

# **Tkinter 8.5 参考：一个 Python 的图形界面**

原文作者：Joh W.Shipman

译者：William

原文发布时间：2013-06-24 12:46

---

## 摘要

本参考介绍了用 **Python** 编程语言中的 **Tkinter** 套件来创建用户图形界面。本参考涵盖了 **ttk** 主题控件。

本参考现可在线浏览<sup>1</sup>同时另有PDF 文档<sup>2</sup>。如有意见请发邮件到**tcc-doc@nmt.edu**。  
如对翻译有任何意见请发邮件至**william0victor@gmail.com**

---

<sup>1</sup><http://www.nmt.edu/tcc/help/pubs/tkinter/>

<sup>2</sup><http://www.nmt.edu/tcc/help/pubs/tkinter/tkinter.pdf>

## 目录

<b>1</b>	<b>Python 的一个跨平台用户图形界面</b>	<b>1</b>
<b>2</b>	<b>一个小程序</b>	<b>2</b>
<b>3</b>	<b>解释</b>	<b>3</b>
<b>4</b>	<b>布局管理</b>	<b>4</b>
4.1	.grid() 方法 . . . . .	4
4.2	其它 grid 管理方法 . . . . .	5
4.3	配置列和行的尺寸 . . . . .	6
4.4	让根窗口可重组 . . . . .	6
<b>5</b>	<b>标准属性</b>	<b>8</b>
5.1	尺寸 . . . . .	8
5.2	坐标系 . . . . .	8
5.3	颜色 . . . . .	8

## 第 1 章 Python 的一个跨平台用户图形界面

*Tkinter* 是 Python 的一个 GUI ( graphical user interface ) 组件。本文档适用于运行在 Linux 操作系统的 X Window 系统中的 Python2.7 和 Tkinter8.5。你的版本或稍有差异。

相关参考：

- Fredrik Lundh, who wrote *Tkinter*, has two versions of his *An Introduction to Tkinter*: a more complete 1999 version<sup>3</sup> and a 2005 version<sup>4</sup> that presents a few newer features.
- *Python 2.7 quick reference*<sup>5</sup>: general information about the Python language
- For an example of a sizeable working application (around 1000 lines of code), see *huey: A color and font selection tool*<sup>6</sup>.

我们将从 *Tkinter* 可见的部分开始：创建控件并布局在屏幕上。稍后我们将探讨如何将应用程序前端的面板关联到后端逻辑。

---

<sup>3</sup><http://www.pythonware.com/library/tkinter/introduction/>

<sup>4</sup><http://effbot.org/tkinterbook/>

<sup>5</sup><http://www.nmt.edu/tcc/help/pubs/python/>

<sup>6</sup><http://www.nmt.edu/tcc/help/lang/python/examples/huey/>

## 第2章一个小程序

下面是一个只含有一个推出按钮的 *Tkinter* 小程序

```
1  #!/usr/bin/env python 1
2  import Tkinter as tk 2
3  class Application(tk.Frame): 3
4      def __init__(self, master=None): 4
5          tk.Frame.__init__(self, master) 4
6          self.grid() 5
7          self.createWidgets()
8      def createWidgets(self):
9          self.quitButton = tk.Button(self, text='Quit',
10                                     command=self.quit) 6
11          self.quitButton.grid() 7
12 app = Application() 8
13 app.master.title('Sample application') 9
14 app.mainloop() 10
```

1. 假如你的系统中已正确安装 Python，该行将会使脚本自动执行。
2. 该行将 *Tkinter* 模块导入到程序的命名空间，但重命名为 *tk*。
3. 你的程序的类必须从 *Tkinter* 的 *Frame* 类继承。
4. 调用父类 *Frame* 的构造函数。
5. 必须让程序真正显示在屏幕上。
6. 创建一个按钮，标记为 “Quit”。
7. 将按钮放入程序中
8. 通过实例化 *Application* 类启动主程序
9. 这个方法将窗口的 *title* 设为 "Sample application"。
10. 启动程序的主程，等待鼠标和键盘事件。

## 第 3 章解释

在我们继续下面的内容前，让我们来解释一些常用的名词。

### **window**

在不同的语境中这个词有不同的意思，但通常它是指你电脑显示屏中的某个矩形区域。

### **top-level window**

一个独立存在于你屏幕中的窗口，它将为你的系统桌面管理器装饰框架和控制器。你可以在桌面上四处移动它。通常，你也可以调整它的尺寸除非程序禁止了。

### **widget**

这个词通常指组成程序用户图形界面的组件。例如：按钮、单选框、文本域、框架以及文本标签。

### **frame**

在 *Tkinter* 中，Frame 控件是组成复杂布局的基本单元。它指一个能够包含其它控件的矩形区域。

### **child,parent**

当任意一个控件被创建时，父级 - 子级关系就已经建立。例如，当你将一个文本标签放入一个框架中，框架就是文本标签的父级。

## 第4章布局管理

接下来我们将讲讲控件，组成程序图形界面的“积木块”。怎样布置控件到窗口中？尽管在 *Tkinter* 中有三种不同“图形管理器”，但是笔者特别愿意用 `.grid()` 图形管理器来布局。这个管理将所有的窗口或框架当做一个表 --- 有行和列的网格。

- 单元格是一行和一列的交叉区域。
- 每列最宽的单元格的宽度是该列的列宽。
- 每行最高的单元格的高度是该行的行高。
- 控件不可能完全充满单元格的所有空间，你可以对单元格进行指定。你既可以不调整控件外的多余控件，也可以伸展水平方位或垂直方位来使控件填满单元格。
- 你可以将多个单元格合并生成一个大单元格。

当你创建了一个控件，除非你在图形管理器中注册了它否则它将不显示。因此，构造和放置控件的两个步骤进行如下：

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

*Constructor* 是如按钮、框架等控件的类，*Parent* 是将被构造的子类控件的父类控件。所有的控件都有 `.grid()` 方法，你可以使用它来告诉图形管理器怎么放置控件。

### 第4.1节.grid() 方法

将一个控件 *w* 显示到程序窗口中：

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

该方法在图形管理器中注册了一个控件 *w*---如果不这么做，控件将只存在内部不会显示出来。见表一“`.grid()` 图形管理器参数”：

column	控件放置的列数，从 0 开始计算。默认值是 0。
columnspan	通常一个控件只占据网格中的一个单元格。然而，你可以选取行中的多个单元格，并在 <code>columnspan</code> 选项中设置单元格的数量来整合他们到一个大单元格。例如， <code>w.grid(row=0,column=2,columnspan=3)</code> ，例中将会把 <i>w</i> 控件放置在一个横跨 0 行 2, 3, 4 列的一个单元格中。
in_	注册 <i>w</i> 作为某个控件如 <i>w<sub>2</sub></i> 的子类，用法 <code>in_=w<sub>2</sub></code> 。当 <i>w</i> 被创建时，新的 <i>w<sub>2</sub></i> 控件必须作为 <i>parent</i> 控件的子类来使用。
ipadx	内部 x padding。这个维度是增加控件内部左边和右边。
ipady	内部 y padding。这个维度是增加控件内部顶部和底部。
padx	外部 x padding。这个维度是增加控件外部左边和右边。
pady	外部 y padding。这个维度是增加控件上部和下部。
row	你想把控件插入的行数，从 0 计数。默认是下一个更高的未被占用的行。
rowspan	通常一个控件只占据网格的一个单元格。你可以选取列内的多个单元格，然后，给选中的单元格设置 <code>rowspan</code> 选项。本选项和 <code>columnspan</code> 选项结合使用来抓取一块单元格。例子， <code>w.grid(row=3,column=2,rowspan=4,columnspan=5)</code> 例中将 <i>w</i> 微件放入一个由 3-6 列 2-6 行组成的区域中。
sticky	本项决定怎样分配单元格中的微件所占空间之外的空间。见下。

- 
- 如果未设置 `sticky` 属性，默认会将微件在单元格中居中放置。
  - 你可以使用 `sticky=NE` (右上), `SE` (右下), `SW` (左下), 或者 `NW` (左上) 来布局微件到单元格的四角。
  - 你可以使用 `sticky=N` (中上), `E` (中右), `S` (中下), 或者 `W` (中左) 来布局微件到一边的相对中间。
  - 使用 `sticky=N+S` 垂直扩展微件但让它水平居中。
  - 使用 `sticky=E+W` 水平扩展微件但让它垂直居中。
  - 使用 `sticky=N+E+S+W` 在水平和垂直方位来扩展微件填充单元格。
  - 其它的组合也可使用。例如, `sticky=N+S+W` 将垂直扩展微件并放置微件在相对东(左)边。

## 第4.2节其它 grid 管理方法

这些 grid 相关的方法在所有控件上都有定义：

### **w.grid\_bbox ( column=None, row=None, col2=None, row2=None )**

返回一个四元组描述微件 `w` 中一些或所有 grid 系统的边界框。前两个数字返回左上角区域的 `x` 和 `y` 坐标, 后两个数字是宽和高。

如果你传递行和列变量, 返回的限定框描述所在行和列的单元格的区域。如果你也传递了 `col2` 和 `row2` 参数, 返回的限定框描述包含从行 `column` 到 `col2`, 列 `row` 到 `row2` 的区域。

例如, `w.grid_bbox(0,0,1,1)` 返回四个单元格的限定框, 不是一个。

### **w.grid\_forget()**

本方法使微件 `w` 从屏幕上消失。它还存在, 只是不可见。你可以使用 `.grid()` 使它再次显示, 但是它将不会记住它的 grid 选项。

### **w.grid\_info()**

返回一个键为 `w` 微件选项名字的字典, 以及这些选项相应的值。

### **w.grid\_location (x,y)**

赋予关联的包含的微件一个坐标, 本方法返回一个数组 (`col,row`) 描述 `w` 微件的网格系统的单元格包含的屏幕坐标。

### **w.grid\_propagate()**

通常, 所有微件传送他们的尺寸, 意味着他们调节来适应内容。然而, 有时你想约束一个微件到确定的尺寸, 忽略它内容的尺寸, 这样做, 调用 `w.grid_propagate(0)` 限制 `w` 微件的尺寸。

### **w.grid\_remove()**

本方法类似 `.grid_forget()`, 但是它的 grid 选项会记住, 所以如果你再 `.grid()` 它, 它将会使用相同的 grid 配置选项。

### **w.grid\_size()**

分别在 `w` 微件 grid 系统中返回一个包含行数和列数的二元组。

### **w.grid\_slaves ( row=None, column=None )**

返回由微件 `w` 管理的微件的目录。如果没有提供参数, 你将会获得所有被管理的微件的目录。使用 `row=` 参数只选择一行微件, 或者使用 `column=` 参数只选择一行的微件。



## 第 4.3 节配置列和行的尺寸

除非你采取确切的措施，网格列内的微件宽将会等于它的最大宽度，并且网格行内的控件高将会是最高的单元格的高。控件里的 `sticky` 属性只控制控件被放置的地方如果所在单元格未被完全填满。

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

w 微件的网格层中，配置 column N 以便所给选项有所给的值。对于选项，请看下表。

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

w 微件的网格层中，配置 row N 以便所给的选项有所给的值。对于选项，请看下表。

以下是用来配置 column 和 row 尺寸的选项。

minsize	以像素为最小单位，列和行的最小尺寸。如果列内或行内没有内容，它将不会显示即使你使用了这一选项
pad	若干像素将会被加入所给的列或行，及以上的列或行中最大的单元格。
weight	为了使一列或一行可伸展，当重新分配额外的空间时，使用此选项并提供一个值，赋予该列或行相对权重。例如，如果一个微件 w 包含一个网格层，这些行将会分配 3/4 的额外空间到第一列及 1/4 到第二列： <div><div>1</div><div>2</div><div><code>w.columnconfigure(0, weight=3)</code> <code>w.columnconfigure(1, weight=1)</code></div></div> 如果没有使用此项，列或行将不会伸展。

## 第 4.4 节让根窗口可重组

你想让用户能调整你的整个程序窗口大小，并将空出的控件分配给内部的控件吗？只需普通的几个操作就能实现。

这需要用到行和列尺寸管理的方法，在第 4.3 小节已经提到，“配置列和行大小” (p.7)，来使你的 *Application* 控件的网格可伸缩。然而，那仅仅是不够的。

思考下第二章讨论的小程序，“一个小程序” (p.2)，这个程序只包含一个退出按钮。如果你运行此程序，并调整窗口大小，按钮会保持同样的尺寸，并居中于窗口。

以下是小程序内 `__createWidgets()` 方法的替代版本。在这个版本中，退出按钮总是填满可用空间。

```
1 def createWidgets(self):  
2     top=self.winfo_toplevel() 1  
3     top.rowconfigure(0,weight=1) 2  
4     top.columnconfigure(0,weight=1) 3  
5     self.rowconfigure(0,weight=1) 4  
6     self.columnconfigure(0,weight=1) 5  
7     self.quit=Button(self, text="Quit", command=self.quit )  
8     self.quit.grid(row=0, column=0, sticky=N+S+E+W) 6
```

1. “top level window” 是屏幕上最顶层的窗口。但是，这个窗口不是 *Application* 的窗口——它是 *Application* 实例的父级。要获取顶层窗口，在程序中的任何控件上调用 `winfo_toplevel()` 方法。参看章节，“通用控件方法” (p.97)。
2. 本行代码使 top level window 的 0 列网格可伸展。
3. 本行代码使顶层窗口的 0 列网格可伸展。
4. 使 0 行的 *Application* 控件的网格可伸展。
5. 使 0 列的 *Application* 控件的网格可伸展
6. 参数 `sticky=tk.N+tk.S+tk.E+tk.W` 使按钮展开填满它所占的单元格

还必须作出一个变化。在构造函数中，如下显示的改变第二行：

---

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

sticky=tk.N+tk.S+tk.E+tk.W 参数对 self.grid() 是必要的, 因此 *Application* 控件将会展开填充它所在 top-level window 网格的单元格。

## 第 5 章标准属性

在我们看控件之前，让我们来看看他们的一些共同指定的属性 ---如大小，颜色以及字体。

- 每个控件都有一些属性选项来影响它的显示和行为，如颜色，大小，文本标签等。
- 当调用控件构造器时，你可以使用如 `text='PANIC'` 或者 `height=20` 等关键字参数。
- 当你创建了一个控件后，后面你可以使用控件的 `.config()` 方法来改变参数。你也可以使用控件的 `.cget()` 方法来获取当前参数的设置。了解更多类似方法请看第 26 章“通用控件方法”(P.97)

### 第 5.1 节尺寸

控件的长，宽及其它的尺寸可以用许多不同的单位。

- 如果你想设置一个整数尺寸，它将默认以像素为单位
- 你可以在设定尺寸时在数字后加入如下字符来指定单位：

表 3. 尺寸单位

c	Centimeters
i	Inches
m	Millimeters
p	Printer's points (about 1/72")

### 第 5.2 节坐标系

当前的大多数显示系统中，坐标系是在左上角，x 轴向右走，y 轴向下走：



最基本的单位是像素，最左上的像素的坐标为 (0,0)。你所指定的整数坐标通常默认以像素为单位，但是坐标都可能指定数值；请见第 5.1 节“尺寸”(P.9)

### 第 5.3 节颜色

*Tkinter* 中一般有两种方法来指定颜色

- 你可以通过 16 进制字符设置红、绿、蓝颜色的比例来设置颜色：

#rgb	Four bits per color
#rrggbb	Eight bits per color
#rrrgggbbb	Twelve bits per color

例如，“#fff”是白色，“#000000”是黑色，“#000fff000”是纯绿色，“#00ffff”是青色（绿色加蓝色）

- 
- 另外你也可以使用内置的标准颜色名。white 白色，black 黑色，red 红色，green 绿色，blue 蓝色，cyan 青色，yellow 黄色，magenta 品红都能用。其它颜色名也许也能用，要看本地安装环境。