方式：

```
import Tkinter as tk
```

在这种形式的导入之后，“东北”的代码是。`tk.NE`

28. 2. *ttk* 小部件集

Ttk 模块包含不同版本的大多数标准 *Tkinter* 窗口小部件和几个新的。这些小部件取代从 *Tkinter* 的相同的名称：

- [节 29, “*ttk*。按钮”](#)。
- [节 30, “*ttk*。Checkbutton”](#)。
- [节 32, “*ttk*。进入”](#)。
- [节 33, “*ttk*。框架”](#)。
- [节 34, “*ttk*。标签”](#)。
- [节 35, “*ttk*。LabelFrame”](#)。
- [节 36, “*ttk*。上方”](#)。
- [节 38, “*ttk*。PanedWindow”](#)。
- [节 40, “*ttk*。单选按钮”](#)。
- [节 41, “*ttk*。规模”](#)。
- [节 42, “*ttk*。滚动条”](#)。

这些小部件是新的和特定 *ttk*:

- [节 31, “*ttk*。组合框”](#)。
- [节 37, “*ttk*。笔记本”](#)。
- [节 39, “*ttk*。进度”](#)。
- [节 43, “*ttk*。分离器”](#)。

28. 2. *ttk* 小部件集

Ttk 模块包含不同版本的大多数标准 *Tkinter* 窗口小部件和几个新的。这些小部件取代从 *Tkinter* 的相同的名称：

- [节 29, “*ttk*。按钮”。](#)
- [节 30, “*ttk*。Checkbutton”。](#)
- [节 32, “*ttk*。进入”。](#)
- [节 33, “*ttk*。框架”。](#)
- [节 34, “*ttk*。标签”。](#)
- [节 35, “*ttk*。LabelFrame”。](#)
- [节 36, “*ttk*。上方”。](#)
- [节 38, “*ttk*。PanedWindow”。](#)
- [节 40, “*ttk*。单选按钮”。](#)
- [节 41, “*ttk*。规模”。](#)
- [节 42, “*ttk*。滚动条”。](#)

这些小部件是新的和特定 *ttk*:

- [节 31, “*ttk*。组合框”。](#)
- [节 37, “*ttk*。笔记本”。](#)
- [节 39, “*ttk*。进度”。](#)
- [节 43, “*ttk*。分离器”。](#)

30. *ttk*. Checkbutton

这个小部件是 *ttk* 版本的[部分 9](#)，[“Checkbutton 小部件”](#)。打造一个 *ttk* 小部件作为给定的控件的子级：`.Checkbuttonparent`

```
w = ttk.Checkbutton(parent, option=value, ...)
```

这里是 *ttk* 小部件的选项。比较这些到 *Tkinter* 版本讨论在[第 7](#)，[“按钮部件”](#)。`.Checkbutton`

表 37. *ttk*. Checkbutton 选项

class_	小部件类的名称。这可能指定当小部件创建，但以后不能更改。窗口小部件类的说明，请参阅 第 27 款“规范外观” 。
command	这个 checkbutton 的状态发生更改时调用的函数。
compound	此选项指定图像相对于文本的相对位置，当您同时指定。值可能是（图像文本的上方），（图像文本下方），（左边的文本图像），或（图像右侧的文本）。如果你提供了和选项但不指定一个值，仅该图像将显示。 tk. TOPtk. BOTTOMtk. LEFTtk. RIGHTimagetextcompound
cursor	光标将显示当鼠标位于 checkbutton; 请参阅 一节 5.8, “游标” 。
image	图像出现在 checkbutton; 请参阅 一节 5.9、“图像” 。
offvalue	默认情况下，当 checkbutton 处于关闭（未选中）状态下，相关联的值为 0。您可以使用选项来指定一个不同的值，为关闭状态。variableoffvalue
onvalue	默认情况下，当 checkbutton 在上（检查）的状态相关联的值为 1。你可以使用选项来指定一个不同的值，为打开状态。variableonvalue
style	要呈现此 checkbutton; 所用的样式见 节 49“使用和自定义 <i>ttk</i> 样式” 。
takefocus	默认情况下， <i>ttk</i> 将列入焦点遍历; 见 节 53“焦点：路由键盘输入” 。要从焦点遍历删除小部件，请使用。 <code>.Checkbuttontakefocus=False</code>
text	要显示在 checkbutton，作为一个字符串的文本。

textvariable	控制在 <code>checkboxbutton</code> ; 显示的文本变量见 节 52“控制变量：小部件背后值” 。
underline	如果此选项有一个非负的值 n ，下划线将显示下面的字符在位置 n 。 <code>text</code>
variable	控制变量的跟踪的当前状态的 <code>checkboxbutton</code> ; 见 节 52“控制变量：小部件背后值” 。通常，您将使用这里和关闭和值分别是 0 和 1。然而，你可能会使用不同的控制变量的类型，并指定选项使用的类型的值。 <code>IntVar</code> <code>offvalue</code> <code>onvalue</code>
width	<p>使用此选项可以指定固定的宽度或最小宽度。以字符为单位; 指定的值一个积极的值设置的固定的宽度的许多平均字符，而消极的宽度设置最小宽度。</p> <p>例如，如果选定字体中的平均字符是 10 像素宽，选项会使文本标签到底 80 像素宽; 选项将使用 80 像素或文本的长度，以较大者为准。<code>width=8</code><code>width=-8</code></p> <p>也可能在关联的样式中指定一个值。如果样式和小部件构造函数调用中指定值，前者优先。<code>width</code></p>

这些选项 *Tkinter* 构件由 *ttk* 部件构造函数 不受支持：
`Checkbox`. `Checkbox`

表 38。 *Tkinter* 不在 *ttk* 选项。 `Checkbox`

activebackground	使用样式映射来控制选项; 见 节 50.2“ttk 样式映射：动态外观变化” 。 <code>background</code>
activeforeground	使用样式映射来控制选项。 <code>foreground</code>
anchor	<p>配置此选项使用的样式; 见节 49“使用和自定义 ttk 样式”。使用此选项来指定文本的位置，当选项分配的额外水平空间。<code>width</code></p> <p>例如，如果您指定的选项和 <code>checkboxbutton</code>，显示这两个文本和图像和 （东），指定的图像将在右端的二十个字符空间，它只是下方的文本样式。 <code>width=20</code><code>compound=tk.TOP</code><code>anchor=tk.E</code></p> <p>当 <code>checkboxbutton</code> 显示图像，但没有文字时，此选项将被忽略。</p>

background 或 bg	配置使用样式的选项。不支持缩写。backgroundbg
bitmap	不受支持。
borderwidth 或 bd	配置此选项使用的样式。
disabledforeground	使用样式地图选项;见 节 50.2“<i>ttk</i> 样式映射： 动态外观变化” 。foreground
font	配置此选项使用的样式。
foreground 或 fg	配置此选项使用的样式。
height	不受支持。
highlightbackground	若要控制重点突出显示的颜色，当 <code>checkboxbutton</code> 不具有焦点时，请使用样式映射来控制选项;见 节 50.2“<i>ttk</i> 样式映射： 动态外观变化” 。 highlightcolor
highlightcolor	您可以指定默认焦点突出显示颜色，通过在样式中设置此选项。您也可以控制使用样式映射的重点突出显示颜色。
highlightthickness	配置此选项使用的样式。此选项不能在所有的主题。
indicatoron	不受支持。
justify	控制如何多行彼此之间的相对水平放置。配置此选项使用的样式;可能的值，或为左对齐、 居中或右对齐，分别。tk.LEFTtk.CENTERtk.RIGHT
offrelief	不受支持。
overrelief	使用样式映射来控制选项;见 节 50.2“<i>ttk</i> 样式映射： 动态外观变化” 。relief
padx	不受支持。
pady	不受支持。
relief	使用样式映射来控制选项;见 节 50.2“<i>ttk</i> 样式映射： 动态外观变化” 。relief
selectcolor	不受支持。
selectimage	不受支持。
state	在 <i>ttk</i> ，没有具有该名称的选项。被广义状态机制;见 节 50.2“<i>ttk</i> 样式映射： 动态外观变化” 。

wraplength	如果您使用具有此选项设置为一些 维度 ，将样式被切成片不再比该维度。 <code>text</code>
------------	--

在 *ttk* 方法包括所有所述[部分 46，“所有 *ttk* 部件的共同方法”](#)，再加上
`:.Checkbutton`

`.invoke()`

此方法将 `checkbutton` 的状态切换。如果回调，它调用该回调，并返回回调返回任何值。`command`

不支持是 *Tkinter* 小部件的以下方法：`configure()`、`configurestate()` 和 `configuretext()`。若要更改状态的 `checkbutton` 通过程序控制，请使用关联控件的方法。

`Checkbutton.deselect().flash().select().toggle().set(variable)`

31. *ttk*. Combobox

这个小部件是和下拉菜单中的组合。在应用程序中，您将看到常见的文本输入区域，与向下的箭头。当用户点击箭头时，会出现一个下拉菜单。如果用户点击其中一个，这种选择将替换当前条目的内容。不过，用户仍可能直接进入入口（当它具有[焦点](#)）时，键入的文本或编辑现有的案文。Entry

打造一个 *ttk* 小部件作为给定的控件的子级：*.Comboboxparent*

```
w = ttk.Combobox(parent, option=value, ...)
```

选项：

表 39. *ttk*. 组合框选项

class_	小部件类的名称。这可能指定当小部件创建，但以后不能更改。窗口小部件类的说明，请参阅 第 27 款“规范外观” 。
cursor	光标将显示当鼠标位于 checkbutton;请参阅 一节 5.8, “游标” 。
exportselection	默认情况下，如果您选择文本内的部件，它是自动导出到剪贴板中。若要避免此出口，使用。 Entryexportselection=0
height	使用此选项可以指定将显示的行的最大数目在下拉的菜单;默认值为 20。如果有更多比这一数字，下拉菜单中将自动包括一个垂直滚动条。values
justify	此选项指定如何将会放置文本在输入区域内当它未完全填充的区域。值可能要左对齐；中心；或右对齐。 tk. LEFTtk. CENTERtk. RIGHT
postcommand	可以使用此选项来提供用户单击向下箭头时将调用的回调函数。此回调可能更改的选项;如果是这样，所做的更改将出现在下拉菜单中。values
style	要呈现此 checkbutton; 所用的样式见 节 49“使用和自定义 <i>ttk</i> 样式” 。
takefocus	默认情况下， <i>ttk</i> 将列入焦点遍历;见 节 53“焦点：路由键盘输入” 。要从焦点遍历删除小部件，请使用。 .Checkbuttontakefocus=False

textvariable	控制输入区域；显示的文本变量见 节 52“控制变量：小部件背后值” 。
validate	可以使用此选项可以请求动态验证的 widget 的文本内容。请参阅 章节 10.2、“条目窗口小部件添加验证” 。
validatecommand	可以使用此选项来指定一个回调函数，动态验证部件的文本内容。请参阅 章节 10.2、“条目窗口小部件添加验证” 。
values	在下拉的菜单中，将显示为一个字符串序列的选择。
width	此选项指定在输入区域的宽度作为字符的数目。实际宽度将此数倍的平均宽度的有效字体中的字符。默认值为 20。
xscrollcommand	如果部件有关联的水平滚动条，设置此选项的那个滚动条的方法。 <code>.set</code>

在 *ttk* 方法包括所有那些 *Tkinter* 部件[部分 10、“条目窗口小部件”](#)，所述上所述[部分 46，“所有 *ttk* 部件的共同方法”](#)，再加上所有的方法加上：`.Combobox`

`.current([index])`

给此方法，要选择的选项的元素之一，作为参数传递该元素的索引。如果你不提供参数，返回的值是文本的当前条目中列表或-1 索引如果当前输入文本不在列表中。`valuesvaluesvalues`

`.set(value)`

在小部件中设置的当前文本。*value*

Ttk 构件的国家会影响其操作。要审问或更改状态，请参阅和[中 46，“方法共同所有 *ttk* 部件”](#)的方法。`.Combobox.instate().state()`

- 小部件的状态是，如果没有用户操作将更改的内容。`disabled`
- 如果构件是在状态和也状态，用户可使用下拉式菜单，更改内容，但不是能直接编辑内容。`!disabledreadonly`

32. *ttk*.Entry

小部件的目的是允许用户输入或编辑单行文本。这是[一节 10](#)、窗口小部件”*ttk* 版本。Entry

打造一个 *ttk* 小部件作为给定的控件的子级：*.Entryparent*

```
w = ttk.Entry(parent, option=value, ...)
```

选项:

表 40. *ttk*. 进入选项

class_	小部件类的名称。这可能指定当小部件创建，但以后不能更改。窗口小部件类的说明，请参阅 第 27 款“规范外观” 。
cursor	光标将显示当鼠标位于 <code>checkbutton</code> ; 请参阅 一节 5.8, “游标” 。
exportselection	默认情况下，如果您选择文本内的部件，它是自动导出到剪贴板中。若要避免此出口，使用。 <code>Entryexportselection=0</code>
font	使用此选项可以指定将显示的文本的字体的小部件; 请参见 5.4 节, “字体” 。作者不清楚的原因，不能与样式指定此选项。
invalidcommand	你可能将此选项设置为验证失败时将调用的回调函数（也就是说，当返回一个 0）。请参阅 章节 10.2、 “条目窗口小部件添加验证” 。 <code>validatecommand</code>
justify	此选项指定如何将会放置文本在输入区域内当它未完全填充的区域。值可能要左对齐; 中心; 或右对齐。 <code>tk.LEFTtk.CENTERtk.RIGHT</code>
show	为了防止如密码字段在屏幕上可见，请设置此选项为一个字符串，其第一个字符将替换为每个字段中的实际字符。例如，如果该字段包含”但你已指定，该字段将显示为””。 <code>sesameshow='*'*****</code>
style	要呈现此 <code>checkbutton</code> ; 所用的样式见 节 49“使用和自定义 <i>ttk</i> 样式” 。

takefocus	默认情况下， <i>ttk</i> 将列入焦点遍历;见 节 53“焦点： 路由键盘输入” 。要从焦点遍历删除小部件，请使用。 <code>.Checkbuttontakefocus=False</code>
textvariable	控制输入区域；显示的文本变量见 节 52“控制变量： 小部件背后值” 。
validate	可以使用此选项来指定一个回调函数，动态验证部件的文本内容。请参阅 章节 10.2、“条目窗口小部件添加验证” 。
validatecommand	请参阅 章节 10.2、“条目窗口小部件添加验证” 。
width	此选项指定在输入区域的宽度作为字符的数目。实际宽度将此数倍的平均宽度的有效字体中的字符。默认值为 20。
xscrollcommand	如果部件有关联的水平滚动条，设置此选项的那个滚动条的方法。 <code>.set</code>

这些选项 *Tkinter* 构件由 *ttk* 部件构造函数 不受支持：Entry.Entry

表 41。 *Tkinter* 不在 *ttk* 选项。进入

background 或 bg	配置选项使用样式;见 第 47 节，“自定义和创建 <i>ttk</i> 主题和样式” 。不支持缩写。backgroundbg
borderwidth 或 bd	配置此选项使用的样式。
disabledbackground	使用样式地图选项;见 节 50.2“<i>ttk</i> 样式映射： 动态外观变化” 。background
disabledforeground	使用样式地图选项;见 节 50.2“<i>ttk</i> 样式映射： 动态外观变化” 。foreground
foreground 或 fg	配置此选项使用的样式。
highlightbackground	若要控制重点突出显示的颜色，当 <code>checkboxbutton</code> 不具有焦点时，请使用样式映射来控制选项;见 节 50.2“<i>ttk</i> 样式映射： 动态外观变化” 。 highlightcolor
highlightcolor	您可以指定默认焦点突出显示颜色，通过在样式中设置此选项。您也可以控制使用样式映射的重点突出显示颜色。

highlightthickness	配置此选项使用的样式。此选项不能在所有的主题。
insertbackground	不受支持。
insertborderwidth	不受支持。
insertofftime	不受支持。
insertontime	不受支持。
insertwidth	不受支持。
readonlybackground	使用样式映射来控制选项;见 节 50.2”<i>ttk</i> 样式映射：动态外观变化 ”。background
relief	配置此选项使用的样式;见 第 47 节，“自定义和创建 <i>ttk</i> 主题和样式 ”。
selectbackground	使用样式映射来控制选项;见 节 50.2”<i>ttk</i> 样式映射：动态外观变化 ”。background
selectborderwidth	使用样式映射来控制选项;见 节 50.2”<i>ttk</i> 样式映射：动态外观变化 ”。borderwidth
selectforeground	使用样式映射来控制选项;见 节 50.2”<i>ttk</i> 样式映射：动态外观变化 ”。foreground

在 *ttk* 的方法包括所有那些[节 10、“条目窗口小部件”](#)中所述的部件 *Tkinter* 上所述[部分 46，“所有 *ttk* 部件的共同方法](#)”，再加上所有的方法。`.Entry`

33. *ttk*.Frame

像这种 *Tkinter* widget，*ttk* 构件是其他小部件的矩形容器。若要创建一个子部件作为给定的控件的子级：`Frame.FrameFrameparent`

```
w = ttk.Frame(parent, option=value, ...)
```

选项包括：

表 42. *ttk*. 框架选项

borderwidth	使用此选项来指定元素的宽度边界;默认值为零。
class_	当您创建此小部件时，您可以提供一个小部件类名称。该名称可用于自定义小部件的外观;请参阅 第 27 款“规范外观” 。一旦创建了该构件，构件类名称不能更改。
cursor	使用此选项可以指定鼠标光标的外观，当它结束的部件;请参阅 一节 5.8, “游标” 。默认值（空字符串）指定光标从父组件继承。
height	此选项是框架的设置高度 尺寸 。如果你想要迫使框架有一个特定的高度，叫上小部件;看到 4.2 节, “其他网格管理方法” 。 <code>.grid_propagate(0)</code>
padding	若要创建框架内外包含小部件的空白区域，对所需的 尺寸 设置此选项。例如，将清除在框架内和周围的窗口小部件里面外面半英寸宽区域。 <code>padding='0.5i'</code>
relief	指定的救济样式的边框;请参见 第 5.6 节、“救济样式” 。这没有任何影响，除非你也增加。 <code>borderwidth</code>
style	使用此选项可以指定自定义小部件样式名称;见 第 47 节, “自定义和创建 <i>ttk</i> 主题和样式” 。
takefocus	使用此选项可以指定是否一个小部件访问期间焦点遍历;见 节 53“焦点：路由键盘输入” 。指定是否你想要去接受焦点;指定是否该小部件是不接受焦点。默认值是一个空的字符串;默认情况下， <i>ttk</i> 部件做不能集中精力。 <code>takefocus=True</code> <code>takefocus=False</code> .Frame
width	此选项是设置框架的宽度 尺寸 。如果你想要迫使帧有特定的宽度，叫上小部件;看到 4.2 节, “其他网格管理方法” 。 <code>.grid_propagate(0)</code>

这些选项 *Tkinter* 部件上没有可用的选项在 *ttk* 构造函数：Frame.Frame

表 43。 *Tkinter* 不在 *ttk* 选项。框架

background 或 bg	配置此选项具有风格;见 第 47 节，“自定义和创建 <i>ttk</i> 主题和样式” 。
highlightbackground	要控制重点突出显示的颜色，当框架没有焦点，请使用样式映射来控制选项;见 节 50.2“<i>ttk</i> 样式映射：动态外观变化” 。highlightcolor
highlightcolor	您可以指定默认焦点突出显示颜色，通过在样式中设置此选项。您也可以控制使用样式映射的重点突出显示颜色。
highlightthickness	配置此选项使用的样式。此选项不能在所有的主题。
padx	不受支持。
pady	不受支持。

34. *ttk*.Label

这个小部件的目的是要显示的文本、 图像或两者。一般内容是静态的但你的程序可以更改的文本或图像。

打造一个 *ttk* 小部件作为给定的控件的子级：*.Labelparent*

```
w = ttk.Label(parent, option=value, ...)
```

选项包括：

表 44. *ttk*. 标签选项

anchor	如果小于指定的文本和/或图像，您可以使用选项来指定位置他们:、 、 或左、 居中或右对齐方式， 分别。您也可以指定此选项使用的样式。widthanchortk.Wtk.CENTERtk.E														
background	使用此选项可以设置背景颜色。您也可以指定此选项使用的样式。														
borderwidth	若要添加标签的边框，请设置此选项的宽度尺寸。您也可以指定此选项使用的样式。														
class_	当您创建此小部件时，您可以提供一个小部件类名称。该名称可用于自定义小部件的外观;请参阅第 27 款“规范外观”。一旦创建了该构件，构件类名称不能更改。														
compound	<div>如果提供两个选项，此选项指定如何显示它们。 textimagecompound</div> <table><tr><td>'bottom'</td><td>显示图像文本的下面。</td></tr><tr><td>'image'</td><td>显示仅图像，不是文本。</td></tr><tr><td>'left'</td><td>到文本的左侧显示图像。</td></tr><tr><td>'none'</td><td>显示图像，如果有的话，否则显示的文本。这是默认值。</td></tr><tr><td>'right'</td><td>右侧的文本显示的图像。</td></tr><tr><td>'text'</td><td>显示的文本，不是图像。</td></tr><tr><td>'top'</td><td>显示在文字上方图像。</td></tr></table>	'bottom'	显示图像文本的下面。	'image'	显示仅图像，不是文本。	'left'	到文本的左侧显示图像。	'none'	显示图像，如果有的话，否则显示的文本。这是默认值。	'right'	右侧的文本显示的图像。	'text'	显示的文本，不是图像。	'top'	显示在文字上方图像。
'bottom'	显示图像文本的下面。														
'image'	显示仅图像，不是文本。														
'left'	到文本的左侧显示图像。														
'none'	显示图像，如果有的话，否则显示的文本。这是默认值。														
'right'	右侧的文本显示的图像。														
'text'	显示的文本，不是图像。														
'top'	显示在文字上方图像。														

cursor	使用此选项可以指定鼠标光标的外观，当它结束的部件;请参阅 一节 5.8, “游标” 。默认值 (空字符串) 指定光标从父组件继承。
font	使用此选项来指定所显示的 字体 样式。您也可以指定此选项使用的样式。text
foreground	使用此选项以指定的 颜色 所显示。您也可以指定此选项使用的样式。text
image	<p>此选项指定图像或图像将显示在增补或而不是文本。值必须是作为科 5.9、“图像”中指定的图像。请参见以上选项为您提供图像和文本的时候，会发生什么。compound</p> <p>您可以指定多个图像将显示根据状态的小部件部件上 (见节 50.2“ttk 样式映射： 动态外观变化”的组件状态的讨论)。要做到这一点，提供此选项的值作为一个元组，在哪里： (i_0, s_1, i_1, s_2, i_2, ...)</p> <ul style="list-style-type: none">i_0是小部件上显示的默认图像。为每一对值后第一次，指定状态或状态，组合，并指定要显示时构件的状态相匹配的图像。$s_i i_i s_i$ <p>每个状态说明符可能是单一的国家名称，选择前面或一系列此类名称。指定该构件必须不应在该缔约国。 $s_i '!' !$</p> <p>例如，假设您有三个实例命名、和，和在你调用构造函数中包括此选项：PhotoImageim1im2im3Label</p> <pre>self.w = ttk.Label(self, ..., image=(im1, 'selected', im2, ('!disabled', 'alternate'), im3), ...)</pre> <p>如果它处于状态，小部件将显示图像。如果它是不在状态或状态，但它是处于状态，它将显示图像。否则，它将显示图像。 im2selectedselecteddisabledalternateim3im1</p>
justify	如果您提供包含换行符 ()，此选项指定如何将水平放置的每一行：左对齐；中心;或右对齐的每一行。您也可以指定此选项使用的样式。text'\n' tk.LEFTtk.CENTERtk.RIGHT

padding	若要添加更多的空间，周围所有四个边的文本和/或图像，此选项设置为所需的 尺寸 。您也可以指定此选项使用的样式。
relief	将此选项设置为 救济 样式来创建 3 d 效果。您将需要增加，使这种效果出现。您也可以指定此选项使用的样式。 borderwidth
style	使用此选项可以指定自定义小部件样式名称;见 第 47 节，“自定义和创建 <i>ttk</i> 主题和样式” 。
takefocus	使用此选项可以指定是否在小部件访问期间焦点遍历;见 节 53“焦点：路由键盘输入” 。指定是否你想要去接受焦点;指定是否该小部件是不接受焦点。 takefocus=True takefocus=False 默认值是一个空的字符串;默认情况下， <i>ttk</i> 部件做不能集中精力。 .Label
text	在小部件中显示的文本的字符串。
textvariable	一个实例（见 节 52“控制变量：小部件背后值” ）;显示小部件上的文本将是其值。如果这两个是指定，将忽略该选项。 StringVar text textvariable text
underline	您可以请求那个文本字符串中的字母由下划线将此选项设置为位置的那封信。例如的选项，并将强调 Q。 text='Quit' underline=0 使用此选项不会更改任何功能。如果你想要对 Q 键或一些像控制转移 Q 的变化作出反应的应用程序，您需要设置绑定使用 事件系统 。
width	若要指定固定的宽度，请将此选项设置为的字符数。若要指定一个最小的宽度，请设置此选项为减去的字符数。如果不指定此选项，标签区域的大小将只足以容纳当前文本和/或图像。 为中非等宽字体显示的文本，该构件的实际宽度将基于字符的字体和不特定数量的字符的平均宽度。 也可以通过样式指定此选项。
wraplength	如果你将此选项设置为一些 维度 ，所有文本将被都切成线不再比此维度。也可以通过样式指定此选项。

Tkinter 版本的下列选项是由 *ttk* 构造函数 不受支持。Label.Label

表 45。 *Tkinter* 标签选项不在 *ttk*。 标签

activebackground	使用样式映射来控制选项; 见 节 50.2”<i>ttk</i> 样式映射： 动态外观变化” 。 background
activeforeground	使用样式映射来控制选项。 foreground
bitmap	不受支持。
disabledforeground	使用样式地图选项; 见 节 50.2”<i>ttk</i> 样式映射： 动态外观变化” 。 foreground
height	不受支持。
highlightbackground	若要控制重点突出显示的颜色， 当标签不具有焦点时， 请使用样式映射来控制选项; 见 节 50.2”<i>ttk</i> 样式映射： 动态外观变化” 。 highlightcolor
highlightcolor	您可以指定默认焦点突出显示颜色， 通过在样式中设置此选项。 您也可以控制使用样式映射的重点突出显示颜色。
highlightthickness	配置此选项使用的样式。 此选项不能在所有的主题。
padx	不受支持。
pady	不受支持。

35. *ttk*.LabelFrame

这是基本 *Tkinter* 小部件，所述[节 13](#)，[“LabelFrame 小部件”](#) *ttk* 版本。

要创建一个新的 *ttk* 小部件作为一个孩子一个给定的部件：`.LabelFrameparent`

```
w = ttk.LabelFrame(parent, option=value, ...)
```

选项包括：

表 46。 *ttk*.LabelFrame 选项

borderwidth	使用此选项可以将小部件周围边框的宽度设置为给定 维度 。也可以使用样式配置此选项。
class_	当您创建此小部件时，您可以提供一个小部件类名称。该名称可用于自定义小部件的外观；请参阅 第 27 款“规范外观” 。一旦创建了该构件，构件类名称不能更改。
cursor	使用此选项可以指定鼠标光标的外观，当它结束的部件；请参阅 一节 5.8 ， “游标” 。默认值（空字符串）指定光标从父组件继承。
height	此选项可以设置为一些 维度 ，以指定框架的高度。如果你不调用方法时，此选项将被忽略；看到 4.2 节 ， “其他网格管理方法” 。 <code>.grid_propagate(0)</code>
labelanchor	使用此选项来指定标签的位置上的部件的边界。默认位置是，哪些地方该标签在左端的上边框。9 个可能的标签职位，指 节 13 、 “LabelFrame 小部件” 。’nw’
labelwidget	<p>而不是文本标签，您可以使用任何小部件作为中 <i>ttk</i> 的标签。创建一些小部件，但没有注册它的方法。然后创建的。如果您指定此选项，该选项，则忽略后者。<code>.LabelFrame.w.grid()LabelFrame.labelwidget=wttext</code></p> <p>例如，如果你不喜欢小而平的默认字体用于标签，你可以使用此选项来显示小部件与字体和其他外观的您的选择。Label</p>
padding	若要添加额外清晰周围这个小部件的内容，请设置此选项的 维度 。由样式，也可以指定此选项。

relief	使用此选项可以指定 3 d 边框样式;请参见 第 5.6 节、“救济样式” 。您将需要指定非零的这种效应出现。由样式，也可以指定此选项。borderwidth
style	使用此选项可以指定自定义小部件样式名称;见 第 47 节，“自定义和创建 <i>ttk</i> 主题和样式” 。
takefocus	使用此选项可以指定是否在小部件访问期间焦点遍历;见 节 53“焦点：路由键盘输入” 。指定是否你想要去接受焦点;指定是否该小部件是不接受焦点。 takefocus=True takefocus=False 默认值是一个空的字符串;默认情况下， <i>ttk</i> 部件做不能集中精力。 .Label
text	此选项的值是一个字符串，它将显示为边界的一部分。
underline	您可以请求那个文本字符串中的字母由下划线将此选项设置为位置的那封信。例如，如果您指定和下划线将出现在下。 text='Panic' underline=2 n' 使用此选项不会更改任何功能。如果你想要对 Q 键或一些像控制转移 Q 的变化作出反应的应用程序，您需要设置绑定使用 事件系统 。
width	此选项可以设置为一些 维度 ，以指定框架的宽度。如果你不调用方法时，此选项将被忽略;看到 4.2 节，“其他网格管理方法” 。 .grid_propagate(0)

以下选项可供 *Tkinter* 构件 不提供构造函数的参数。LabelFrame

表 47。 *Tkinter* 不在 *ttk* 选项。LabelFrame

background 或 bg	配置选项使用样式;见 第 47 节，“自定义和创建 <i>ttk</i> 主题和样式” 。不支持缩写。backgroundbg
highlightbackground	到焦点的颜色突出显示的控件没有焦点，使用样式映射来控制选项;见 节 50.2“<i>ttk</i> 样式映射：动态外观变化” 。LabelFramehighlightcolor
highlightcolor	您可以指定默认焦点突出显示颜色，通过在样式中设置此选项。您也可以控制使用样式映射的重点突出显示颜色。
highlightthickness	配置此选项使用的样式。此选项不能在所有的主题。

Ttk 小部件支持[节 46, “所有 *ttk* 部件通用方法”](#)中所描述的所有方法。`.LabelFrame`

36. *ttk*. Menubutton

小部件是始终是可见的下拉菜单中的一部分。它总是与控制什么显示当用户点击一个部件的组合使用。MenubuttonMenuMenubutton

还有小部件没有 *ttk* 版本。使用[节 15、“菜单小部件”](#)中所述的定期 *Tkinter* 小部件。Menu

若要创建新的 *ttk* 构件为孩子的一些小部件，请使用此构造函数
：*.Menubuttonparent*

```
w = ttk.Menubutton(parent, option=value, ...)
```

选项包括：

表 48. *ttk*. 上方选项

class_	小部件类的名称。这可能指定当小部件创建，但以后不能更改。窗口小部件类的说明，请参阅 第 27 款“规范外观” 。											
compound	<p>如果提供两个选项，此选项指定图像相对于文本的位置。值可能是 （图像文本的上方），（图像文本下方），（左边的文本图像），或 （图像右侧的文本）。</p> <p>imagetextcompoundtk.TOPtk.BOTTOMtk.LEFTtk.RIGHT</p> <p>当你提供了和选项，但不指定选项，将会显示图像和文本也不会。imagetextcompound</p>											
cursor	光标将显示当鼠标位于按钮;请参阅 一节 5.8，“游标” 。											
direction	<p>此选项指定相对于上方的下拉菜单中，显示的位置。</p> <table><tr><td>above</td><td>则会出现菜单上方的上方。</td></tr><tr><td>below</td><td>则会出现菜单下方上方。</td></tr><tr><td>flush</td><td>菜单中将显示在上方，以便菜单的西北角，正值上方的西北角。</td></tr><tr><td>left</td><td>菜单中将出现只是左侧上方。</td></tr><tr><td>right</td><td>只是右侧上方将出现的菜单。</td></tr></table>		above	则会出现菜单上方的上方。	below	则会出现菜单下方上方。	flush	菜单中将显示在上方，以便菜单的西北角，正值上方的西北角。	left	菜单中将出现只是左侧上方。	right	只是右侧上方将出现的菜单。
above	则会出现菜单上方的上方。											
below	则会出现菜单下方上方。											
flush	菜单中将显示在上方，以便菜单的西北角，正值上方的西北角。											
left	菜单中将出现只是左侧上方。											
right	只是右侧上方将出现的菜单。											
image	图像出现在上方;请参阅 一节 5.9、“图像” 。											

menu	相关的小部件。用于建立此相互连接的过程，请参阅 一节 15、“菜单小部件” 。Menu
style	要呈现此上方；所用的样式见 节 49“使用和自定义 ttk 样式” 。
takefocus	默认情况下， <code>ttk</code> 将列入焦点遍历；见 节 53“焦点：路由键盘输入” 。要从焦点遍历删除小部件，请使用。 <code>.Menubuttontakefocus=False</code>
text	要显示在上方，作为一个字符串的文本。
textvariable	控件出现在上方；文本变量见 节 52“控制变量：小部件背后值” 。
underline	如果此选项有一个非负的值 n ，下划线将显示下面的字符在位置 n 。
width	如果标签是文本，此选项指定文本区域的绝对宽度在按钮上，作为字符数；实际宽度是这一数字乘以当前字体中字符的平均宽度。对于图像标签，则忽略此选项。也可能在一种风格配置选项。

Tkinter 按钮，[节 16，“上方小部件”](#)，所述的下列选项是不支持由 `ttk:Menubutton`。Menubutton

表 49。Tkinter 不在 ttk 选项。上方

activebackground	使用样式映射来控制选项；见 节 50.2“ttk 样式映射：动态外观变化” 。background
activeforeground	使用样式映射来控制选项。foreground
anchor	配置此选项使用的样式；见 节 49“使用和自定义 ttk 样式” 。使用此选项来指定文本的位置，当选项分配的额外水平空间。width
bitmap	不受支持。
borderwidth 或 bd	配置使用样式的选项。不支持缩写。borderwidthbd
buttonbackground	不受支持。
buttoncursor	不受支持。
buttondownrelief	不受支持。

buttonup	不受支持。
disabledforeground	使用样式地图选项;见 节 50.2“<i>ttk</i> 样式映射： 动态外观变化” 。foreground
font	配置此选项使用的样式。
foreground 或 fg	配置使用样式的选项。foreground
height	不受支持。
highlightbackground	若要控制重点突出显示的颜色，当上方不具有焦点时，请使用样式映射来控制选项;见 节 50.2“<i>ttk</i> 样式映射： 动态外观变化” 。highlightcolor
highlightcolor	您可以指定默认焦点突出显示颜色，通过在样式中设置此选项。您也可以控制使用样式映射的重点突出显示颜色。
highlightthickness	配置此选项使用的样式。
justify	如果包含换行符（ <code>\n</code> ）字符，文本将占据多行的上方。该选项控制如何每行水平放置。配置此选项使用的样式;值可能是、或线路，分别为左对齐、居中或右对齐。 text'\n' justifytk.LEFTtk.CENTERtk.RIGHT
padx	不受支持。
pady	不受支持。
relief	配置此选项使用的样式;见 节 49“使用和自定义 <i>ttk</i> 样式” 。
wraplength	如果此选项设置为一些 维度 ，将使用的样式切成片不再比该维度。text

37. *ttk*. Notebook

构件的目的是内容的提供一个区域，用户可以通过点击该地区，像这些顶部的选项卡选择页: Notebook



- 每次用户点击这些选项卡之一小部件将显示子窗格通常与该选项相关，每个窗格中将一个小部件，虽然一个窗格可以是任何部件。Frame
- 孩子窗格中当前显示的选项卡被指选定的选项卡。
- 您将使用的部件的方法来附加一个新的选项卡和其相关的内容。其他方法让你删除或暂时隐藏选项卡。Notebook.add()
- 每个选项卡有其自己的控制其外观和行为的选项集。介绍了这些选项在表 51, [ttk 选项选项卡。笔记本小部件](#)。
- 大量的这个小部件的方法使用的主意来引用选项卡之一。不同的值可以是任何的: *tabId**tabId*
 - 整数值是指选项卡的位置: 0 表示第一个选项卡, 1, 第二次, 等等。
 - 你总是可以使用子部件本身引用选项卡。
 - 形式的字符串是指目前包含小部件的相对于点的选项卡。例如, 字符串将指定包含点 37 像素从构件沿选项卡的顶部边缘的左侧的选项卡。“@*x, y*”(*x, y*)“@37, 0”
 - 该字符串是指当前选定哪个选项卡。“current”
 - 在对构件的方法的调用, 使用字符串“end”来确定显示选项卡的当前数目。Notebook.index()“end”

若要创建一个子部件作为孩子的一些小部件, 请使用此构造函数:

Notebook*parent*

```
w = ttk.Notebook(parent, option=value, ...)
```

选项包括:

表 50. *ttk*. 笔记本选项

class_	小部件类的名称。这可能指定当小部件创建, 但以后不能更改。窗口小部件类的说明, 请参阅 第 27 款“规范外观” 。
cursor	光标将显示当鼠标位于笔记本; 请参阅 一节 5.8, “游标” 。
height	以像素为单位来分配给该构件高度。

padding	若要添加一些额外的空间，在外面的小部件，此选项设置为这一数额的空间作为一个 维度 。
style	要呈现此上方；所用的样式见 节 49“使用和自定义 ttk 样式” 。
takefocus	默认情况下， <i>ttk</i> 将列入焦点遍历;见 节 53“焦点： 路由键盘输入” 。要从焦点遍历删除小部件，请使用。 <code>.Notebooktakefocus=False</code>
width	以像素为单位来分配给该构件的宽度。

上一个 *ttk* 小部件的方法包括所有所述[部分 46，“所有 ttk 部件的共同方法”](#)，再加上：`.Notebook`

`.add(child, **kw)`

参数通常是一个小部件，包装的独生子女窗格的内容。如果不是一个小部件的子窗格，作为下一个可用的选项卡，添加和关键字参数定义为新窗格中的选项卡选项。这些选项定义它[表 51， *ttk* 选项选项卡。笔记本小部件”](#)。`childFramechildNotebookchildkw`

如果是目前隐藏的窗格中，该选项卡将出现在其前的位置。`child`

`.enable_traversal()`

一旦您调用此方法，将工作几个额外的键绑定：

- 控制选项卡将在当前选择的一个后选择选项卡。如果当前选择了最后一个选项卡，选择将旋转回第一个选项卡。
- Shift-控制-Tab 则恰好相反： 它将移动到上一个选项卡，环绕到最后一个选项卡如果选择第一个选项卡。
- 您可以配置特定的热键，直接选择一个选项卡。若要执行此操作，请使用和选项卡选项强调每个选项卡中的人物之一。然后用户可以跳转到标签通过键入 Alt-该选项卡上带下划线的字符在哪里。`textunderlinexx`

如果您有多个小部件在同一应用程序中，这些功能将无法正常工作，除非每个孩子窗格小部件创建作为父小插件。`NotebookNotebook`

`.forget(child)`

此方法会永久删除指定的部件的一组选项卡。`child`

`.hide(tabId)`

选项卡，确定由暂时删除从可见中的选项卡集。你可以通过再次调用方法恢复它。 `tabIdNotebook.add()`

`.index(tabId)`

为给定，此方法返回对应的选项卡的数字索引。还有一个例外： 如果参数是字符串，该方法将返回选项卡的总数。 `tabId"end"`

`.insert(where, child, **kw)`

此方法在指定的使用任何关键字参数来描述新的选项卡和窗格的位置插入小部件。关键字选项，请参阅[表 51, `ttk` 选项选项卡。笔记本小部件](#)”。 `childwhere`

参数可以是任何的： *where*

- "end"将新选项卡毕竟现有的选项卡。
- 现有的儿童小部件;在这种情况下新插入之前，现有的小部件。
child

`.select([tabId])`

如果您调用此方法没有参数，它将返回当前显示的选项卡小部件[窗口名称](#)。

若要显示特定的窗格中，调用此方法与 `a` 作为参数。 Notebook *tabId*

`.tab(tabId, option=None, **kw)`

设置选项卡选项的笔下，儿童窗格或找出那孩子窗格设置哪些选项，请使用此方法。选项卡选项是[表 51 所述 `ttk` 选项卡选项。笔记本小部件](#)”。 *tabId*

如果没有关键字参数的方法调用，它将返回选项卡窗格的选项目前实际上由指定字典。 *tagId*

要找出特定选项卡选项的当前值，请调用此方法的参数""，该方法将返回该选项卡选项的值。 `Xoption=X`

若要设置一个或多个选项卡为所描述的孩子，请调用此方法使用关键字参数。例如，如果是，这个调用会改变第一个选项卡上显示的文本：
`tagIdself.nbNotebook`

```
self.nb.tab(0, text='Crunchy frog')
```

`.tabs()`

此方法返回一个[窗口名称](#)的列表的子窗格中，从第一个到最后一个的顺序。Notebook

这里有选项卡中使用的选项和方法。`.add().tab()`

表 51。Tk 的选项卡选项。笔记本小部件

compound	如果您提供两个，并且要显示在选项卡上，该选项指定如何显示它们。允许的值描述的位置相对于文本，图像和可以是任何的、 <code>tk.TOP</code> 、 <code>tk.LEFT</code> 、 <code>tk.RIGHT</code> 、 <code>tk.CENTER</code> 或。例如，将定位到文本的左侧图像。 <code>imagecompound=tk.LEFT</code>
padding	使用此选项可以添加额外的空间，周围所有四个边的面板的内容。值是一个 维度 。例如，将添加 <code>0.1</code> 周围每个侧面板内容的空间。 <code>padding='0.1i'</code>
sticky	使用此选项可以指定面板内容的位置如果它未完全填充面板区域。值是 ".grid() 法"4.1 节 所述的参数相同。例如，会在右下方（东南）角定位内容。 <code>stickysticky=tk.E+tk.S</code>
image	若要使图形的图像显示在选项卡上，作为此选项的值提供的 图像 。请参阅上面的选项指定的相对位置和当您同时指定。 <code>compoundimage</code>
text	要在选项卡上显示的文本。
underline	如果此选项有一个非负的值 <i>n</i> ，下划线将显示下面的字符在位置 <i>n</i> 的选项卡上的文本。

37. 1. 虚拟事件 *ttk* 构件.Notebook

只要选定的选项卡更改 *ttk* 的小部件，它会生成“”虚拟的事件;请参阅[一节 54. 8， “虚拟事件”](#)。 . Notebook<<NotebookTabChanged>>

38. *ttk*. PanedWindow

这是[节 19](#)、 “小部件” *ttk* 版本。打造一个 *ttk* 小部件作为给定的控件的子级：
`.PanedWindowparent`

```
w = ttk.PanedWindow(parent, option=value, ...)
```

此构造函数的选项给出在[表 52](#)， “*ttk*. PanedWindow 选项”。

表 52. *ttk*. PanedWindow 选项

class_	小部件类的名称。这可能指定当小部件创建，但以后不能更改。窗口小部件类的说明，请参阅 第 27 款“规范外观” 。
cursor	光标将显示当鼠标位于 <code>checkbutton</code> ；请参阅 一节 5.8， “游标” 。
height	高度 维度 的小部件。
orient	堆栈窗口小部件的孩子肩并肩，请使用。要把它们堆顶部到底部，请使用。默认的选项是。 <code>orient=tk.HORIZONTAL</code> <code>orient=tk.VERTICAL</code> <code>tk.VERTICAL</code>
style	要呈现这个小部件；所用的样式见 节 49“使用和自定义 <i>ttk</i> 样式” 。
takefocus	默认情况下， <i>ttk</i> 将不列入焦点遍历；见 节 53“焦点： 路由键盘输入” 。若要添加要集中遍历的构件，请使用。 <code>.PanedWindowtakefocus=True</code>
width	宽度 维度 的小部件。

这些选项 *Tkinter* 构件由 *ttk* 构造函数 不受支持：`.PanedWindow.PanedWindow`

表 53. *Tkinter* 不在 *ttk* 选项。PanedWindow

background 或 bg	配置使用样式的选项。不支持缩写。 <code>backgroundbg</code>
borderwidth 或 bd	不受支持。
cursor	当鼠标在构件；会出现光标请参阅 一节 5.8， “游标” 。
handlepad	不受支持。

handlesize	不受支持。
opaqueresize	不受支持。
relief	不受支持。
sashrelief	不受支持。
sashwidth	不受支持。
showhandle	不受支持。

在 *ttk* 方法包括所有所述[部分 46, “所有 *ttk* 部件的共同方法”](#)，再加上
`∴.PanedWindow`

`.add(w[], weight=N)`

将一个新窗格添加到窗口，任何小部件在哪里（但通常）。如果您提供一个选项，它描述了在堆叠的维度中，相对于其他窗格窗格的大小。例如，为，如果窗格 0 具有和窗格 1 具有，最初第一个窗格将 1/4 的高度和二个窗格中将有 3/4。
`wFrameweightorient=tk.VERTICALweight=1weight=3`

`.forget(what)`

删除一个窗格。该参数可能要么是索引的窗格中，计数从零或孩子小部件。

`.insert(where, w[], weight=N)`

到窗户处所指定的位置添加一个新窗格，在那里可能是索引或要在其之前插入新窗格窗格窗口小部件。*wwherewhere*

`.panes()`

此方法返回的列表的子部件。`PanedWindow`

39. *ttk*.Progressbar

这个小部件的目的是保证用户发生了一些事情。它可以在两种模式之一运行：

- 在模式中，widget 显示移动从开始结束在程序控制下的指示器。
determinate
- 在模式中，widget 被动画所以用户会相信的东西是在进步。在此模式下，指示器的小部件端点之间来回跳。indeterminate

指示器的当前位置在任何一种模式，一个数字值。您可以指定一个值，您可以直接设置的指标值。您还可以指定指标值移动给定的数量，每次经过给定的时间间隔。maximum

要创建新的 *ttk* 构件作为给定的控件的子级：`.Progressbarparent`

```
w = ttk.Progressbar(parent, option=value, ...)
```

此构造函数的选项给出在[表 54, “*ttk*. 进度选项”](#)。

表 54. *ttk*. 进度选项

class_	小部件类的名称。这可能指定当小部件创建，但以后不能更改。 窗口小部件类的说明，请参阅 第 27 款“规范外观” 。
cursor	光标将显示当鼠标位于 checkbutton; 请参阅 一节 5.8, “游标” 。
length	沿其长轴作为一个 维度 小部件的大小。
maximum	最大值的指标要求; 默认值为 100。
mode	<p>如果您的程序不能准确地描绘出这个小部件是应该显示的相对进度，使用。在此模式下，一个矩形小部件一旦你使用方法将端点之间来回跳。mode='indeterminate'.start()</p> <p>如果您的程序具有某种程度的相对进展，使用。在此模式下，您的程序可以将指示器移动到指定位置沿构件的轨道。 mode='determinate'</p>
orient	<p>此选项指定的方向： 使用或。</p> <p>orient=tk.HORIZONTALorient=tk.VERTICAL</p>
style	要呈现这个小部件；所用的样式见 节 49“使用和自定义 <i>ttk</i> 样式” 。

takefocus	默认情况下， <i>ttk</i> 将不列入焦点遍历;见 节 53“焦点： 路由键盘输入” 。若要添加要集中遍历的构件，请使用。 <i>.Progressbartakefocus=True</i>
variable	使用此选项可以将 控制变量 链接到小部件，以便您可以获取或设置指标的当前值。

在 *ttk* 的方法包括所述[部分 46， “方法共同所有 *ttk* 部件”](#)加上：*.Progressbar*

.start([*interval*])

开始移动指示器每毫秒;默认值为 50 毫秒。每一次，指示器移动，如果你调用方法。*interval.step()*

.step([*delta*])

这种方法增加指标值;默认增量是 1.0。在确定的模式，该指标将永远不会超过选项的值。在不确定的模式下，该指标将反转方向和倒计时，一旦它达到最大值。*delta*maximum

.stop()

此方法停止通过调用方法启动自动进度。*.start()*

40. *ttk*. Radiobutton

这个小部件是[节 20](#)、部件“*ttk*”版本。要创建一个 *ttk* 小部件作为一个给定小部件，在[表 55](#)，“*ttk* 给出值是孩子. 单选按钮选项”
`ttk.Radiobutton(parent, option)`

```
w = ttk.Radiobutton(parent, option=value, ...)
```

表 55. *ttk*. 单选按钮选项

class_	小部件类的名称。这可能指定当小部件创建，但以后不能更改。窗口小部件类的说明，请参阅 第 27 款“规范外观” 。
command	此单选按钮的状态发生更改时调用的函数。
compound	此选项指定图像相对于文本的相对位置，当您同时指定。值可能是（图像文本的上方），（图像文本下方），（左边的文本图像），或（图像右侧的文本）。如果你提供了和选项但不指定一个值，仅该图像将显示。 tk.TOPtk.BOTTOMtk.LEFTtk.RIGHTimagetextcompound
cursor	光标将显示当鼠标位于单选按钮;请参阅 一节 5.8, “游标” 。
image	图像显示在单选按钮;请参阅 一节 5.9、“图像” 。
style	要呈现此单选按钮;所用的样式见 节 49“使用和自定义 <i>ttk</i> 样式” 。
takefocus	默认情况下， <i>ttk</i> 将列入焦点遍历;见 节 53“焦点：路由键盘输入” 。要从焦点遍历删除小部件，请使用。 <code>.Radiobutton.takefocus=False</code>
text	旁边的单选按钮，以字符串形式显示的文本。
textvariable	控制在单选按钮;显示的文本变量见 节 52“控制变量：小部件背后值” 。
underline	如果此选项有一个非负的值 <i>n</i> ，下划线将显示下面的字符在位置 <i>n</i> 。text
value	与此单选按钮关联的值。选择组中的一个单选按钮时，此选项的值将存储在该集团的控制变量。

variable	控制变量是共享的其他单选按钮组;见 节 52“控制变量：小部件背后值” 。此变量的类型将单选按钮组中的选项为指定的类型相同。value
width	<p>使用此选项可以指定固定的宽度或最小宽度。以字符为单位;指定的值一个积极的值设置的固定的宽度的许多平均字符,而消极的宽度设置最小宽度。</p> <p>也可能在关联的样式中指定一个值。如果样式和小部件构造函数调用中指定值,前者优先。width</p>

这些选项 *Tkinter* 构件由 *ttk* 构造函数 不受支持：Radiobutton. Radiobutton

表 56。 *ttk* 单选按钮选项不在 *ttk*。 单选按钮

activebackground	使用样式映射来控制选项;见 节 50.2“ttk 样式映射：动态外观变化” 。background
activeforeground	使用样式映射来控制选项。foreground
anchor	<p>配置此选项使用的样式;见节 49“使用和自定义 ttk 样式”。使用此选项来指定文本的位置,当选项分配的额外水平空间。width</p> <p>例如,如果您指定的选项和单选按钮,显示这两个文本和图像和 (西),指定的图像将左端的三十个字符空间,它只是上面的文本样式。 width=30compound=tk.BOTTOManchor=tk.W</p> <p>当单选按钮显示的图像,但没有文字时,此选项将被忽略。</p>
background 或 bg	配置使用样式的选项。不支持缩写。backgroundbg
bitmap	不受支持。
borderwidth 或 bd	配置此选项使用的样式。
disabledforeground	使用样式地图选项;见 节 50.2“ttk 样式映射：动态外观变化” 。foreground
font	配置此选项使用的样式。
foreground 或 fg	配置使用样式的选项。不支持缩写。foregroundfg

height	不受支持。
highlightbackground	若要控制重点突出显示的颜色，当上方不具有焦点时，请使用样式映射来控制选项;见 节 50.2“<i>ttk</i> 样式映射： 动态外观变化” 。highlightcolor
highlightcolor	您可以指定默认焦点突出显示颜色，通过在样式中设置此选项。您也可以控制使用样式映射的重点突出显示颜色。
highlightthickness	配置此选项使用的样式。
indicatoron	不受支持。
justify	控制如何多行彼此之间的相对水平放置。配置此选项使用的样式;可能的值，或为左对齐、 居中或右对齐，分别。tk.LEFTtk.CENTERtk.RIGHT
offrelief	不受支持。
overrelief	不受支持。
padx	不受支持。
pady	不受支持。
relief	配置此选项使用的样式。
selectcolor	不受支持。
selectimage	不受支持。
state	在 <i>ttk</i> ，没有具有该名称的选项。被广义状态机制;见 节 50.2“<i>ttk</i> 样式映射： 动态外观变化” 。
wraplength	如果您使用具有此选项设置为一些 维度 ，将样式被切成片不再比该维度。text

在 *ttk* 方法包括所有所述[部分 46，“所有 *ttk* 部件的共同方法”](#)，再加上
`:.Radiobutton`

`.invoke()`

当你在 *ttk* 上调用此方法时，结果都是一样，因为如果用户点击了它：指示器被设置和关联设置为单选按钮。如果与此单选按钮相关联，来调用这个函数，并且该方法返回的函数返回不管;否则它将返回。`.Radiobuttonvariablevaluecommand.invoke()`None

41. *ttk*. Scale

这是一节 21 世纪 》 、 [“规模小部件” *ttk* 版本](#)。要创建一个 *ttk* 小部件作为一个给定小部件，在[表 57](#)，[“*ttk* 给出值是孩子. 规模选项”](#)：
`Scaleparentoption`

```
w = ttk.Scale(parent, option=value, ...)
```

表 57. *ttk*. 规模选项

class_	小部件类的名称。这可能指定当小部件创建，但以后不能更改。窗口小部件类的说明，请参阅 第 27 款“规范外观” 。
command	这个小部件的状态发生更改时调用的函数。此函数将接收一个参数，部件上，所示的新值。float
cursor	光标将显示当鼠标位于规模;请参阅 一节 5.8, “游标” 。
from_	使用此选项结合使用选项 （如下所述） 可以限制到一个数值范围内的值。例如，将只允许值之间 - 10 和 20 具包容性。请参阅下面的选项。tofrom_=-10to=10increment
length	规模小部件的长度。这是 <i>x</i> 尺寸如果规模是水平的或如果垂直 <i>y</i> 维度。默认值为 100 个像素。允许的值，请参阅 第 5.1 条, “维度” 。
orient	如果你想要的规模来运行沿 <i>x</i> 尺寸，或运行平行于 <i>y</i> 设置-轴。默认值是垂直的。orient=tk.HORIZONTALorient=tk.VERTICAL
style	要呈现此单选按钮；所用的样式见 节 49“使用和自定义 <i>ttk</i> 样式” 。
takefocus	默认情况下，一个 <i>ttk</i> 小部件将列入焦点遍历;见 节 53“焦点：路由键盘输入” 。要从焦点遍历删除小部件，请使用。 <code>Scaletakefocus=False</code>
to	值，该值定义一端的范围;另一端是选项，上文所定义的。值可以大于或小于值。为垂直尺度的值定义底部的规模;为水平的尺度，右端。默认值为 100。floatfrom_tofrom_to
value	使用此选项可以设置初始值的构件;默认值为 0.0。variable

variable	使用此选项可以将 控制变量 与部件相关联。通常，这将是一个实例，其中包含类型的值。您可以改用一个实例，但存储在它的值将被截断为类型。tk.DoubleVarfloattk.IntVarint
----------	---

这些选项 *Tkinter* 构件由 *ttk* 部件构造函数 不受支持：Scale.Scale

表 58。 *Tkinter* 不在 *ttk* 选项。规模

activebackground	使用样式映射来控制选项;见 节 50.2“ttk 样式映射：动态外观变化” 。background
background 或 bg	配置选项使用样式;此选项控制滑块的颜色。不支持缩写。backgroundbg
borderwidth 或 bd	配置此选项使用的样式。
digits	不受支持。
font	不受支持。
foreground 或 fg	不受支持。
highlightbackground	不受支持。
highlightcolor	不受支持。
highlightthickness	不受支持。
label	不受支持。
relief	不受支持。
repeatdelay	不受支持。
repeatinterval	不受支持。
resolution	不受支持。
showvalue	不受支持。
sliderlength	配置此选项使用的样式。
sliderrelief	配置此选项使用的样式。
state	在 <i>ttk</i> ，没有具有该名称的选项。被广义状态机制;见 节 50.2“ttk 样式映射：动态外观变化” 。
tickinterval	不受支持。

troughcolor	配置此选项使用的样式。
width	配置此选项在样式中使用的选项。sliderthickness

在 *ttk* 方法包括所有所述[部分 46, “所有 *ttk* 部件的共同方法”](#)，再加上
：`.Scale`

`.get()`

返回显示小部件上的当前值。

`.set(newValue)`

更改到小部件的当前值。*newValue*

42. *ttk*.Scrollbar

这是一节 22、 部件” *ttk* 版本。若要创建 *ttk* 小时一个给定小部件，在表 59, “*ttk* 给出值是. 滚动条选项”: `Scrollbarparentoption`

```
w = ttk.Scrollbar(parent, option=value, ...)
```

表 59. *ttk*. 滚动条选项

class_	小部件类的名称。这可能指定当小部件创建，但以后不能更改。 窗口小部件类的说明，请参阅 第 27 款“规范外观” 。
command	移动滚动条时要调用过程。调用序列的讨论，请参见 第 22.1 条, “滚动条命令回调” 。
cursor	当鼠标在滚动条上；会出现光标请参阅 一节 5.8, “游标” 。
orient	为一个垂直的水平滚动条，设置一个（缺省方向）。 <code>orient=tk.HORIZONTAL</code> <code>orient=tk.VERTICAL</code>
style	要呈现此滚动条；所用的样式见 节 49“使用和自定义 <i>ttk</i> 样式” 。
takefocus	默认情况下， <i>ttk</i> 将不列入焦点遍历；见 节 53“焦点：路由键盘输入” 。若要添加要集中遍历的构件，请使用。 <code>.Scrollbartakefocus=True</code>

这些选项 *Tkinter* 构件由 *ttk* 构造函数 不受支持：`Scrollbar.Scrollbar`

表 60. 不在 *ttk* 选项。滚动条

activebackground	使用样式映射来控制选项；见 节 50.2“<i>ttk</i> 样式映射：动态外观变化” 。 <code>background</code>
activerelief	使用样式映射来控制选项；见 节 50.2“<i>ttk</i> 样式映射：动态外观变化” 。 <code>relief</code>
background 或 bg	配置选项使用样式；此选项控制滑块的颜色。不支持缩写。 <code>backgroundbg</code>
borderwidth 或 bd	配置使用样式的选项。不支持缩写。 <code>borderwidthbd</code>
elementborderwidth	不受支持。

highlightbackground	不受支持。
highlightcolor	不受支持。
highlightthickness	不受支持。
jump	不受支持。
relief	配置此选项使用的样式。
repeatdelay	不受支持。
repeatinterval	不受支持。
troughcolor	配置此选项使用的样式。
width	配置此选项使用的样式。你可能会发现，配置更好的选择;在一些主题，增加五月不增加箭头的大小。 arrowsizewidth

在 *ttk* 方法包括所有所述[部分 46, “所有 *ttk* 部件的共同方法”](#)，再加上
`Scrollbar`

`.delta(dx, dy)`

给定的鼠标移动的像素数，此方法返回的值应该添加到当前的滑块位置，来实现，同样的动作。值必须是在闭区间 $[-1.0, 1.0]$ 。*(dx, dy)* float

`.fraction(x, y)`

给出一个像素位置，此方法返回相应的归一化在该位置最接近的间隔 $[0.0, 1.0]$ 滑块位置。*(x, y)*

`.get()`

返回两个数字 *(,)* 描述的滑块当前位置。值分别; 给出了滑块，为水平和垂直滚动条的左边或上边边缘位置值的位置的右边或底部。每个值是在区间 $[0.0, 1.0]$ 左侧或顶端的立场是 0.0 和 1.0 是最右边或底部的位置。例如，如果滑块从半路延伸到四分之三的沿槽的方式，你可能会回来元组 *(0.5, 0.75)*。*abab*

`.set(first, last)`

要连接到另一个小部件，滚动条设置的或滚动条上的方法。参数具有相同的含义由该方法返回的值。请注意，移动滚动条上的滑块并不移动相应的部件。`wxscrollcommandscrollcommand.set.get()`

43. *ttk*. Separator

使用这个小部件可放置分离其他小部件的水平或垂直的线。小部件呈现为 2 个像素宽线。一定要使用[.grid\(\) 方法](#)的选项来拉伸构件，或者它将显示为只有一个单一的像素。`sticky`

若要创建 *ttk* 小部件一个给定小部件，在[表 61](#)，"*ttk* 给出值是. 分离器选项"
`_.Separatorparentoption`

```
w = ttk.Separator(parent, option=value, ...)
```

表 61. *ttk*. 分离器选项

<code>class_</code>	小部件类的名称。这可能指定当小部件创建，但以后不能更改。窗口小部件类的说明，请参阅 第 27 款“规范外观” 。
<code>orient</code>	设置为垂直水平分隔符，一个 （缺省方向）。 <code>orient=tk.HORIZONTALorient=tk.VERTICAL</code>
<code>style</code>	要呈现此滚动条；所用的样式见 节 49“使用和自定义 <i>ttk</i> 样式” 。您可以配置的唯一风格特征是，它指定分隔符条形图的颜色；默认颜色为灰黑色。 <code>background</code>

Ttk 小部件上可用的唯一方法是[部分 46](#)，"[方法共同所有 *ttk* 部件](#)"中列出的。`_.Separator`

44. *ttk*.Sizegrip

使用这个小部件提供一个小工具，用户可以使用来调整整个应用程序窗口的大小。通常这个小部件将位于右下角的应用程序。一定要让整个应用程序调整大小使用[第 4.3 节“配置列和行大小”](#)和[“制作根窗口调整大小”](#)的[第 4.4 节](#)中所述的技术。

若要创建 *ttk* 小部件一个给定小部件，在[表 62](#)，*ttk* 给出值是 *.Sizegrip* 选项 *__::Sizegrip**parentoption*

```
w = ttk.Sizegrip(parent, option=value, ...)
```

表 62. *ttk*.Sizegrip 选项

class_	小部件类的名称。这可能指定当小部件创建，但以后不能更改。窗口小部件类的说明，请参阅 第 27 款“规范外观” 。
style	要呈现这个小部件；所用的样式见 节 49“使用和自定义 <i>ttk</i> 样式” 。您可以配置的唯一风格特征是，它指定小部件的颜色。background

45. *ttk*.Treeview

Ttk 构件的目的是呈现一个层次结构，以便用户可以使用鼠标操作来显示或隐藏任何结构的一部分。*.Treeview*

协会与“树”一词是由于编程实践：树状结构是一个平凡的在程序设计。严格地说，一个小部件所示的层次结构是一片森林：还有没有人根，只是收藏的顶级节点，每一种可能包含第二级节点，每一种可能包含三级节点，依此类推。

Treeview

您可能会遇到此特定的演示文稿作为浏览目录或文件夹层次结构中的一种方式。整个层次结构显示像缩进的大纲，在那里每个目录是在单独的行，和下面这一行的缩进显示每个目录的子目录：



用户可以点击目录到崩溃（附近）的图标，隐藏所有的项目。他们也可以点击再次扩大（开放）的图标，以便显示目录或文件夹中的项目。

小部件概括了这一概念，所以，你可以使用它来显示任何层次的结构，和读者可以折叠或展开子树上的这种结构用鼠标。*Treeview*

第一，一些定义：

项目

在小部件中显示的实体之一。文件浏览器，一个项目可能是一个目录或一个文件。

每个项目是与文本的标签，关联，也可能是与图像关联。

iid

树中的每个项目有一个唯一标识符字符串，称为 *iid*。你可以供 *iid* 值自己，或者你可以让 *ttk* 生成它们。

儿童

直接下面某一项目层次结构中的项目。一个目录，例如，可能有两种孩子：文件和子目录。

父

对于一个给定的项目，如果它位于层次结构顶层据说是没有父级；如果它不是最高一级，父是包含它的项目。

祖先

祖先的项目包括其父，其父项的父，树的顶尖水平等等。

可见

顶级项目始终是可见的。否则，项目是可见的只有当它的祖先扩大。

后裔

项的子项，包括儿童，其儿童的儿童和等等。另一种说法这是项目的子树包含它的后裔。

标签

您的程序可以将一个或多个 *标记* 字符串与每个项目相关联。您可以使用这些标签控制项的外观。例如，您可以用标记，标记与标记的目录和文件，然后指定标记的项目使用黑体字体。'd' 'f' 'd'

也可能将事件标记，与相关联，以便某些事件将导致某些处理程序被称为具有该标记的所有项。例如，您可以设置文件浏览器，以便当用户点击一个目录，浏览器更新其内容以反映当前的文件结构。

您的小部件的结构将具有多个列。第一列，我们会打电话 *图标列*，显示的图标，折叠或展开的项目。在其余的列，您可以显示你喜欢的任何信息。

Treeview

例如，一个简单的文件浏览器小部件可能会这样用两列目录图标的第一列和第二列中的目录或文件的名称。或者，您可能希望在其他列中显示文件大小、权限和其他相关的数据。

小部件的操作甚至允许您使用它作为一个树编辑器。您的程序可以从其主要树中的位置删除整个子树，然后在以后附加在一个完全不同的点。Treeview

这里是设置一个小部件的一般程序。Treeview

1. *Ttk* 构造函数创建的小部件。使用关键字参数指定的显示，将符号名称分配给每个列的列数。`.treeviewcolumns`
2. 使用 `column()` 方法建立了列标题（如果你想他们）和配置列属性，如大小和拉伸性。`.column().heading()`

3. 开始与顶级条目，使用方法填充树。每次调用此方法将一个项添加到树。使用此方法的关键字参数指定是否该项目最初展开或折叠。`.insert()` open

如果您想要提供这一项目的 `iid` 值，使用关键字参数。如果您省略此参数，`ttk` 会编造一个并返回它作为方法调用的结果。`iid.insert()`

使用此方法的关键字参数指定应显示什么内容在这一项目的每一列中时它是可见的。`values`

若要创建在一个给定的部件内的部件：`Treeviewparent`

```
w = ttk.Treeview(parent, option=value, ...)
```

构造函数将返回新的小部件。其选项包括：`Treeview`

class_	当您创建此小部件时，您可以提供一个小部件类名称。该名称可用于自定义小部件的外观;请参阅 第 27 款“规范外观” 。一旦创建了该构件，构件类名称不能更改。
columns	<p>列标识符字符串序列。内部使用这些字符串，以确定该小部件中的列。图标栏，始终是其标识符，包含折叠/展开图标，而且始终是第一列。'#0'</p> <p>你用参数指定的列是除了图标列。<code>columns</code></p> <p>例如，如果您指定，三列将出现在小部件： 第一个图标列，然后两个更多的列，其内部的标识符是和。 <code>columns=('Name', 'Size')'Name''Size'</code></p>
cursor	使用此选项可以指定鼠标光标的外观，当它结束的部件;请参阅 一节 5.8, “游标” 。默认值（空字符串）指定光标从父组件继承。
displaycolumns	<p>选择实际上显示哪些列并确定其演示文稿的顺序。可能的值：</p> <ul style="list-style-type: none">• '#all' 选择所有列，并将它们显示在参数所定义的顺序。<code>columns</code>• 列数字（整数位置，从 0 开始计数） 或从参数列标识符的列表。<code>columns</code> <p>例如，假设您指定。这意味着每次调用方法将需要一个参数提供值，将显示。让我们叫这个序列逻辑</p>

	<p><i>列顺序</i>。columns=(' Name', ' Size', ' Date').insert() values=(<i>name, size, date</i>)</p> <p>进一步假设，在构造函数中指定。<i>物理列序列</i>，实际上会在小部件中，显示的列将三： 图标列将第一位，其次是日期列 （列逻辑序列中索引 2），其次是名称列 （逻辑列索引为 0）。大小列不会出现。columns=(2, 0)</p> <p>您可以通过指定列标识符而不是逻辑列位置获得相同的效果:。columns=(' Date', ' Name')</p>																									
height	在小部件中的行所需的高度。																									
padding	<p>使用此参数将周围小部件里面内容的额外空间。您可以提供单个维度或达四个维度，根据此表解释序列：</p> <table><tr><th>给出的值</th><th>左</th><th>返回页首</th><th>权利</th><th>底部</th></tr><tr><td>一个</td><td>一个</td><td>一个</td><td>一个</td><td>一个</td></tr><tr><td><i>b</i></td><td>一个</td><td><i>b</i></td><td>一个</td><td><i>b</i></td></tr><tr><td><i>b c</i></td><td>一个</td><td><i>c</i></td><td><i>b</i></td><td><i>c</i></td></tr><tr><td><i>具有</i></td><td>一个</td><td><i>b</i></td><td><i>c</i></td><td><i>d</i></td></tr></table>	给出的值	左	返回页首	权利	底部	一个	一个	一个	一个	一个	<i>b</i>	一个	<i>b</i>	一个	<i>b</i>	<i>b c</i>	一个	<i>c</i>	<i>b</i>	<i>c</i>	<i>具有</i>	一个	<i>b</i>	<i>c</i>	<i>d</i>
给出的值	左	返回页首	权利	底部																						
一个	一个	一个	一个	一个																						
<i>b</i>	一个	<i>b</i>	一个	<i>b</i>																						
<i>b c</i>	一个	<i>c</i>	<i>b</i>	<i>c</i>																						
<i>具有</i>	一个	<i>b</i>	<i>c</i>	<i>d</i>																						
selectmode	<p>此选项控制什么允许用户用鼠标选择。值可以是：</p> <table><tr><td>selectmode=' browse'</td><td>用户可能只能选择一个项目在一段时间。</td></tr><tr><td>selectmode=' extended'</td><td>用户可以一次选择多个项目。</td></tr><tr><td>selectmode=' none'</td><td>用户不能选择用鼠标的项目。</td></tr></table>	selectmode=' browse'	用户可能只能选择一个项目在一段时间。	selectmode=' extended'	用户可以一次选择多个项目。	selectmode=' none'	用户不能选择用鼠标的项目。																			
selectmode=' browse'	用户可能只能选择一个项目在一段时间。																									
selectmode=' extended'	用户可以一次选择多个项目。																									
selectmode=' none'	用户不能选择用鼠标的项目。																									
show	若要取消标签在每一列的顶部，请指定。默认设置是显示列标签。show=' tree'																									
style	使用此选项可以指定自定义小部件样式名称;见 第 47 节，“自定义和创建 ttk 主题和样式” 。																									
takefocus	使用此选项可以指定是否一个小部件访问期间焦点遍历;见 节 53“焦点： 路由键盘输入” 。指定是否你想要去接受焦点;指定是否该小部件是不接受焦点。默认值是一个空的字																									

	字符串;默认情况下， <code>ttk</code> 部件获得焦点。 <code>takefocus=True</code> <code>takefocus=False</code> . <code>Treeview</code>
--	---

上一个小程序在这里备有的方法。`Treeview`

`.bbox(item, column=None)`

对于具有 `iid` 项，如果该项目是当前可见，此方法返回一个元组，哪里的小部件，相对于该项目的左上角坐标和的宽度和高度，以像素为单位的项目，并。如果看不到该项目，则该方法返回空字符串。
`item(x, y, w, h) (x, y) wh`

如果省略可选参数，你得到的整个行的边界框。若要获取一个特定列项的行的边框，使用列的整数索引，或者其列标识符在哪里。
`column``column=CC`

`.column(cid, option=None, **kw)`

这种方法配置所，指定的逻辑列的外观可能是一个列的索引或列标识符。若要配置图标列，请使用的值。`cid``cid #0`

在小程序中的每一列有其自己的此表中的选项集：`Treeview`

<code>anchor</code>	指定列的内容的位置的位置的 锚 。默认值为。 <code>'w'</code>
<code>id</code>	列名称。此选项是只读和集时调用的构造函数。
<code>minwidth</code>	以像素为单位；列的最小宽度默认值为 20。
<code>stretch</code>	如果此选项，则将调整列的宽度，当调整大小的窗口小部件。默认设置为。 <code>True</code> <code>1</code>
<code>width</code>	以像素为单位；列的初始宽度默认值为 200。

- 如果没有价值或任何其他关键字参数提供，则该方法返回指定列的列选项字典。`option`
- 若要查询命名选项的当前值，请使用参数。`X option=X`
- 若要设置一个或多个列的选项，可以通过使用如上所示，例如，要居中列内容选项名称的关键字参数。`anchor=tk.CENTER`

`.delete(*items)`

论据是 `iid` 值。在小程序中有匹配的 `iid` 值的所有项目都毁，所有他们的后裔。

`.detach(*items)`

论据是 iid 值。在小部件中有匹配的 iid 值的所有项目都是可见的小部件，以及他们的后代从中都删除。

项目不受破坏。你可能会附加到可见的树，使用下面描述的方法。`.move()`

`.exists(iid)`

返回如果存在与给定，或其他部件中的项目。如果项目不是当前可见的因为它已被删除的方法，它仍是存在的目的的方法。

`True iid False.detach().exists()`

`.focus([iid])`

如果您不提供此方法的参数，你回来要么 iid 的项的当前具有焦点，或如果没有项目具有焦点。’’

通过将其 iid 作为参数传递给此方法，可以将[焦点](#)给项目。

`.get_children([item])`

返回由参数指定项的子项的 iid 值的元组。如果省略该参数，你得到一个元组包含 iid 值的顶级项。*item*

`.heading(cid, option=None, **kw)`

使用此方法来配置显示的列标题构件为所指定的列的顶部，这可能是一个列的索引或列标识符。使用参数的值来配置通过图标列的标题。

cidcid #0

每个标题有其自己的具有这些名称和值的选项集：

anchor	锚点，指定标题在列；内的对齐方式请参见 5.5 条 、“ 锚 ”。默认值为。tk.W
command	当用户点击这个列标题时要调用的过程。
image	图形的列标题，（与或而不是文本标题），请将此选项设置的图像，作为指定 节 5.9 、“ 图像 ”。
text	你想要显示在列标题中的文本。

- 如果你不提供任何关键字参数，该方法将返回显示当前设置的列的标题选项一本字典。

- 若要询问一些标题选项的当前值，使用参数的形式;该方法将返回该选项的当前值。 $X_{option}=X$
- 您可以设置一个或多个标题选项通过提供他们作为关键字参数如`""`。`anchor=tk.CENTER`

`.identify_column(x)`

在给定 x 坐标，此方法返回一个字符串标识的列包含的 x 坐标的形式。`'#n'`

假设图标列显示的值为 0 的图标列;1 第二物理列;2 第三个物理列;等等。还记得物理列数可能不同于在哪里你重新排列它们使用构造函数的参数的情况下的逻辑列号。`ndisplaycolumnsTreeview`

如果不显示图标列，则的值是 1 物理第一列，2 为第二个，依此类推。 n

`.identify_element(x, y)`

返回的名称的元素位置相对于窗口小部件，或如果没有元素出现在那个位置。元素名称讨论[节 50、`"ttk 元素层"`](#)。 $(x, y)''$

`.identify_region(x, y)`

在给定坐标点相对于小部件，此方法返回一个字符串，指示该构件的哪一部分包含这一点。返回值可能包括：

<code>'nothing'</code>	重点不是内向的小部件的功能部分。
<code>'heading'</code>	点是其中一个列标题。
<code>'separator'</code>	这个点位于内列标题行，但在列之间的分隔符。使用方法来确定哪一列位于只是对这种分离器的左边。 <code>.identify_column()</code>
<code>'tree'</code>	这个点位于在图标栏内。
<code>'cell'</code>	这个点位于项目行中但不是在图标栏内。

`.identify_row(y)`

如果 y -坐标是内的项目之一，此方法将返回该项的 `iid`。如果那垂直坐标不是在项目内，此方法返回一个空字符串。 y

`.index(iid)`

此方法返回与指定相对于其父，从零开始计数项的索引。 iid

`.set_children(item, *newChildren)`

使用此方法更改的儿童 iid 的项集。该参数是一个 iid 字符串序列。不在有任何当前儿童将被删除。

itemnewChildrenitemnewChildren

`.insert(parent, index, iid=None, **kw)`

此方法将新项添加到树中，并返回该项目 iid 值。参数：

<i>parent</i>	若要插入一个新的顶级项目，使该参数为空字符串。父项的 iid 作为现有项的子级插入一个新的项目，使这一论点。
<i>index</i>	此参数指定的位置之间这位家长的孩子在那里你想要添加新项目。例如，若要插入的项目作为新的第一个孩子，使用值为零；若要在父级的第一个孩子后插入它，请使用值为 1；等等。若要作为父级的最后一个子级添加新项目，使此参数的值。'end'
<i>iid</i>	作为一个字符串值，可供 iid 的项目。如果你不提供 iid，一会自动生成，并由该方法返回。

你也可以作为此方法的关键字参数指定项目选项的数目。

<i>image</i>	通过提供一个参数，指定的位置作为图像在 科 5.9、“图像” ，可能只是右边的图标这项行显示的图像。 <i>image=II</i>
<i>open</i>	此选项指定是否该项目最初将开放。如果你提供，这一项目将会封闭。如果您提供项目的儿童将可见每当项目本身是可见。默认值为。 <i>open=Falseopen=TrueFalse</i>
<i>tags</i>	您可能提供一个或多个要与此项目相关联的标记字符串。值可以是单个字符串或一个字符串序列。
<i>text</i>	您可能提供在这一项目的图标列中显示的文本。如果给出，此文本将出现只是右边的图标，并且右侧的图像，如果提供。
<i>values</i>	此参数提供该项目的每一列中显示的数据项目。提供的值的逻辑列顺序。如果提供的值太少，则剩余的列将显示为空白在这一项目；如果提供的值太多，将丢弃临时演员。

`.item(iid[, option[, **kw]])`

使用此方法来设置或检索的选项中指定的项目。请参阅上面的项目选项的名称的方法。*iid.insert()*

不带任何参数，它返回的字典的键是选项名称和相应的值是这些选项的设置。若要检索给定选项的值，将作为第二个参数传递选项的名称。若要设置一个或多个选项，它们作为关键字参数传递给该方法。

`.move(iid, parent, index)`

移动到指定位置的项下的值指定的项目。和参数的工作，同时作为这些参数的方法。*iidparentindexparentindex.index()*

`.next(iid)`

如果指定的项目不是其父级的最后一个子级，此方法返回的 *iid* 以下的儿童;如果它是其父级的最后一个子级，则此方法返回一个空字符串。如果指定的项的顶级项，该方法将返回 *iid* 的下一个顶级项目或空字符串，如果指定的项是最后一个顶级项目。*iid*

`.parent(iid)`

如果指定的项目是一个顶级的项，此方法返回一个空字符串;否则它将返回该项的父的 *iid*。*iid*

`.prev(iid)`

如果指定的项目不是其父级的第一个孩子，此方法返回的 *iid* 的前儿童;否则，将返回一个空字符串。如果指定的项是一个顶级的项目，此方法将返回 *iid* 的以前的顶级项目或一个空字符串，如果它是第一的顶级项。*iid*

我们会见面 (*iid*)

此方法确保指定的项目，是可见的。它封闭的任何的祖先打开的。小部件要滚动，如果有必要，以便显示该项目。*iid*

`.selection_add(items)`

除了已经选定的任何项目，添加指定的。参数可能是 *iid* 的单一的 *iid* 或一连串。*items*

`.selection_remove(items)`

取消选择参数，可以是单一的 *iid* 或序列 *iid* 所指定的任何项目。

`.selection_set(items)`

只有指定将被选中;如果任何其他项目被选中之前，他们将变成未选中。*items*

`.selection_toggle(items)`

参数可能是 iid 的单一的 iid 或一连串。为每个项目指定的参数，如果它被选择，取消选择它；如果未选中，请选择它。

`.set(iid, column=None, value=None)`

使用此方法检索或设置指定的项的列的值。具有一个参数，该方法将返回一个字典： 键列标识符，并且每个相关值是相应的列中的文本。

iid

带有两个参数，该方法返回的数据值从其列标识符是参数的选定项的列。具有三个参数指定列项的值设置为第三个参数。column

`.tag_bind(tagName, sequence=None, callback=None)`

此方法将绑定到已标记的所有项目参数所指定的事件处理程序。和参数工作相同的[节 26、“通用构件方法”](#)中描述的方法和参数。

callback *tagName*sequencecallbacksequencefunc.bind()

`.tag_configure(tagName, option=None, **kw)`

此方法可以审问，或者设置影响已标记的所有项的外观的选项。标签选项包括： *tagName*

'background'	背景 颜色 。
'font'	文本字体 。
'foreground'	前景 颜色 。
'image'	图像 显示在给定标记的项目。

当调用具有一个参数，它返回一个字典的当前标记选项。若要返回特定选项的值，请使用作为第二个参数。 *XX*

若要设置一个或多个选项，如使用关键字参数。foreground='red'

`.tag_has(tagName[], iid])`

用一个参数调用，此方法返回 iid 值进行标记的所有项的列表。如果您提供 iid 作为第二个参数，该方法返回如果与那 iid 的项目都有标记，否则。 *tagName*True *tagName*False

`.xview(*args)`

这是常用的方法为连接到一个可滚动的小部件的一个水平滚动条。有关详细信息，请参阅[第 22.1 条](#)，[“滚动条命令回调”](#)。

`.yview(*args)`

这是常用的方法为连接到一个可滚动的小部件的一个垂直滚动条。有

45.1。 *ttk* 构件虚拟事件.Treeview

在小部件内的某些状态更改生成虚拟事件，您可以使用以应对这些变化；请参阅[一节 54.8，*“虚拟事件”*](#)。Treeview

- 每当更改所选内容，要么成为选定的项目中或成为未选定的小部件生成""事件。<<TreeviewSelect>>
- 无论何时打开项目，小部件生成""事件。<<TreeviewOpen>>
- 每当一个项目封闭的小部件生成""事件。<<TreeviewClose>>

46. 共同所有 *ttk* 小部件的方法

此处所示的方法是可用在所有 *ttk* 部件上。

.cget(*option*)

此方法返回指定的值。*option*

.configure(*option=value, ...*)

若要设置一个或多个小部件选项，请使用关键字参数的窗体。例如，要设置一个小部件的你可能会使用参数如””。

*option=value*fontfont=(' serif', 12)

如果你不提供任何参数，该方法将返回所有的部件的当前选项值的字典。在这本词典，键将选项名称，和相关的每个值将一个元组：

(*name, dbName, dbClass, default, current*)

<i>name</i>	选项名称。
<i>dbName</i>	数据库选项的名称。
<i>dbClass</i>	数据库类的选项。
<i>default</i>	选项的默认值。
<i>current</i>	选项的当前值。

.identify(*x, y*)

用于确定哪些元素是在小部件内的给定位置。如果在小部件内某处相对于小部件的点，此方法返回在该地位；或元素的名称否则，将返回一个空字符串。(*x, y*)

.instate(*stateSpec, callback=None, *args, **kw*)

这样做的目的来确定部件是否在指定的状态或状态的组合。

如果您提供一个可调用的值作为参数和小部件的状态相匹配或状态参数，就可调用所指定的组合将与定位参数和关键字参数调用。如果部件的状态不匹配，将不会调用。

*callbackstateSpec*args**kwstateSpeccallback*

如果您没有提供一个参数，该方法将返回如果部件的状态相匹配，否则。*callbackTruestateSpecFalse*

论点的结构，请参阅[一节 46.1 州“指定小部件在 *ttk*”](#)。 *stateSpec*

.state(stateSpec=None)

要么使用此项来查询一个小部件来确定其当前的状态，或要设置或清除一个状态。

如果你提供描述[节 46.1 州“指定小部件在 *ttk*”](#)的形式参数，该方法将设置或清除国家根据这一论点的小部件。 *stateSpec*

例如，构件，该方法将清除的小部件集，将其状态设置。

```
ww.state(['!disabled', 'selected']) 'disabled' 'selected'
```

46.1。 在 *ttk* 中指定小组件状态

在 *ttk* 的几种方法需要一个参数，指定一个特定的构件的状态的组合。这种说法可能是下列任一操作：*stateSpec*

- 如一个单一的国家名称。小部件是在这种状态，例如，当鼠标光标位于该按钮和鼠标按钮 1 被按下。'pressed' *ttk.Button*
- 一个单一的国家名字前面带有感叹号 (!) ;仅当该状态是关闭，这匹配时构件的状态。!

例如，参数指定一个小部件，目前未被按下状态。

stateSpec ' !pressed'

- 国家名称或状态名称序列之前先。只有当其所有组成部分与这类匹配。例如，值为一个小部件仅当匹配禁用该窗口部件，它具有焦点。
' !' *stateSpecstateSpec* (' !disabled', ' focus')

47. 自定义和创建 *ttk* 主题和样式

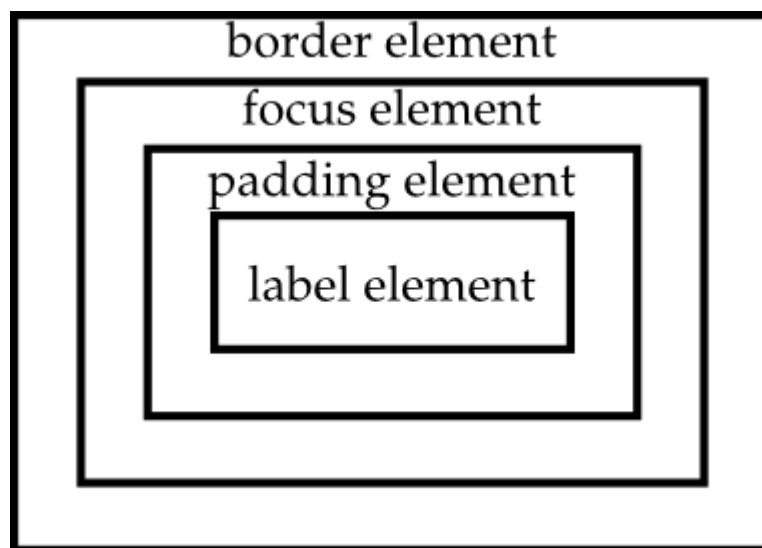
小部件与 *ttk* 设计涉及到三个抽象级别：

- 一个 *主题* 是一个完整“外观和感觉”，“自定义外观的所有小部件。
- *风格* 是外观的一种小部件的说明。每个主题附带一组预定义的样式，但您可以自定义内置样式或创建您自己的新样式。

短语“的小部件”在前一段在技术上是指“类”的小部件。然而，在 *ttk* 的世界，这是不同于 Python 的类。在 *ttk* 内的一个小部件类是字符的字符串。例如，一个股票的小部件的 *ttk* 类是的字符串。

Button' TButton'

- 每种样式是由一个或多个 *元素* 组成。例如，一个典型的按钮样式具有四个要素：边框外面；一个焦点元素，更改颜色，当小部件已经有输入 [焦点](#)；填充元素；和按钮的标签（文本、图像或两者）。



我们将讨论发现、使用和自定义的每个单独的部分中的这些图层。

- [节 48“发现和使用 *ttk* 主题”](#)。
- [节 49“使用和自定义 *ttk* 样式”](#)。
- [节 50、 “*ttk* 元素层”](#)。

48. 找到和使用 *ttk* 主题

大量的与主题相关的操作要求您具有可用（在类 Python 感觉）类的一个实例。例如，若要获得可用主题列表中您的安装：`ttk.Style()`

```
>>> import ttk
>>> s=ttk.Style()
>>> s.theme_names()
('clam', 'alt', 'default', 'classic')
```

该方法返回一个元组包含可用的样式的名称。主题给你的原始、预 *ttk* 外观。`.theme_names()` 'classic'

要确定你得到默认的主题，请使用不带任何参数的方法。若要更改当前主题，请调用此相同的方法与所需的主题名称作为参数：`.theme_use()`

```
>>> s.theme_use()
'default'
>>> s.theme_use('alt')
>>> s.theme_use()
'alt'
```


49. 使用和自定义 *ttk* 样式

在一个给定的主题，每个小部件具有默认 *窗口小部件类*; 我们使用这个术语来区分 *ttk* 类从 Python 类。

每个插件也有一种 *风格*。一个小部件的默认样式由其窗口小部件类，但您可以指定不同的样式。

在 *ttk*，窗口小部件类和样式指定为字符串。在所有情况下，一个小部件的默认样式名称前缀的小部件名称; 例如，默认按钮窗口小部件类是。有一些例外情况：`'T' TButton`

表 63。 *Ttk* 窗口小部件类的样式名称

窗口小部件类	样式名称
Button	TButton
Checkbutton	TCheckbutton
Combobox	TCombobox
Entry	TEntry
Frame	TFrame
Label	TLabel
LabelFrame	TLabelFrame
Menubutton	TMenubutton
Notebook	TNotebook
PanedWindow	TPanedwindow (!) TPanedWindow
Progressbar	Horizontal.TProgressbar 或者，取决于该选项。 Vertical.TProgressbarorient
Radiobutton	TRadiobutton
Scale	Horizontal.TScale 或者，取决于该选项。 Vertical.TScaleorient
Scrollbar	Horizontal.TScrollbar 或者，取决于该选项。 Vertical.TScrollbarorient

窗口小部件类	样式名称
Separator	TSeparator
Sizegrip	TSizegrip
Treeview	Treeview (!) TTreview

在运行时，可以通过调用其方法来检索一个部件的窗口小部件类。`.winfo_class()`

```
>>> b=ttk.Button(None)
>>> b.winfo_class()
'TButton'
>>> t=ttk.Treeview(None)
>>> t.winfo_class()
'Treeview'
>>> b.__class__      # Here, we are asking for the Python class
<class ttk.Button at 0x21c76d0>
```

样式的名称可能有两种形式之一。

- 内置的样式是一个单个的词： 或为例。'TFrame' 'TRadiobutton'
- 若要创建派生自一个内置样式的新样式，请使用窗体的样式名称。例如，要创建一种新型的小部件来存放日期，可能会调用它。
'newName.oldName' Entry' Date. TEntry'

每个样式具有一组对应的选项，以定义其外观。例如，按钮有一个选项，更改该按钮的文本的颜色。`foreground`

若要更改样式的外观，请使用它的方法。此方法的第一个参数是样式的您想要配置，其次是样式的关键字参数指定的选项名称和值您想要更改的名称。例如，若要使所有你的按钮使用绿色文本，哪里是在类的实例
：`.configure() sttk.Style`

```
s.configure('TButton', foreground='green')
```

要创建新样式基于一些样式，首先创建一个实例，然后调用其方法使用窗体的名称。例如，假设你不想使用栗色文本上你所有的按钮，但是您想创建一个新的样式，并使用栗色的文本，并且想要调用新样式：

```
oldNamettTk.Style.configure() 'newName.oldName' 'Kim.TButton'
```

```
s = ttk.Style()
s.configure('Kim.TButton', foreground='maroon')
```

然后要在新的类中创建一个按钮您可以使用这样的事情：

```
self.b = ttk.Button(self, text='Friday', style='Kim.TButton',
                    command=self._fridayHandler)
```

你可以样式甚至生成整个层次结构。例如，如果您配置命名样式，该样式将从样式继承的所有选项，那就是，你不在样式中设置的任何选项将样式中的该选项相同。

```
'Panic.Kim.TButton''Kim.TButton''Panic.Kim.TButton'Kim.TButton'
```

当 *ttk* 确定要使用的选项的值时，它看起来第一的风格；如果没有为该样式中的该选项的值，它看起来的风格；如果该样式不定义选项，看起来风格。

```
'Panic.Kim.TButton''Kim.TButton''TButton'
```

还有一个根风格的名字是。若要更改每个小部件的一些功能的默认外观，您可以配置这种风格。例如，假设您想要 12 磅的 Helvetica，（除非重写的另一种风格或选项）的所有文本。此配置将做到：'.font

```
s = ttk.Style()
s.configure('.', font=('Helvetica', 12))
```

50. *ttk* 元素层

Ttk 元素是组成一个小部件的作品之一。了解如何将元素组装成样式，请阅读这些部分。

- [节 50.1“*ttk* 布局：构建一种风格”](#)：一个小部件中的元素的静态结构。
- [节 50.2“*ttk* 样式映射：动态外观变化”](#)：构件的国家，这些国家是如何影响其外观。

50.1。 *ttk* 布局： 构建一种风格

一般情况下，破碎的小部件的装配都采用腔，一个空的空间，是充满着元素的想法。

例如，在这一主题，按钮具有四个同心圆要素。由外至内，它们是重点突出、边框、填充、和标签的元素。`classic`

每个这些元素有指定多少腔“坚持”四个边的一个属性。例如，如果元素具有属性，这意味着它必须坚持以左（西）和其腔，右（东）两侧伸展，但它不会垂直拉伸。`'sticky' sticky='ew'`

大多数内置 *ttk* 样式使用布局的想法组织组成一个小部件的不同层。那就假设的实例检索该样式布局使用此窗体，方法调用在哪里 `widget` 类的名称。

`Sttk.StylewidgetClass`

`S.layout(widgetClass)`

一些窗口小部件类没有布局;在这些情况下，此方法调用将引发异常。

`tk.TclError`

为有一个布局窗口小部件类，则返回的值是元组的列表。内每个元组，是元素的名称，是一本字典描述的元素。`(eltName, d) eltNamed`

这本词典可能有以下键的值：

`'sticky'`

一个字符串，定义此元素的方式定位在其父级内。此字符串可能包含零个或更多字符、`、`、`、`和`、`，指的双方有相同的约定对于锚箱。例如，值会拉伸此元素中坚持南、北、西三面腔内其父元素。

`'n''s''e''w' sticky='nsw'`

`'side'`

对于包含多个子元素，此值定义了元素的子元素在它里面放置的方式。值可能是`、``、``、`或`。'left''right''top''bottom`

`'children'`

如果在此元素内部的元素，此项在词典中的是使用相同的格式作为顶级的布局，就是两个元素的元组的列表的子元素的布局。`(eltName, d)`

让我们深入剖析此会话示例的主题的股票部件的布局。Button' classic'

```
>>> import ttk
>>> s = ttk.Style()
>>> s.theme_use('classic')
>>> b = ttk.Button(None, text='Yo')
>>> bClass = b.winfo_class()
>>> bClass
'TButton'
>>> layout = s.layout('TButton')
>>> layout
[('Button.highlight', {'children': [('Button.border', {'border':
'1', 'children': [('Button.padding', {'children': [('Button.label',
{'sticky': 'nswe'})]}, 'sticky': 'nswe'})]}, 'sticky': 'nswe'})],
'sticky': 'nswe'})]
```

所有这些括号、方括号和大括号，使这种结构有点难理解。在这里，它是以大纲形式：

- 最外层的元素是重点突出；它具有风格。它的属性是，意思它应该在所有的四个方向，以填补其腔扩大。'Button.highlight''sticky''nswe'
- 重点突出显示的唯一的子是具有样式的边框元素。它的宽度为 1 个像素，以及其属性还指定它坚持其谐振腔由的突出显示元素内部定义的所有四个边。'Button.border''border''sticky'
- 边框内是填充的一层，风格。其属性还指定它填充其腔。
'Button.padding' sticky
- 内部填充层是文本（或图像或两者），在按钮上显示。它的风格是，与通常的属性。'Button.label' sticky='nswe'

每个元素都有一本字典的 *元素选项* 会影响该元素的外观。这些选项的名称是所有定期 *Tkinter* 选项如、`anchor`、`justify`、`background` 或 `highlightthickness`。

'anchor''justify''background''highlightthickness'

要获取选项名称的列表，请使用此窗体的方法调用，其中是类的一个实例：

`Sttk.Style`

```
S.element_options(styleName)
```

其结果是一个选项字符串序列，每个前面加一个连字符。继续我们会话之上，哪里的实例：`sttk.Style`

```
>>> d = s.element_options('Button.highlight')
>>> d
```

```
('highlightcolor', 'highlightthickness')
```

若要找出哪些属性与元素选项相关联，请使用此窗体的方法调用：

```
s.lookup(layoutName, optName)
```

继续我们的示例：

```
>>> s.lookup('Button.highlight', 'highlightthickness')
1
>>> s.lookup('Button.highlight', 'highlightcolor')
'#d9d9d9'
>>> print s.element_options('Button.label')
('-compound', '-space', '-text', '-font', '-foreground', '-
underline',
'-width', '-anchor', '-justify', '-wraplength', '-embossed', '-
image',
'-stipple', '-background')
>>> s.lookup('Button.label', 'foreground')
'black'
```

50.2。 *ttk* 样式映射： 动态外观变化

Ttk 小部件可以改变其外观在程序执行期间。例如，当一个小部件是 *禁用*，它将不响应鼠标或键盘操作。通常一个残疾的小部件呈现不同的外观，以便用户可能意识到小部件不会响应鼠标。

一般情况下，每个 *ttk* 小部件具有一组 *状态标志*，你可以使用，使构件变化在执行过程中的外观。每个状态可能设置（打开）或重置（关闭）独立于其他国家。国家和它们的含义：

active	鼠标是目前在小部件内。
alternate	这种状态被留给应用程序使用。
background	在 Windows 或 MacOS 下，小部件位于不是前台窗口一个窗口。
disabled	小部件不会响应用户操作。
focus	小部件当前具有 焦点 。
invalid	小部件的内容不是当前有效的。
pressed	小部件当前被按下（例如，所单击的按钮）。
readonly	小部件将不允许任何用户操作来改变其当前值。例如，一个只读的小部件将不允许编辑其内容。Entry
selected	小部件处于选中状态。例如，checkboxbuttons 和单选按钮，在“打开”状态。

一些国家将为例，更改以响应用户操作的状态。您的程序可以审问，清除，或通过使用[“通用所有 *ttk* 小部件的方法”一节 46](#)中描述的功能设置任何国家。

pressedButton

更改外观的一个小部件的逻辑被绑在它的元素之一。审问或建立动态行为的一个特定的风格，给定的 *ttk* 的实例，使用此方法，在哪里的元素的名称，例如，或。 `s.StylestyleName'Button.label''border'`

```
s.map(styleName, *p, **kw)
```

要确定给定的样式元素的一种选择的动态行为，选项名称作为第二个的位置参数，传递和该方法将返回的状态变化规格列表。

每个状态变化规范是一个序列。这个序列是指当小部件的当前状态匹配所有部件、选项设置为值。每个项目是任一国家的名称，或国家名称前面的“”。若要匹配，小部件必须在所有国家由项目描述，不要跟我“”，并且它必须不是任何从开始的国家“”。 $(s_0, s_1, n) s_i n s_i ! !!$

例如，假设您有一个实例的类，和你这样称呼它：`sttk.Style`

```
changes = s.map('TCheckbutton', 'indicatorcolor')
```

进一步假设返回值是：

```
[('pressed', '#ececec'), ('selected', '#4a6984')]
```

这意味着当 `checkbutton` 处于状态，应设置其选项，颜色，和 `checkbutton` 处于状态时，应将其选项设置为。

```
pressedindicatorcolor '#ececec' selectedindicatorcolor '#4a6984'
```

您还可以通过将一个或多个关键字参数传递给该方法更改元素的动态行为。例如，要得到上面的示例中的行为，请使用此方法调用：`.map()`

```
s.map('TCheckbutton',  
      indicatoron=[('pressed', '#ececec'), ('selected',  
      '#4a6984')])
```

这里是一个更复杂的例子。假设您想要创建自定义按钮样式基于标准的类。我们就叫我们的风格；因为我们的名字都是以“”，它将自动继承标准样式特征。这里是如何设置这个新的样式：`TButtonWild.TButton.TButton`

```
s = ttk.Style()  
s.configure('Wild.TButton',  
            background='black',  
            foreground='white',  
            highlightthickness='20',  
            font=('Helvetica', 18, 'bold'))  
s.map('Wild.TButton',  
      foreground=[('disabled', 'yellow'),  
                  ('pressed', 'red'),  
                  ('active', 'blue')],  
      background=[('disabled', 'magenta'),  
                  ('pressed', '!focus', 'cyan'),  
                  ('active', 'green')],  
      highlightcolor=[('focus', 'green'),  
                      ('!focus', 'red')],
```

```
relief=[('pressed', 'groove'),
        ('!pressed', 'ridge')])
```

- 此按钮最初将在黑色的背景上，使用 20 像素宽重点突出显示上显示白色文本。
- 如果该按钮的状态是，它将洋红色背景上显示黄色文本。'disabled'
- 如果目前正在按按钮，文本会变成红色;提供该按钮的作用不具有焦点，背景将青色。元组是如何使属性依赖于状态的组合的一个例子。
('pressed', '!focus', 'cyan')
- 如果该按钮处于活动状态（根据光标），文本将蓝色的绿色背景。
- 当按钮具有焦点和红色，当它不，重点突出显示将变为绿色。
- 按钮将显示脊救济，它不被按下时，和槽救济，当它被按下。

51. 连接应用程序的逻辑部件

前面的几节谈到了如何安排和配置小部件 —— 应用程序的前面板。

接下来，我们会谈论如何向上部件到执行用户请求的操作的逻辑连接。

- 若要使您的应用程序响应鼠标单击或键盘输入的事件，有两种方法：
 - 某些控件，如按钮有一个属性，它允许您指定一个程序，称为 *处理程序*，当用户单击该控件时将调用。command

使用小部件的事件的顺序是很具体的虽然。用户必须移动鼠标指针到小工具来使用鼠标按钮 1，然后按下鼠标按钮 1，然后释放鼠标按钮 1 仍然在部件上。没有其他一连串事件将“按”一个小部件。ButtonButton

- 有是一个更一般的机制，可以让您的应用程序的反应以及许多更多种类的投入： 新闻或释放任何键盘键或鼠标按钮;移动鼠标到、 围绕、 或从一个小部件;和许多其他事件。与处理程序，在该机制中你写了特定类型的事件发生时将调用的处理程序。这种机制被讨论下[一节 54、“事件”](#)。command
- 很多小部件要求您使用 *控制变量*，连接小部件在一起和你的程序，以便您可以读取和设置属性的窗口小部件的特殊对象。控制变量将在下一节中讨论。

52. 控制变量： 小部件背后的价值观

Tkinter 控制变量是一个特殊的对象，就像一个常规的 Python 变量，它是一个值，例如数字或字符串的容器。

控制变量的一个特质是它可以由若干不同的部件，共享和控制变量可以记住所有窗口小部件的当前共享它。尤其是，这意味着，如果您的程序存储一个值到一个控制变量用其 `.set()` 方法，链接到该控制变量的任何部件会自动更新在屏幕上。 *vccv*

Tkinter 号码使用了控制变量的重要功能，例如：

- Checkbuttons 使用控制变量来保存 checkbutton 的当前状态（或关闭）。
- 一个单一的控制变量由一组共同的单选按钮，可以用于告诉其中一人当前设置。所以，当你设置一个，清除任何其他设置的单选按钮组中，当用户单击一个单选按钮组中时，共享此控制变量是由哪个 **Tkinter** 组单选按钮的机制。
- 控制变量包含若干个应用程序的文本字符串。通常在小部件中显示的文本被链接到一个控制变量。在几个其他控件，也可以使用字符串值控制变量以保存文本，例如 checkbuttons 和单选按钮的标签和内容的小部件。EntryLabel

例如，可以将小部件链接到一个小部件，以便当用户更改条目中的文本，按 Enter 键时，标签会自动更新以显示相同的文本。EntryLabel

要控制变量，请使用这些类的构造函数，具体取决于哪种类型的值，您需要将存储在其中之一：

```
v = tk.DoubleVar() # Holds a float; default value 0.0
v = tk.IntVar()    # Holds an int; default value 0
v = tk.StringVar() # Holds a string; default value ''
```

所有控制变量都具有这两种方法：

.get()

返回该变量的当前值。

.set(*value*)

更改该变量的当前值。当主回路接下来无所事事；如果任何小部件选项拼命给该变量，将更新这些小部件在控制此更新周期上看到 [.update_idletasks\(\)](#) 在 [第 26、“通用构件方法”](#) 的详细信息。

以下是一些意见如何控制变量的使用与特定部件：

Button

您可以设置它的。随时改变了该变量，按钮上的文本将更新以显示新值。这不是必要的除非该按钮的文本实际上正在改变：使用属性，如果是静态的按钮的标签。`textvariableStringVar``text`

Checkbox

通常情况下，会将所需的小部件的选项设置为，并且该变量将设置为1，当启用了 `checkboxbutton` 并且到 0 时，它就处于关闭状态。。然而，你可以选择这两个国家与不同的值和选项，分别。

`variableIntVar``onvalue``offvalue`

您甚至可以使用 `a` 作为 `checkboxbutton` 的变量和字符串的值的供应和。下面是一个示例：`StringVar``offvalue``onvalue`

```
self.spamVar = tk.StringVar()
self.spamCB = tk.Checkbutton(self, text='Spam?',
                             variable=self.spamVar, onvalue='yes', offvalue='no')
```

如果这个 `checkboxbutton` 上，将返回的字符串;如果 `checkboxbutton` 是关闭的那相同的调用将返回字符串。此外，您的程序可以打开 `checkboxbutton` 通过调用。`self.spamVar.get()``'yes'``'no'``.set('yes')`

你也可以到 `checkboxbutton` 的选项。然后，您可以更改该变量使用的方法，`checkboxbutton` 上的文本标签。`textvariableStringVar.set()`

Entry

将其选项设置为。使用该变量的方法来检索当前显示在小部件中的文本。您还可以变量的方法以更改窗口小部件中显示的文本。

`textvariableStringVar.get().set()`

Label

您可以将设置其选项。然后对该变量的方法的任何调用将更改标签上显示的文本。这不是必要的如果该标签的文本是静态的;应用程序运行时不更改的标签时，请使用属性。`textvariableStringVar.set()``text`

Menubutton

如果你想要能够改变菜单按钮上显示的文本，请将其选项设置为使用该变量的方法来更改显示的文本。`textvariableStringVar.set()`

Radiobutton

选项必须设置为控制变量，或 `a`。功能组中的所有单选按钮必须共享相同的控制变量。`variableIntVarStringVar`

设置每个单选按钮的选项组中为不同的值。每当用户设置单选按钮，该变量将被设置为该单选按钮、选项和共享组的所有其他单选将被清除。`valuevalue`

您可能想知道，所处的状态是一组单选按钮中永远不会被设置时控制变量和用户永远不会点击他们吗？每个控制变量的默认值：`对`，`对`，`，`。`。如果有单选按钮之一，将最初设置该单选按钮。如果没有单选按钮选项匹配的变量的值，都将出现单选按钮被清除。`
`0IntVar0.0DoubleVar''StringVarvaluevalue`

如果您想要在您的应用程序执行期间更改单选按钮上的文本标签，请将其选项设置为。`然后您的程序可以通过将新的标签文本传递给该变量的方法更改的文本标签。``textvariableStringVar.set()`

Scale

对于一个规模小部件，将其选项设置为控制变量的任何类，并设置其和选项，将两端的规模的限制值。`variablefrom_to`

例如，您可以使用和设置的比例和。`然后每个用户到小部件会改变变量的值 0 到 100 包容性之间的一些值。``IntVarfrom_=0to=100`

您的程序也可以通过使用该方法对控制变量移动滑块。继续上面的示例中，会将滑块移动到位置四分之三的沿其槽方式。`.set().set(75)`

若要设置一个小部件的值，请使用。`ScalefloatDoubleVar`

您可以使用 `a` 作为控制变量的一个小部件。你将仍然需要提供数字和值，但该构件的数值将被转换为存储在字符串。使用缩放的选项来控制这种转换的精度。`StringVarScalefrom_toStringVardigits`

53. 重点：路由键盘输入

说一个小部件有**聚焦**手段那键盘输入是目前针对该窗口部件。

- 由**焦点遍历**，我们指的是当用户用 tab 键移动小部件到部件将访问的窗口小部件的序列。有关详细信息，请参阅下面这个序列的规则。
- 您可以遍历倒退使用 shift tab 键。
- 小部件打算接受键盘输入和条目或文本小部件当前具有焦点时，如果你键入的任何字符将添加到它的文本。通常的编辑字符如 ← 和 → 将有他们平常的影响。EntryText
- 因为小部件可以包含制表符字符，你必须使用特殊的键序列控制选项卡将过去的文本小部件焦点移动。Text
- 由焦点遍历和当他们具有焦点时，通常会访问大多数其他类型的窗口小部件：
 - Button 小部件可以是“按下”通过按空格键。
 - Checkbutton 小部件可以使用空格键的设置和清除状态之间进行切换。
 - 在小部件，↑ 和 ↓ 键向下滚动向上或向下一行；上一页和下一页的键滚动页；和空格键选择当前行或取消选中它，如果它已经被选择。Listbox
 - 您可以通过按空格键来设置一个小部件。Radiobutton
 - 卧式小部件回应 ← 和 → 键，和垂直部分响应 ↑ 和 ↓。Scale
 - 在小部件中，上一页和下一页的键由来访移动滚动条。↑ 和 ↓ 键会按单位，移动垂直滚动条和 ← 和 → 键会按单位移动水平滚动条。Scrollbar
- 很多小部件提供大纲要求**重点突出**显示用户哪些部件有亮点。这通常是一个薄的黑色框架位于郊外的部件的边框（如果有）。通常没有重点突出显示（具体而言，帧、标签和菜单）的小部件，可以将选项设置为一个非零的值，以使重点突出可见。highlightthickness
- 您还可以更改使用选项重点突出显示的颜色。highlightcolor
- 类、部件和没有通常访问的焦点。然而，你可以设置其选项来让他们列入焦点遍历。您还可以通过将其选项设置为从焦点遍历任何部件。FrameLabelMenutakefocus1takefocus0

在 tab 键遍历窗口小部件的顺序是：

- 是同一个父儿童的小部件，焦点转小部件被创建的顺序相同。
- 为父窗口小部件包含其他窗口小部件（如帧），重点访问父部件第一次（除非其选项），然后它访问孩子小部件，以递归方式，在创建它们的顺序。takefocus0

综上所述：若要设置焦点遍历顺序你的部件，在该命令中创建它们。通过设置其选项为 0，并为那些其默认选项，设置为，如果你想要将它们添加到订单从遍历顺序中删除的小部件。`takefocus01`

上述的描述在 *Tkinter* 输入焦点的默认运作。还有另一种，完全不同的方式来处理它 —— 让去鼠标所到之处的焦点。下[节 26、“通用构件方法”](#)，引用的方法。`.tk_focusFollowsMouse()`

您还可以添加、更改或删除任何键盘上的键功能里面任何小部件通过使用事件绑定的方式。详细信息，请参阅[一节 54、“事件”](#)。

53.1。 集中在 *ttk* 小部件中

如果你创建一个 *ttk* 小部件并不指定其选项，默认情况下，所有的 *ttk* 部件获得焦点除了为、 和。

takefocusFrameLabelLabelFramePanedWindowProgressbarScrollbarSeparator
Sizegrip

54. 事件： 对刺激的反应

*事件*就是发生在您的应用程序 —— 例如，在用户按下某个键或单击或拖动鼠标 —— 对应用程序需要对其作出反应。

窗口小部件通常有大量的内置行为。例如，按钮将响应鼠标单击通过调用其回调。又例如，如果您将焦点移动到条目窗口小部件，然后按一封信，那封信获取添加到小部件的内容。`command`

然而， *Tkinter* 事件绑定功能允许您添加、 更改或删除行为。

第一，一些定义：

- *事件*是您的应用程序需要了解一些发生。
- *事件处理程序*是在您的应用程序获取事件发生时调用的函数。
- 当您的应用程序设置到小部件发生事件时调用的事件处理程序时，我们叫它 *绑定*。

54. 1. 级别的绑定

你可以将一个处理程序绑定到事件在任何三个层次：

1. 实例绑定： 您可以将事件绑定到一个特定的小部件。例如，可能将上一页键在画布部件绑定到一个处理程序，使画布上向上滚动一页。若要绑定事件的一个小部件，该窗口部件调用[.bind\(\)](#)方法（见[第 26, “通用构件方法”](#)）。

例如，假设您有命名的画布部件和您想要在画布上绘制橙色的 blob，每当用户单击鼠标按钮 2（中间的按钮）。若要实现此行为：

```
self.canv
```

```
self.canv.bind('<Button-2>', self.__drawOrangeBlob)
```

第一个参数是一个*序列描述符*，告诉 *Tkinter*，每当鼠标中键下山时，它将调用*事件处理程序*命名。（见[节 54.6“写您的处理程序：Event 类”](#)，下面，概述如何编写处理程序，如）。请注意您省略括号后的处理程序的名称，这样，Python 会通过引用中的处理程序，而不是试图马上打电话给它。

```
self.__drawOrangeBlob.__drawOrangeBlob()
```

2. 类具有约束力： 您可以将事件绑定到类的所有小部件。例如，您可能设置了所有小部件来响应鼠标单击按钮通过英语和日语标签之间来回更改。若要将事件绑定到类的所有小部件，调用[.bind_class\(\)](#)方法在任何部件上（见上文[第 26, “通用构件方法”](#)，）。Button

例如，假设您有好几幅油画，和你想要设置鼠标按钮 2 绘制它们之中的任何一个橙色的 blob。不必要求他们每一个人，你可以设置他们完了一个调用这样的事情：[.bind\(\)](#)

```
self.bind_class('Canvas', '<Button-2>',  
               self.__drawOrangeBlob)
```

3. 应用程序绑定： 您可以设置一个绑定，以便调用处理程序不论什么小部件具有焦点或鼠标下的某一事件。例如，可能将 printscrn 键绑定到的应用程序，所有的小部件，以便它打印屏幕无论什么构件获取该密钥。若要将绑定应用程序级别的事件，在任何部件上调用[.bind_all\(\)](#)方法（见[第 26, “通用构件方法”](#)）。

这里是如何可能绑定 printscrn 键，其“键名”是：'Print'

```
self.bind_all('<Key-Print>', self.__printScreen)
```

54. 2。 事件序列

Tkinter 具有强大而普遍的方法，允许您定义的事件，具体和一般的你想要确切绑定到处理程序。

一般情况下，事件序列是一个字符串，包含一个或多个 *事件模式*。每个事件模式描述了一件事情可以发生。如果序列中有一个以上的事件模式，只有当所有的模式发生在那相同的序列时，将会调用处理。

事件模式的一般形式为：

```
<[modifier...]... type[-detail]>
```

- 整个模式括内。<...>
 - *事件类型*描述事件，如关键的按键或鼠标单击的一般的种类。请参阅[一节 54. 3, “事件类型”](#)。
 - 您可以添加可选项目之前要指定期间其他按键沮丧的 shift 或控制键的组合的类型或鼠标点击。[节 54. 4, “事件修饰符”](#) *modifier*
 - 您可以添加可选的项目来描述你正在寻找什么键或鼠标按钮。对于鼠标按钮，这是为按钮 1，按钮 2，2 或 3 为按钮 3 1。 *detail*
 - 常见的安装按钮 1 的左侧和右侧，按钮 3，但左撇子可以交换这些位置。
 - 对于键盘上的键，这是关键的字符（为单字符键喜欢或密钥）或键的名称；所有键名的列表，请参阅[一节 54. 5, “键名称”](#)。
- A*

这里是一些例子来给你风味的事件模式：

<Button-1>	用户按下第一个鼠标按钮。
<KeyPress-H>	用户按 H 键。
<Control-Shift-KeyPress-H>	用户按下控制换档 H.

54.3。 事件类型

全套的事件类型是相当大，但很多人并不常用。这里的大多数那些你将需要：

类型	名称	描述
36	Activate	一个小部件从静止状态到正在积极改变。这是指选择一个小部件，如按钮更改从处于非活动状态（灰显）为积极的变化。state
4	Button	用户按下鼠标按钮之一。部分指定了哪个按钮。鼠标滚轮支持在 Linux 中，使用（向上滚动）下，（向下滚动）。Linux 下，您的处理程序为鼠标轮绑定将区分之间向上滚动和向下滚动通过检查领域的实例;见 节 54.6“写您的处理程序：Event 类” 。detailButton-4Button-5.numEvent
5	ButtonRelease	用户松懈的鼠标按钮。这可能是更好的选择，在大多数情况下比事件，因为如果用户不小心按下按钮，他们可以移动鼠标关闭要避免设置关闭事件的构件。Button
22	Configure	用户更改大小的一个小部件，例如通过拖动角或窗口的一侧。
37	Deactivate	一个小部件改变从正在积极向正在处于非活动状态。这是指在一个小部件，如从活动更改为非活动状态（灰显）单选按钮选项中更改。state
17	Destroy	一个小部件被销毁。
7	Enter	用户将鼠标指针移到一个小部件的可见部分。（这是不同于输入的关键，这也是一个事件，他的名字实际上是关键）。KeyPress' return'
12	Expose	每当至少部分的应用程序或窗口小部件有被其他窗口覆盖了后变为可见时，将发生此事件。
9	FocusIn	一个小部件有输入的焦点（见 节 53“焦点：路由键盘输入” 为输入焦点是一般性介绍。）这种情况可能发生在响应用户事件（如使用 tab 键将焦点移动小部件之间）或以编程方式（例如，程序都将调用上一个小部件）。.focus_set()

类型	名称	描述
10	FocusOut	输入的焦点被搬出一个小部件。如，用户可导致此事件，或您的程序可以导致它。FocusIn
2	KeyPress	用户按键盘上的键。部分指定哪个键。此关键字可能被缩写。detailKey
3	KeyRelease	用户松懈的一把钥匙。
8	Leave	用户移动鼠标指针移出一个小部件。
19	Map	一个小部件是被映射，就是使应用程序中可见。这会发生，例如，当您调用小部件的方法。 <code>.grid()</code>
6	Motion	用户移动鼠标指针完全内一个小部件。
38	MouseWheel	用户移动鼠标滚轮向上或向下。目前，此绑定的工作原理对 Windows 和 MacOS，但不是在 Linux 下。Windows 和 MacOS，请参阅中的实例字段的讨论 节 54.6“写您的处理程序：Event 类” 。对于 Linux，看到上面下的注。 <code>.deltaEventButton</code>
18	Unmap	一个小部件被取消映射并不再可见。发生这种情况，例如，当你使用的部件的方法。 <code>.grid_remove()</code>
15	Visibility	发生时至少部分的应用程序窗口在屏幕上可见。

54. 4。 事件修饰符

您可以在事件序列中使用的修饰符名称包括：

Alt	当用户按住 alt 键，则该属性值为 true。
Any	此修饰符概括了一种事件类型。例如，事件模式适用于任何键的紧迫。’<Any-KeyPress>’
Control	当用户按住 control 键，则该属性值为 true。
Double	指定两个事件在时间附近一起发生。例如，描述了两个印刷机在快速连续的 1 按钮。<Double-Button-1>
Lock	当用户已按 shift 锁定，则该属性值为 true。
Shift	当用户按住 shift 键，则该属性值为 true。
Triple	像，但指定快速连续的三个事件。Double

您可以使用短形式的事件。这里有一些例子：

- ’<1>’ 是相同。’<Button-1>’
- ’x’ 是相同。’<KeyPress-x>’

请注意，你可以离开了封闭的大多数单字符按键，但你不能为空格字符（他的名字是）或少-比（）字符（他的名字是）。’<…>’ ’<space>’ ’<less>’

54.5。 键名称

为事件模式的细节部分或事件指定您要绑定哪个键。（请参阅修饰符，上面，如果你想要得到所有按键或键释放）。KeyPressKeyReleaseAny

下表显示了几种不同的方式对名称键。见[节 54.6“写您的处理程序：Event 类”](#)，下面，对对象的详细信息，其属性将描述这些相同的方式键。Event

- 列显示“键符号”，键的字符串名称。这对应于的对象的属性。`.keysym.keysymEvent`
- 列是“键的代码”。这标识按了哪个键，但代码并不反映各种修饰符来像 shift 键和控制键和数码锁定键的状态。这样，例如，两者有相同的键代码。`.keycodeaA`
- 列显示一个数字代码相当于键符号。与不同的是，这些代码是不同的不同的改性剂。例如，在数字小键盘（键符号）和（键符号）数字小键盘上的向下箭头上的数字 2 有相同的键代码（88），但不同的值（65433 和 65458，分别）。`.keysym_num.keycodeKP_2KP_Down.keysym_num`
- “Key”列显示的文本，你通常会发现在物理键，如选项卡。

有很多更多键命名为国际字符集。此表显示只有“拉丁语-1”为常见的美国类型 101 键键盘设置。当前受支持的设置，请参阅[Tk keysym 值的手册页](#)。

. keysym	. keycode	. keysym_num	键
Alt_L	64	65513	左手按住 alt 键
Alt_R	113	65514	右边的 alt 键
BackSpace	22	65288	退格键
Cancel	110	65387	休息
Caps_Lock	66	65549	大写锁定键
Control_L	37	65507	左手控制键
Control_R	109	65508	右控制键
Delete	107	65535	删除
Down	104	65364	↓
End	103	65367	结束
Escape	9	65307	按 esc 键

. keysym	. keycode	. keysym_num	关键
Execute	111	65378	SysReq
F1	67	65470	功能键 F1
F2	68	65471	功能键 F2
F _i	66+i	65469+i	功能键 F _我
F12	96	65481	功能键 F12
Home	97	65360	首页
Insert	106	65379	插入
Left	100	65361	←
Linefeed	54	106	换行符（控制 J）
KP_0	90	65438	键盘上的 0
KP_1	87	65436	键盘上的 1
KP_2	88	65433	键盘上的键 2
KP_3	89	65435	3 键盘上的键
KP_4	83	65430	键盘上的键 4
KP_5	84	65437	键盘上的键 5
KP_6	85	65432	6 键盘上的键
KP_7	79	65429	7 键盘上的键
KP_8	80	65431	键盘上的 8
KP_9	81	65434	在键盘上 9
KP_Add	86	65451	+ 键盘上的键
KP_Begin	84	65437	键盘上的键中心键（5 相同的密 钥）
KP_Decimal	91	65439	键盘上的小数点（.）.
KP_Delete	91	65439	键盘上的删除
KP_Divide	112	65455	/键盘上键
KP_Down	88	65433	键盘上的键 ↓

. keysym	. keycode	. keysym_num	关键
KP_End	87	65436	键盘上的键结束
KP_Enter	108	65421	输入键盘上的键
KP_Home	79	65429	主键盘上的键
KP_Insert	90	65438	插入键盘上的键
KP_Left	83	65430	键盘上的 ←
KP_Multiply	63	65450	键盘上的键 ×
KP_Next	89	65435	键盘上的键下一页
KP_Prior	81	65434	键盘上的上一页
KP_Right	85	65432	键盘上的键 →
KP_Subtract	82	65453	-键盘上键
KP_Up	80	65431	在键盘上 ↑
Next	105	65366	下一页
Num_Lock	77	65407	锁定
Pause	110	65299	暂停
Print	111	65377	Printscrn 键
Prior	99	65365	上一页
Return	36	65293	输入密钥（控制-M）。名称是指鼠标相关的事件，不按键;请参阅 第54、“事件”Enter
Right	102	65363	→
Scroll_Lock	78	65300	鼠标
Shift_L	50	65505	左 shift 键
Shift_R	62	65506	右 shift 键
Tab	23	65289	Tab 键
Up	98	65362	↑

54.6。 写您的处理程序： 类 Event

上面的部分告诉你如何描述你想要处理，哪些的事件以及如何将它们绑定。现在让我们谈谈写作当事件实际发生时将调用的处理程序。

该处理程序将传递一个对象，描述了发生了什么事。该处理程序可以是一个函数或方法。这里是正则函数的调用序列：Event

```
def handlerName(event):
```

作为一种方法：

```
def handlerName(self, event):
```

传递给处理程序的对象的属性说明如下。其中有些属性始终设置，但一些只为特定类型的事件。Event

.char	如果事件与相关或为密钥产生一个规则的 ASCII 字符，则此字符串将设置为该字符。（像删除的特殊键，请参阅属性，下面）。KeyPressKeyRelease.keysym
.delta	对于事件，此属性包含一个整数，其标志是积极向上滚动，负向下滚动。根据 Windows，此值将多 120;例如，向上一步，滚动 120 手段和-240 手段滚动两个步骤。根据苹果 Mac，它将有 1，所以 1 就意味着向上一步，滚动和-2 滚动下来两个步骤。Linux 的鼠标滚轮支持，请参阅 一节 54.3, “事件类型” 中的事件绑定的说明。MouseWheelButton
.height	如果事件是，此属性设置为该构件新的高度，以像素为单位。Configure
.keycode	为或事件，此属性设置为一个标识的键的数字代码。然而，它并不标识的那把钥匙上的字符产生，所以，“”和“”具有相同的值。这一领域的可能值，请参阅 一节 54.5, “键名称” 。KeyPressKeyReleaseX.keyCode
.keysym	或事件涉及一个特殊键，此属性设置为键的字符串的名称，例如，上一页键。名称的完整列表，请参阅 一节 54.5, “键名称” 。KeyPressKeyRelease'Prior'.keysym
.keysym_num	为或事件，这设置为数值字段的版本。对于定期产生单个字符的键，此字段设置为整数值的键的 ASCII 码。特殊的键，请参阅 节 54.5, “键名称” 。KeyPressKeyRelease.keysym

<code>. num</code>	如果事件与相关的鼠标按钮，则将此属性设置为按钮序号（1、 2 或 3）。为鼠标滚轮支持下 Linux、 绑定和事件;当鼠标滚轮向上滚动，此字段将为 4 或 5 时向下滚动。 Button-4Button-5
<code>. serial</code>	每次在服务器处理客户端请求递增整数序列号。您可以使用值来找到确切的时间序列的事件： 那些与较低的值更早发生。 <code>. serial</code>
<code>. state</code>	一个整数，描述所有修饰符键的状态。请参阅表的修饰符面具下面为此值的解释。
<code>. time</code>	此属性设置为一个整数，它没有绝对的意义，但是是递增每一毫秒。这允许您的应用程序来确定，例如，两个鼠标点击之间的时间长度。
<code>. type</code>	描述事件的类型的数字代码。此代码的解释，请参阅 一节 54.3，“事件类型” 。
<code>. widget</code>	总是设置到小部件引起的事件。例如，如果事件是发生在画布的鼠标单击，此属性将实际的小部件。 <code>Canvas</code>
<code>. width</code>	如果事件是，此属性设置为该构件新的宽度，以像素为单位。 <code>Configure</code>
<code>. x</code>	鼠标事件，相对于左上角的小部件的时间坐标。 x
<code>. y</code>	鼠标事件，相对于左上角的小部件的时间坐标。 y
<code>. x_root</code>	鼠标事件，相对于屏幕左上角的时间坐标。 x
<code>. y_root</code>	鼠标事件，相对于屏幕左上角的时间坐标。 y

使用这些面具来测试要看到什么修改键的值的位，并在事件过程中按下按钮
：`. state`

面具	修饰符
0x0001	转变。
0x0002	大写锁定。
0x0004	控制。
0x0008	左手按住 Alt。
0x0010	Num Lock 键。

面具	修饰符
0x0080	右边的 alt 键。
0x0100	鼠标按钮 1。
0x0200	鼠标按钮 2。
0x0400	鼠标按钮 3。

这里是一个事件处理程序的示例。以上[节 54.1, “具有约束力的水平”](#)，是一个示例，演示如何将鼠标的点击 2 绑定到处理程序称为命名在画布上。这里是该处理程序：`self.canvself.__drawOrangeBlob()`

```
def __drawOrangeBlob(self, event):
    '''Draws an orange blob in self.canv where the mouse is.
    '''
    r = 5    # Blob radius
    self.canv.create_oval(event.x-r, event.y-r,
                           event.x+r, event.y+r, fill='orange')
```

此处理程序被调用时，当前的鼠标位置是。方法绘制一个圆其边界框是在那个位置上的平方和为中心，两边长度。`(event.x, event.y).create_oval()2*r`

54. 7. 额外的参数欺骗

有时你想要将其他参数传递给除了事件的处理程序。

这里是一个例子。假设您的应用程序具有的十个 `checkbuttons` 的小部件都存储在列表中，按中的 `checkboxbutton` 数字索引数组。`self.cbListrange(10)`

进一步假设您想要编写一个处理程序命名为在所有十个这些 `checkbuttons` 事件。该处理程序可以获取实际的部件所指的传中，对象的属性触发它但它如何去那 `checkboxbutton` 索引中发现吗？`__cbHandler<Button-1>Checkboxbutton.widgetEventsself.cbList`

我会很高兴写我们处理额外的参数，`checkboxbutton` 号，就像这样：

```
def __cbHandler(self, event, cbNumber):
```

但是，事件处理程序传递只有一个参数，该事件。所以我们不能使用上述的功能由于中的参数个数不匹配。

幸运的是，Python 的能力为函数参数提供默认值给出我们的方式了。看一看这段代码：

```
def __createWidgets(self):
    ...
    self.cbList = []    # Create the checkboxbutton list
    for i in range(10):
        cb = tk.Checkbutton(self, ...)
        self.cbList.append(cb)
        cb.grid( row=1, column=i)

        def handler(event, self=self, i=i):
            return self.__cbHandler(event, i)
        cb.bind('<Button-1>', handler)
    ...
def __cbHandler(self, event, cbNumber):
    ...
```

这些行定义了一个新的函数，预计三个参数。第一个参数是传递给所有事件处理程序的对象，第二和第三个参数将被设置为它们的默认值——我们需要通过它的额外参数。`handlerEvent`

这种技术可以扩展以提供任意数量的附加参数处理程序。

54.8。 虚拟事件

你可以创建你自己新的种类的事件称为*虚拟事件*。你可以给他们你想要只要它封闭的双成对的任何名称。<<...>>

例如，假设您想要创建一个称为，由鼠标左键 3 或暂停键触发的新事件。若要创建此事件，请在任何部件上调用此方法：<<panic>>*w*

```
w.event_add('<<panic>>', '<Button-3>',  
            '<KeyPress-Pause>')
```

然后您可以在任何情况下使用序列。例如，如果您使用此调用：'<<panic>>'

```
w.bind('<<panic>>', h)
```

任何在小部件中暂停按键或鼠标按钮 3 将触发该处理程序。*wh*

见[.event_add\(\)](#)，[.event_delete\(\)](#)和[.event_info\(\)](#)下[节 26、“通用构件方法”](#)有关创建和管理虚拟事件详细信息。

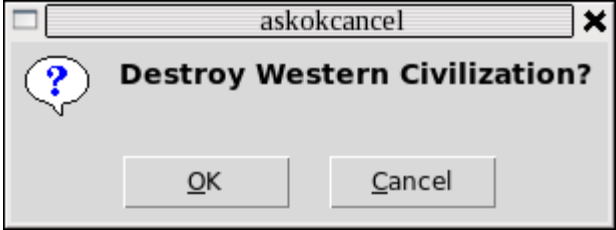
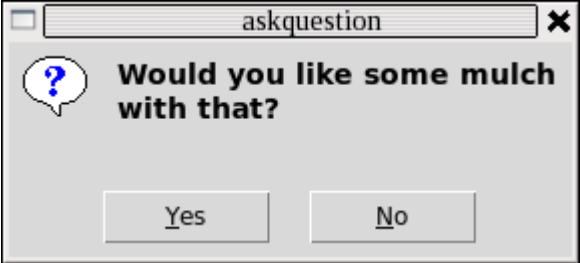
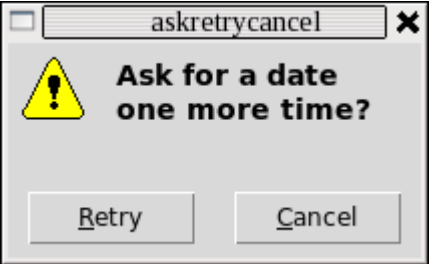
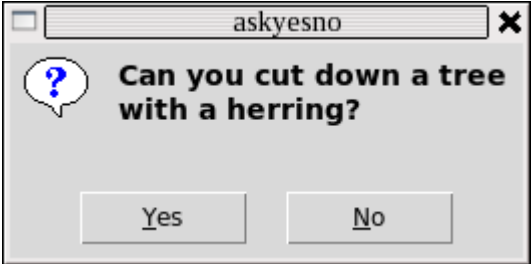
55. 弹出对话框

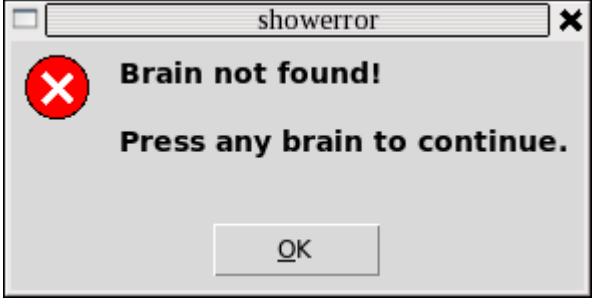

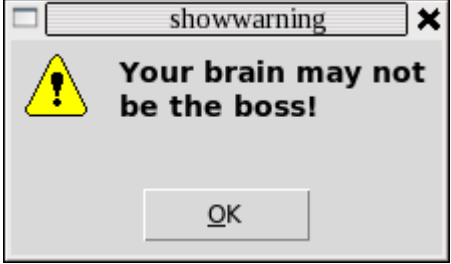
Tkinter 提供可以为您创建弹出式对话框窗口的三个模块：

- [节 55.1， “tkMessageBox 对话框模块”](#)，提供各式各样的常见弹出窗口为简单的任务。
- [节 55.2， “tkFileDialog 模块”](#)，允许用户浏览文件。
- [节 55.3， “tkColorChooser 模块”](#)，允许用户选择一种颜色。

55. 1. 对话框模块 tkMessageBox

一旦你导入模块，您可以创建任何这些七个常见类型的弹出式菜单中从该表调用函数。tkMessageBox

 A screenshot of a Tkinter dialog box titled 'askokcancel'. It has a question mark icon and the text 'Destroy Western Civilization?'. There are 'OK' and 'Cancel' buttons at the bottom.	<pre>. askokcancel(<i>title</i>, <i>message</i>, <i>options</i>)</pre>
 A screenshot of a Tkinter dialog box titled 'askquestion'. It has a question mark icon and the text 'Would you like some mulch with that?'. There are 'Yes' and 'No' buttons at the bottom.	<pre>. askquestion(<i>title</i>, <i>message</i>, <i>options</i>)</pre>
 A screenshot of a Tkinter dialog box titled 'askretrycancel'. It has a yellow warning triangle icon and the text 'Ask for a date one more time?'. There are 'Retry' and 'Cancel' buttons at the bottom.	<pre>. askretrycancel(<i>title</i>, <i>message</i>, <i>options</i>)</pre>
 A screenshot of a Tkinter dialog box titled 'askyesno'. It has a question mark icon and the text 'Can you cut down a tree with a herring?'. There are 'Yes' and 'No' buttons at the bottom.	<pre>. askyesno(<i>title</i>, <i>message</i>, <i>options</i>)</pre>

	<pre><code>.showerror(<i>title</i>, <i>message</i>, <i>options</i>)</code></pre>
	<pre><code>.showinfo(<i>title</i>, <i>message</i>, <i>options</i>)</code></pre>
	<pre><code>.showwarning(<i>title</i>, <i>message</i>, <i>options</i>)</code></pre>

在每种情况下，是要在顶部的窗口装饰中显示的字符串。该参数是一个字符串，它出现在正文中的弹出窗口；在此字符串内行将破碎在换行符（`\n`）。

参数可以是任何这些选择。*option*

default

哪个按钮应该是默认的选择？如果不指定此选项，第一个按钮（“确定”，“是”或“重试”）将成为默认选项。

若要指定哪个按钮是默认选择，使用，在哪里中定义这些常量之一：`default=CtkMessageBoxCANCELIGNOREOKNORETRYYES` 或。

icon

选择哪个图标将出现在弹出窗口中。使用这些常量之一在哪里在中定义的窗体参数：`icon=IltkMessageBoxERRORINFOQUESTIONWARNING` 或。

parent

如果不指定此选项，弹出窗口显示您根窗口的上方。为了使一些子窗口 *W* 的上方出现弹出窗口，请使用参数。parent=*W*

每个“”弹出函数返回一个值，取决于哪个按钮用户被迫删除弹出窗口。ask...

- askokcancel 和所有返回值：“OK”“Yes”选择为“否”或“取消”选择。
askretrycancelaskyesnoboolTrueFalse
- askquestion 返回为“是”或“否”。u' yes' u' no'

55.2. 模块 tkinterFileDialog

该模块提供两个不同的弹出窗口，您可以使用若要使用户能够查找现有文件或创建新的文件。tkFileDialog

. askopenfilename(*option=value*, ...)

意为例，用户想要选择一个现有的文件。如果用户选择不存在的文件，一个弹出式窗口会出现通知他们所选的文件不存在。

. asksaveasfilename(*option=value*, ...)

用于用户要创建一个新文件或替换现有的文件的情况。如果用户选择一个现有的文件，弹出窗口将显示通知，该文件已经存在，并问如果他们真的想要替换它。

这两个函数的参数是相同的：

defaulttextextension=*s*

默认文件扩展名，以句点 (.) 开头的字符串。如果用户的答复包含句点，这种说法没有任何影响。有没有时间的情况下，它被追加到用户的答复。'.'

例如，如果你提供的参数和用户输入，返回的文件名是会的。
defaulttextextension='.jpg' gojiro gojiro.jpg

filetypes=[(*label₁*, *pattern₁*), (*label₂*, *pattern₂*), ...]

在文件列表中出现两个元素的元组包含文件类型名称和模式，将选择的内容列表。在下面的屏幕图片，注意到下拉式菜单标记“类型的文件:”。您提供的参数将会填充此下拉列表。每个是一个文件类型名称（如示例中的“PNG”）和一种模式，选择给定类型的文件 (“(*.png)”中的示例)。filetypes*pattern*

initialdir=*D*

要最初显示的目录的路径名称。默认目录是当前工作目录。

initialfile=*F*

最初在显示文件名称“文件的名称:”字段中，如果有的话。

parent=*W*

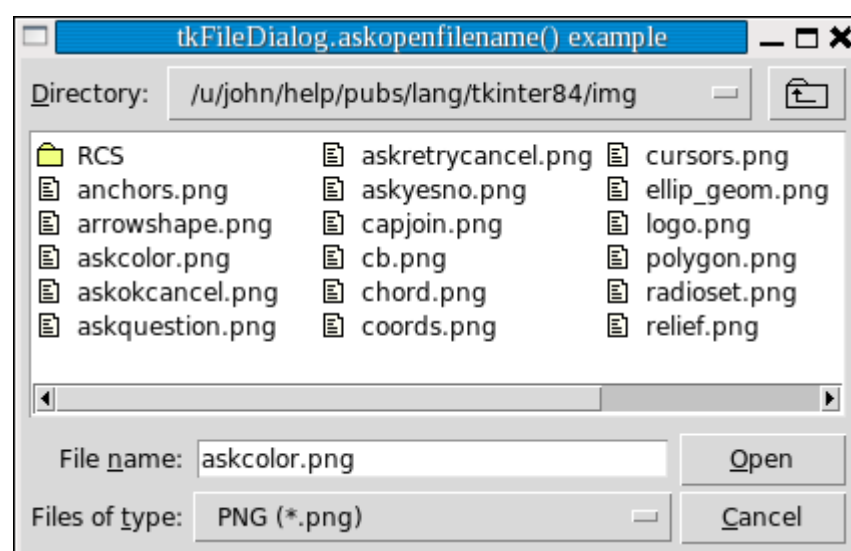
若要使弹出窗口出现在一些窗口，请提供此参数。默认行为是弹出窗口将出现在您的应用程序的根窗口的上方。//

`title=T`

如果指定了，是要显示为弹出窗口标题的字符串。*T*

如果用户选择一个文件，则返回的值是文件的所选的完整路径名称。如果用户使用 **取消** 按钮，则函数将返回一个空字符串。

下面是一个示例：



55.3. 模块 tkColorChooser

为给您的应用程序的用户一个弹出式窗口，他们可以使用选择一种颜色，导入模块，调用此函数：tkColorChooser

```
result = tkColorChooser.askcolor(color, option=value, ...)
```

论据如下：

color

要显示的初始颜色。默认初始颜色为浅灰色。

title=text

指定将显示在弹出窗口的标题区。默认的标题是“色”。*text*

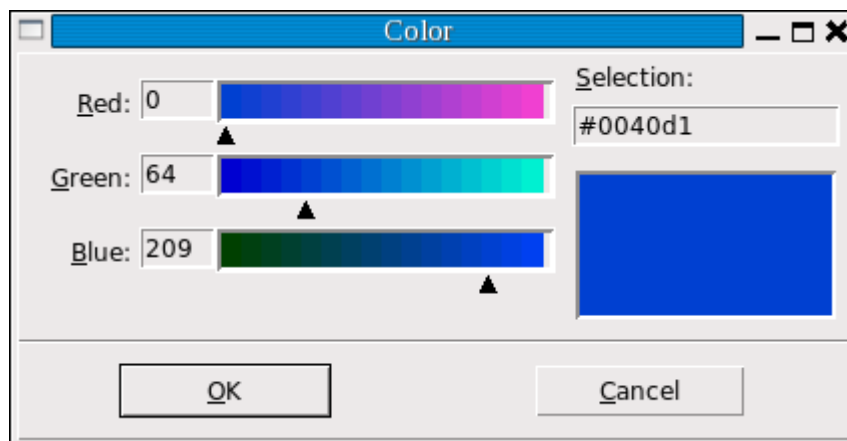
parent=//

使弹出显示在窗口的上方。默认行为是它出现在你的根窗口的上方。*//*

如果用户单击 **确定** 按钮在弹出的菜单，返回的值将一个元组，哪里元组包含红色、绿色和蓝色值在 [0255] 范围分别，是作为定期 *Tkinter* 颜色对象选定的颜色。(*triple*, *color*) *triple*(*R*, *G*, *B*) *color*

如果用户单击 **取消**，此函数将返回。(None, None)

这里是什么弹出窗口看起来像作者的系统上：



John W. Shipman

Comments welcome: tcc-doc@nmt.edu

最后更新： 2013 年-12-31 17:59

网址： <http://www.nmt.edu/tcc/help/pubs/tkinter/web/tkColorChooser.html>

