# Tkinter 布局管理器

pack、grid 和 place 均用于管理同在一个父组件下的所有组件的布局，

其中：

pack 是按添加顺序排列组件

grid 是按行/列形式排列组件

place 则允许程序员指定组件的大小和位置

## Tkinter 布局管理器（一）：pack

### 何时使用 pack 管理器？

对比 grid 管理器，pack 更适用于少量组件的排列，但它在使用上更加简单（就像我们前边所有的例子中，展示一个组件我们一般都直接使用 .pack()，多简单~）。

如果你需要创建相对复杂的布局结构，那么建议是使用多个**框架（Frame）结构构成**，或者使用 grid 管理器实现。

注意：不要在同一个父组件中混合使用 pack 和 grid，因为 Tkinter 会很认真地在那儿计算到底先使用那个布局管理器......**以至于你等了半个小时，Tkinter 还在那儿纠结不出结果**！

### 用法

我们常常会遇到的一个情况是将一个组件放到一个容器组件中，并填充整个父组件。这儿给大家举个例子，我们生成一个 Listbox 组件并将它填充到 root 窗口中：

import tkinter as tk

root = tk.Tk()

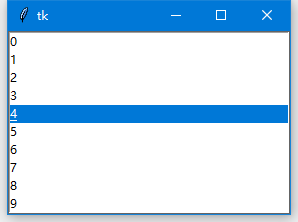
listbox = tk.Listbox(root)

listbox.pack(fill="both", expand=True)

for i in range(10):

        listbox.insert("end", str(i))

root.mainloop()



其中，fill 选项是告诉窗口管理器该组件将填充整个分配给它的空间，"both" 表示同时横向和纵向扩展，"x" 表示横向，"y" 表示纵向；expand 选项是告诉窗口管理器将父组件的额外空间也填满。

默认下，pack 是将添加的组件依次纵向排列：

import tkinter as tk

root = tk.Tk()

tk.Label(root, text="Red", bg="red", fg="white").pack(fill="x")

tk.Label(root, text="Green", bg="green", fg="black").pack(fill="x")

tk.Label(root, text="Blue", bg="blue", fg="white").pack(fill="x")

root.mainloop()



如果想要组件横向挨个儿排放，你可以使用 side 选项：

import tkinter as tk

root = tk.Tk()

tk.Label(root, text="Red", bg="red", fg="white").pack(side="left")

tk.Label(root, text="Green", bg="green", fg="black").pack(side="left")

tk.Label(root, text="Blue", bg="blue", fg="white").pack(side="left")

root.mainloop()



### 方法

注：下边所有方法适用于所有组件

pack(\*\*options)

-- 下方表格详细列举了各个选项的具体含义和用法：

|  |  |
| --- | --- |
| 选项 | 含义 |
| anchor | 1. 控制组件在 pack 分配的空间中的位置 |
| 2. "n", "ne", "e", "se", "s", "sw", "w", "nw", 或者 "center" 来定位（ewsn 代表东西南北，上北下南左西右东） |
| 3. 默认值是 "center" |
| expand | 1. 指定是否填充父组件的额外空间 |
| 2. 默认值是 False |
| fill | 1. 指定填充 pack 分配的空间 |
| 2. 默认值是 NONE，表示保持子组件的原始尺寸 |
| 3. 还可以使用的值有："x"（水平填充），"y"（垂直填充）和 "both"（水平和垂直填充） |
| in\_ | 1. 将该组件放到该选项指定的组件中 |
| 2. 指定的组件必须是该组件的父组件 |
| ipadx | 指定水平方向上的内边距 |
| ipady | 指定垂直方向上的内边距 |
| padx | 指定水平方向上的外边距 |
| pady | 指定垂直方向上的外边距 |
| side | 1. 指定组件的放置位置 |
| 2. 默认值是 "top" |
| 3. 还可以设置的值有："left"，"bottom"，"right" |

选项 含义

anchor

1. 控制组件在 pack 分配的空间中的位置

2. "n", "ne", "e", "se", "s", "sw", "w", "nw", 或者 "center" 来定位（ewsn 代表东西南北，上北下南左西右东）

3. 默认值是 "center"

expand

1. 指定是否填充父组件的额外空间

2. 默认值是 False

fill

1. 指定填充 pack 分配的空间

2. 默认值是 NONE，表示保持子组件的原始尺寸

3. 还可以使用的值有："x"（水平填充），"y"（垂直填充）和 "both"（水平和垂直填充）

in\_

1. 将该组件放到该选项指定的组件中

2. 指定的组件必须是该组件的父组件

ipadx 指定水平方向上的内边距

ipady 指定垂直方向上的内边距

padx 指定水平方向上的外边距

pady 指定垂直方向上的外边距

side

1. 指定组件的放置位置

2. 默认值是 "top"

3. 还可以设置的值有："left"，"bottom"，"right"

pack\_configure(\*\*options)

-- 跟 pack() 一样

pack\_forget()

-- 将组件从屏幕中“删除”

-- 并没有销毁该组件，只是看不到了

-- 可以通过 pack 或其他布局管理器显示已“删除”的组件

pack\_info()

-- 以字典的形式返回当前 pack 的选项

pack\_propagate(flag)

-- 如果开启，父组件会自动调节尺寸以容纳所有子组件

-- 默认值是开启（flag = True）

-- 该方法仅适用于父组件

pack\_slaves()

-- 以列表的形式返回该组件的所有子组件

-- 该方法仅适用于父组件

### pack()方法的实际使用

下面的示例是在窗口内创建4个窗体，在每一个窗体内创建三个按钮。使用了不同的参数创建这些窗体与按钮。

1. #pack()方法使用
2. from tkinter import \*
3. #主窗口
4. win = Tk()
5. #第一个窗体
6. frame1 = Frame (win, relief=RAISED, borderwidth=2)
7. frame1 .pack(side=TOP, fill=BOTH, ipadx=13, ipady=13, expand=0)
8. Button(frame1,text="Button 1") .pack(side=LEFT, padx=13, pady=13)
9. Button(frame1, text="Button 2") .pack(side=LEFT, padx=13, pady=13)
10. Button(frame1, text="Button 3") .pack (side=LEFT, padx=13, pady=13)
11. #第二个窗体
12. frame2 = Frame (win, relief=RAISED, borderwidth=2)
13. frame2 . pack (side=BOTTOM, fill=NONE, ipadx="1c", ipady="1c", expand=1)
14. Button (frame2, text="Button 4") .pack (side=RIGHT, padx="1c", pady="1c")
15. Button (frame2,text="Button 5") .pack (side=RIGHT, padx="1c", pady="1c")
16. Button (frame2,text="Button 6") .pack (side=RIGHT, padx="1c", pady="1c")
17. #第三个窗体
18. frame3 = Frame (win, relief=RAISED, borderwidth=2)
19. frame3. pack (side=LEFT, fill=X, ipadx="0.1i", ipady="0.1i", expand=1)
20. Button (frame3, text="Button 7") .pack(side=TOP, padx="0.1i", pady="0.1i")
21. Button (frame3, text="Button 8") .pack(side=TOP, padx="0.1i", pady="0.1i")
22. Button(frame3, text="Button 9") .pack(side=TOP, padx="0.1i", pady="0.1i")
23. #第四个窗体
24. frame4 = Frame (win, relief=RAISED, borderwidth=2)
25. frame4. pack (side=RIGHT, fill=Y, ipadx="13p", ipady="13p", expand=1)
26. Button(frame4, text="Button 13") . pack (side=BOTTOM, padx="13p",
27. pady="13p")
28. Button (frame4, text="Button 11") .pack (side=BOTTOM, padx="13p" ,
29. pady="13p")
30. Button (frame4, text="Button 12") .pack (side=BOTTOM, padx="13p",
31. pady="13p")
32. #开始窗口的事件循环
33. win . mainloop()

保存为 .pyw 文件后，直接双击运行该文件，结果如图 1 所示。

#### 程序运行结果 图1：程序运行结果

#### 示例代码解析

上述示例代码分析如下：

* 第 6 行：创建第一个 Frame 控件，以作为窗体。此窗体的外形突起，边框厚度为 2 像素。
* 第 7 行：此窗体在窗口的顶端（side=TOP），当窗口改变大小时，窗体本应会占满整个窗口的剩余空间（fill=BOTH），但因设置 expand=0，所以窗体维持不变。控件与窗体边界之间的水平距离是 13 像素，垂直距离是 13 像素。
* 第 8～13 行：在第一个窗体内创建三个按钮。这三个按钮从窗体的左边开始排列（side=LEFT），控件之间的水平距离是13像素，垂直距离是 13 像素。
* 第 13 行：创建第二个 Frame 控件，以作为窗体。此窗体的外形突起，边框厚度为 2 像素。
* 第 14 行：此窗体在窗口的底端（side=BOTTOM），当窗口改变大小时，窗体不会占满整个窗口的剩余空间（fill=NONE）。控件与窗体边界之间的水平距离是 1 厘米，垂直距离是 1 厘米。
* 第 15～17 行：在第一个窗体内创建三个按钮。这三个按钮从窗体的右边开始排列（side=RIGHT），控件之间的水平距离是1厘米，垂直距离是 1 厘米。
* 第 20 行：创建第三个 Frame 控件，以作为窗体。此窗体的外形突起，边框厚度为 2 像素。
* 第 21 行：此窗体在窗口的左边（side=LEFT），当窗口改变大小时，窗体会占满整个窗口的剩余水平空间（fill=X）。控件与窗体边界之间的水平距离是 0.1 英寸，垂直距离是 0.1 英寸。
* 第 22～24 行：在第一个窗体内创建三个按钮。这三个按钮从窗体的顶端开始排列（side=TOP)，控件之间的水平距离是 0.1 英寸，垂直距离是 0.1 英寸。
* 第 27 行：创建第四个 Frame 控件，以作为窗体。此窗体的外形突起，边框厚度为 2 像素。
* 第 28 行：此窗体在窗口的右边（side=RIGHT），当窗口改变大小时，窗体会占满整个窗口的剩余垂直空间（fill=Y）。控件与窗体边界之间的水平距离是 13 点（1点等于1/72英寸），垂直距离是 13 点。
* 第 29～31 行：在第一个窗体内创建三个按钮。这三个按钮从窗体的底端开始排列（side=BOTTOM），控件之间的水平距离是 13 点，垂直距离是 13 点。

## Tkinter 布局管理器（二）：grid

展开

Tkinter 布局管理器之grid

pack、grid 和 place 均用于管理同在一个父组件下的所有组件的布局，其中：

pack 是按添加顺序排列组件

grid 是按行/列形式排列组件

place 则允许程序员指定组件的大小和位置

### 何时使用 grid 管理器？

grid 管理器可以说是 Tkinter 这三个布局管理器中最灵活多变的。如果你只希望学习使用一个布局管理器，那么 grid 绝对是首选。当你在设计对话框的时候，使用 gird 尤其便捷。如果你此前一直在用 pack 构造窗口布局，那么学习完 grid 你会悔恨当初为啥不早学它。使用一个 grid 就可以简单的实现你用很多个框架和 pack 搭建起来的效果。

**注意：不要在同一个父组件中混合使用 pack 和 grid，因为 Tkinter 会很认真地在那儿计算到底先使用那个布局管理器......以至于你等了半个小时，Tkinter 还在那儿纠结不出结果！**

### 用法

使用 grid 排列组件，只需告诉它你想要将组件放置的位置（行/列，row 选项指定行，cloumn 选项指定列）。此外，你并不用提前指出网格（grid 分布给组件的位置称为网格）的尺寸，因为管理器会自动计算。

import tkinter as tk

root = tk.Tk()

# column 默认值是 0

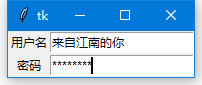
tk.Label(root, text="用户名").grid(row=0)

tk.Label(root, text="密码").grid(row=1)

tk.Entry(root).grid(row=0, column=1)

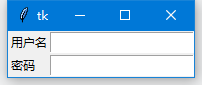
tk.Entry(root, show="\*").grid(row=1, column=1)

root.mainloop()



默认情况下组件会居中显示在对应的网格里，你可以使用 sticky 选项来修改这一特性。该选项可以使用的值有 "e"，"w"，"s"，"n"（ewsn 分别表示东西南北，即上北下南左西右东）以及它们的组合。因此，我们可以通过 sticky = "w" 使得 Label 左对齐：

tk.Label(root, text="用户名").grid(row=0, sticky="w")

tk.Label(root, text="密码").grid(row=1, sticky="w")

有时候你可能需要用几个网格来放置一个组件，可以做到吗？当然可以，你只需要指定 rowspan 和 columnspan 就可以实现跨行和跨列的功能：

import tkinter as tk

root = tk.Tk()

# column 默认值是 0

tk.Label(root, text="用户名").grid(row=0, sticky="w")

tk.Label(root, text="密码").grid(row=1, sticky="w")

tk.Entry(root).grid(row=0, column=1)

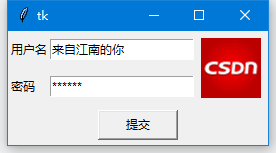
tk.Entry(root, show="\*").grid(row=1, column=1)

photo = tk.PhotoImage(file="logo.gif")

tk.Label(root, image=photo).grid(row=0, column=2, rowspan=2, padx=5, pady=5) #rowspan=2 跨两行

tk.Button(text="提交", width=10).grid(row=2, columnspan=3, pady=5) #columnspan=3 跨三列

root.mainloop()



### 方法

注：下边所有方法适用于所有组件

grid(\*\*options)

-- 下方表格详细列举了各个选项的具体含义和用法：

|  |  |
| --- | --- |
| 选项 | 含义 |
| column | 1. 指定组件插入的列（0 表示第 1 列） |
| 2. 默认值是 0 |
| columnspan | 指定用多少列（跨列）显示该组件 |
| in\_ | 1. 将该组件放到该选项指定的组件中 |
| 2. 指定的组件必须是该组件的父组件 |
| ipadx | 指定水平方向上的内边距 |
| ipady | 指定垂直方向上的内边距 |
| padx | 指定水平方向上的外边距 |
| pady | 指定垂直方向上的外边距 |
| row | 指定组件插入的行（0 表示第 1 行） |
| rowspan | 指定用多少行（跨行）显示该组件 |
| sticky | 1. 控制组件在 grid 分配的空间中的位置 |
| 2. 可以使用 "n", "e", "s", "w" 以及它们的组合来定位（ewsn代表东西南北，上北下南左西右东） |
| 3. 使用加号（+）表示拉长填充，例如 "n" + "s" 表示将组件垂直拉长填充网格，"n" + "s" + "w" + "e" 表示填充整个网格 |
| 4. 不指定该值则居中显示 |

选项 含义

Column

1. 指定组件插入的列（0 表示第 1 列）

2. 默认值是 0

columnspan 指定用多少列（跨列）显示该组件

in\_

1. 将该组件放到该选项指定的组件中

2. 指定的组件必须是该组件的父组件

ipadx 指定水平方向上的内边距

ipady 指定垂直方向上的内边距

padx 指定水平方向上的外边距

pady 指定垂直方向上的外边距

row 指定组件插入的行（0 表示第 1 行）

rowspan 指定用多少行（跨行）显示该组件

sticky

1. 控制组件在 grid 分配的空间中的位置

2. 可以使用 "n", "e", "s", "w" 以及它们的组合来定位（ewsn代表东西南北，上北下南左西右东）

3. 使用加号（+）表示拉长填充，例如 "n" + "s" 表示将组件垂直拉长填充网格，"n" + "s" + "w" + "e" 表示填充整个网格

4. 不指定该值则居中显示

grid\_bbox(column=None, row=None, col2=None, row2=None)

-- 返回一个 4 元组描述该组件所在的限定矩形-- 如果指定 column 和 cow 参数，则返回该位置（column, cow）的组件的限定矩形描述

-- 如果指定 4 个参数，则返回从（column, cow）到（col2, row2）所有组件的限定矩形描述

-- 例如 grid\_bbox(0, 0, 1, 1) 返回的是 4 个组件所在的限定矩形

grid\_columnconfigure(index, \*\*options)

-- 设置列的属性

-- 注意：设置的是该组件所拥有的 grid 的列

-- 可以设置的选项及含义如下：

选项 含义

minsize 指定该列的最小宽度

pad 指定该列中最大网格的水平边距

weight 1. 指定列与列之间的相对距离

2. 默认值是 0

3. 这个你比较难理解，小甲鱼还是详细解说下：初创建窗口的时候，grid 会自动根据组件的尺寸分配窗口的尺寸，当你拉伸窗口的尺寸时就会有空白显示出来。这个选项正是指定列与列之间是否填充空白，默认是不填充的。另外，该选项的值是指定填充空白的倍数，例如 weight = 2 的列会比 weight = 1 的列填充多一倍的空白。所以需要平均填充的话，只需要所有的列都设置 weight = 1 即可。

grid\_configure(\*\*options)

-- 跟 grid() 一样

grid\_forget()

-- 将组件从屏幕中“删除”

-- 并没有销毁该组件，只是看不到了

-- 可以通过 grid 或其他布局管理器显示已“删除”的组件，但该组件所在网格的选项设置不会恢复

grid\_info()

-- 以字典的形式返回当前 grid 的选项

grid\_location(x, y)

-- 返回位于（或接近）给定坐标（x, y）的网格位置

-- 返回值是一个 2 元组表示网格对应的（列，行）

grid\_propagate(flag)

-- 如果开启，父组件会自动调节尺寸以容纳所有子组件

-- 默认值是开启（flag = True）

-- 该方法仅适用于父组件

grid\_remove()

-- 跟 grid\_forget() 一样，但恢复的时候会记住该组件所在网格的选项设置

grid\_rowconfigure(index, \*\*options)

-- 设置行的属性

-- 注意：设置的是该组件所拥有的 grid 的行

-- 可以设置的选项及含义如下：

选项 含义

minsize 指定该行的最小高度

pad 指定该列中最大网格的垂直边距

weight 1. 指定行与行之间的相对距离

2. 默认值是 0

3. 这个你比较难理解，不懂可以参考上边 grid\_columnconfigure() 的详细解释

grid\_size()

-- 返回该组件所拥有的 grid 的尺寸

-- 返回值是一个 2 元组，表示（列, 行）分别的网格数

grid\_slaves(row=None, column=None)

-- 以列表的形式返回该组件的所有子组件

-- 该方法仅适用于父组件

### grid()方法应用实例

下面的示例是使用 grid() 方法创建一个 5×5 的按钮数组。

1. #grid()方法
2. from tkinter import \*
3. #主窗口
4. win = Tk()
5. #创建窗体
6. frame = Frame (win, relief=RAISED, borderwidth=2)
7. frame.pack (side=TOP, fill=BOTH, ipadx=5, ipady=5, expand=1)
8. #创建按钮数组
9. **for** i **in** range (5):
10. **for** j **in** range (5):
11. Button (frame, text=" (" + str(i) + ","+ str(j)+ ")") .grid(row=i,column=j )
12. #开始窗口的事件循环
13. win. mainloop()

保存 demo.pyw 文件后，直接双击运行该文件，结果如图 1 所示：

#### 程序运行结果 图1：程序运行结果

#### 示例代码分析如下：

* 第 6 行：创建一个 Frame 控件，以作为窗体。此窗体的外形突起，边框厚度为 2 像素。
* 第 7 行：此窗体在窗口的顶端（side=TOP），当窗口改变大小时，窗体会占满整个窗口的剩余空间（fill=BOTH）。控件与窗体边界之间的水平距离是 5 像素，垂直距离是 5 像素。
* 第 13～12 行：创建一个按钮数组，按钮上的文字是（row, column）。str(i) 是将数字类型的变量 i 转换为字符串类型。str(j) 是将数字类型的变量 j 转换为字符串类型。

## 设置行，列尺寸

一个插件，其 grid 网格中的列宽，等于该插件里最宽 cell 的宽度，grid网格中的行高，等于该插件里最高 cell 的高度。 sticky 只控制插件的布放位置，插件并不会因此而填充整个 cell 。

如果你想自行调整行与列的尺寸，可以使用以下函数来设置：

**w.columnconfigure(N, option=value, …)**

在w 插件的grid布局内，可以为第N列配置相应的option选项，详细的option选项见下表。

**w.rowconfigure(N, option=value, …)**

在w 插件的grid布局内，可以为第N行配置相应的option选项，详细的option选项见下表。

**表2. .grid() 结构管理器的行列配置选项**

|  |  |
| --- | --- |
| option | 解释 |
| minsize | 列或行的最小像素尺寸。如果该行或列中没有任何插件，仍旧不会显示任何东西。 |
| pad | 给行或列中最大的 cell 额外分配对应数值的像素。 |
| weight | 给这个参数设置一定的数值（权重），就能够使该列或行以此权重，在多余的空间中伸缩，平铺。比如，w 插件使用了 grid 布局，并且有这两行代码： w.columnconfigure(0, weight=3) w.columnconfigure(1, weight=1) ，这会将多余的 3/4 空间分配给第一列，其余的1/4 空间分配给第二列。如果没有使用此option，就不会伸缩行或列。 |

## Tkinter 布局管理器（三）：place

pack、grid 和 place 均用于管理同在一个父组件下的所有组件的布局，其中：

pack 是按添加顺序排列组件

grid 是按行/列形式排列组件

place 则允许程序员指定组件的大小和位置

### 何时使用 place 管理器？

通常情况下不建议使用 place 布局管理器，因为对比起 pack 和 grid，place 要做更多的工作。不过纯在即合理，place 在一些特殊的情况下可以发挥妙用。请看下边例子。

### 用法

将子组件显示在父组件的正中间：

import tkinter as tk

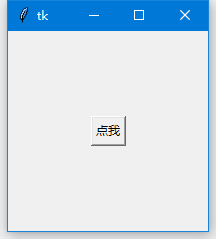
root = tk.Tk()

def callback():

print("正中靶心")

tk.Button(root, text="点我", command=callback).place(relx=0.5, rely=0.5, anchor="center")

root.mainloop()



在某种情况下，或许你希望一个组件可以覆盖另一个组件，那么 place 又可以派上用场了。下边例子我们演示用 Button 覆盖 Label 组件：

import tkinter as tk

root = tk.Tk()

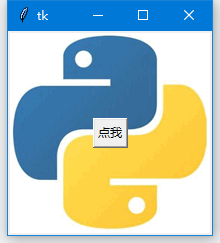
photo = tk.PhotoImage(file = "Python.gif")

tk.Label(root, image = photo).pack()

def callback():

print("正中靶心")

tk.Button(root, text="点我", command=callback).place(relx=0.5, rely=0.5, anchor="center")

root.mainloop()

应该不难看出，relx 和 rely 选项指定的是相对于父组件的位置，范围是 00 ~ 1.0，因此 0.5 表示位于正中间。那么 relwidth 和 relheight 选项则是指定相对于父组件的尺寸：

import tkinter as tk

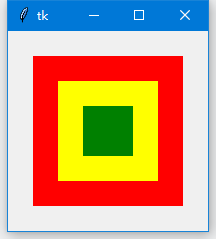
root = tk.Tk()

tk.Label(root, bg="red").place(relx=0.5, rely=0.5, relheight=0.75, relwidth=0.75, anchor="center")

tk.Label(root, bg="yellow").place(relx=0.5, rely=0.5, relheight=0.5, relwidth=0.5, anchor="center")

tk.Label(root, bg="green").place(relx=0.5, rely=0.5, relheight=0.25, relwidth=0.25, anchor="center")

root.mainloop()



上图无论你如何拉伸改变窗口，三个 Label 的尺寸均会跟着改变（神同步~）。x 和 y 选项用于设置偏移（像素），如果同时设置 relx（rely）和 x（y），那 place 将优先计算 relx 和 rely，然后再实现 x 和 y 指定的偏移值。

### 方法

注：下边所有方法适用于所有组件

place(\*\*options)

-- 下方表格详细列举了各个选项的具体含义和用法：

|  |  |
| --- | --- |
| 选项 | 含义 |
| **anchor** | 1. 控制组件在 place 分配的空间中的位置 |
| 2. "n", "ne", "e", "se", "s", "sw", "w", "nw", 或 "center" 来定位（ewsn代表东西南北，上北下南左西右东） |
| 3. 默认值是 "nw" |
| **bordermode** | 1. 指定边框模式（"inside" 或 "outside"） |
| 2. 默认值是 "inside" |
| **height** | 指定该组件的高度（像素） |
| **in\_** | 1. 将该组件放到该选项指定的组件中 |
| 2. 指定的组件必须是该组件的父组件 |
| **relheight** | 1. 指定该组件相对于父组件的高度 |
| 2. 取值范围 0.0 ~ 1.0 |
| **relwidth** | 1. 指定该组件相对于父组件的宽度 |
| 2. 取值范围 0.0 ~ 1.0 |
| **relx** | 1. 指定该组件相对于父组件的水平位置 |
| 2. 取值范围 0.0 ~ 1.0 |
| **rely** | 1. 指定该组件相对于父组件的垂直位置 |
| 2. 取值范围 0.0 ~ 1.0 |
| **width** | 指定该组件的宽度（像素） |
| **x** | 1. 指定该组件的水平偏移位置（像素） |
| 2. 如同时指定了 relx 选项，优先实现 relx 选项 |
| **y** | 1. 指定该组件的垂直偏移位置（像素） |
| 2. 如同时指定了 rely 选项，优先实现 rely 选项 |



选项 含义

anchor 1. 控制组件在 place 分配的空间中的位置

2. "n", "ne", "e", "se", "s", "sw", "w", "nw", 或 "center" 来定位（ewsn代表东西南北，上北下南左西右东）

3. 默认值是 "nw"

bordermode 1. 指定边框模式（"inside" 或 "outside"）

2. 默认值是 "inside"

height 指定该组件的高度（像素）

in\_ 1. 将该组件放到该选项指定的组件中

2. 指定的组件必须是该组件的父组件

relheight 1. 指定该组件相对于父组件的高度

2. 取值范围 0.0 ~ 1.0

relwidth 1. 指定该组件相对于父组件的宽度

2. 取值范围 0.0 ~ 1.0

relx 1. 指定该组件相对于父组件的水平位置

2. 取值范围 0.0 ~ 1.0

rely 1. 指定该组件相对于父组件的垂直位置

2. 取值范围 0.0 ~ 1.0

width 指定该组件的宽度（像素）

x 1. 指定该组件的水平偏移位置（像素）

2. 如同时指定了 relx 选项，优先实现 relx 选项

y 1. 指定该组件的垂直偏移位置（像素）

2. 如同时指定了 rely 选项，优先实现 rely 选项

place\_configure(\*\*options)

-- 跟 place() 一样

place\_forget()

-- 将组件从屏幕中“删除”

-- 并没有销毁该组件，只是看不到了

-- 可以通过 place 或其他布局管理器显示已“删除”的组件

place\_info()

-- 以字典的形式返回当前 place 的选项

place\_slaves()

-- 以列表的形式返回该组件的所有子组件

-- 该方法仅适用于父组件

slaves()

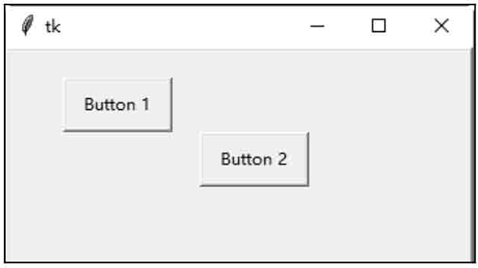
-- 跟 place\_slaves() 一样

### place()方法应用实例

下面的示例是使用 place() 方法创建两个按钮。第一个按钮的位置在距离窗体左上角的 (40,40) 坐标处，第二个按钮的位置在距离窗体左上角的 (140, 80) 坐标处。按钮的宽度均为 80 像素，高度均为 40 像素。

1. #place()方法
2. from tkinter import \*
3. #主窗口
4. win = Tk()
5. #创建窗体
6. frame = Frame (win, relief=RAISED, borderwidth=2, width=400, height=300)
7. frame. pack (side=TOP, fill=BOTH,ipadx=5, ipady=5, expand=1)
8. #第一个按钮的位置在距离窗体左上角的(40，40)坐标处
9. button1 = Button ( frame, text="Button 1")
10. button1.place (x=40,y=40, anchor=W, width=80, height=40)
11. #第二个按钮的位置在距离窗体左.上角的(140，80) 坐标处
12. button2 = Button (frame, text="Button 2")
13. button2 .place(x=140,y=80, anchor=W, width=80, height=40)
14. #开始窗口的事件循环
15. win. mainloop()

保存 demo1.pyw 文件后，直接双击运行该文件，结果如图 1 所示：

  
图1：程序运行结果

#### 示例代码分析如下：

* 第 6 行：创建一个 Frame 控件，以作为窗体。此窗体的外形突起，边框厚度为 2 像素。窗体的宽度是 400 像素，高度是 300 像素。
* 第 7 行：此窗体在窗口的顶端（side=TOP），当窗口改变大小时，窗体会占满整个窗口的剩余空间（fill=BOTH）。widget 与窗体边界之间的水平距离是5像素，垂直距离是 5 像素。
* 第 13～11 行：创建第一个按钮。位置在距离窗体左上角的（40, 40）坐标处，宽度是 80 像素，高度是 40 像素。
* 第 14～15 行：创建第二个按钮。位置在距离窗体左上角的（140, 80）坐标处，宽度是 80 像素，高度是 40 像素。

# 用tkinter.pack设计复杂界面布局

置顶 快乐清风客 2018-11-04 12:41:59 17161 收藏 98

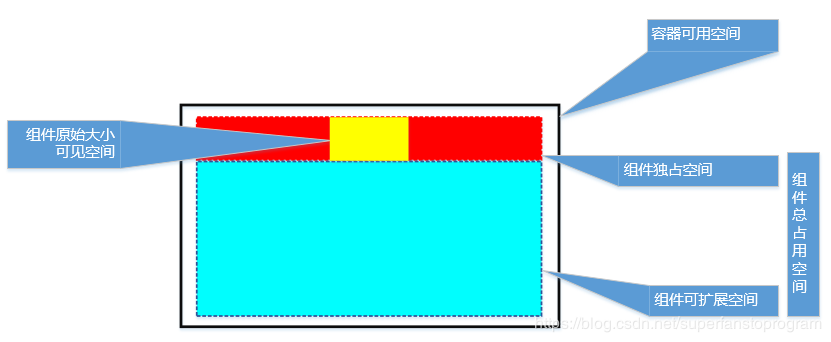
展开

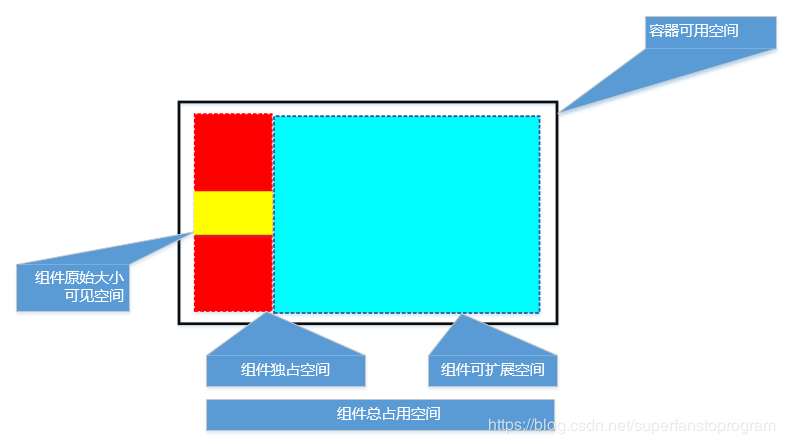
对程序员来说，界面设计的实现，要解决的一个主要问题就是如何根据界面设计的需求，将组件以规定的大小放置在规定的位置。tkinter是python语言中常用的界面实现模块，其中pack()方法，可创建浮动，自动伸缩扩展的软件界面，如以下界面。

如果你对以上用tkinter.pack实现的界面布局有兴趣，觉得有些实现效果可以参考借鉴，请继续阅读下面的内容。

## 一. pack的组件空间布局原理分析

用pack进行的布局界面组件，可用以下一些界面布局空间来进行分析

side = top 的布局空间

side = left 的布局空间

1. 容器可用空间

系统严格按代码的前后组织顺序依次排列组件，前辈组件还没有占据的空间，是当前组件的可用空间，pack布局是在当前组件的可用空间内进行的。如上图的黑边白底的区域。（为了说明，我留出padding空间，若没有padding，组件将占据整个目前可用空间）

2.组件原始大小可见空间

设置了width, height, 或由具体内容确定的组件原始大小，也就是可以看得见的组件宽高。如上图的黄色区域。

3.组件的独占空间

组件不单只占据它原始可见的位置，而是会同时占据可用空间的整行空间或整列空间。如上图红色区域。

side = top / bottom 时，组件占据整行，后辈组件只能依次排在它的下面 / 上面；组件为上下排列方式

side = left / right 时， 组件占据整列，后辈组件只能依次排在它的右边 / 左边；组件为左右排列方式

pack() , 我分析下来，其实就是 pack(side=top)

组件独占空间，没有进行填充(fill)时，不可见，但我们必须了解到它的存在，因为它直接影响组件本身及后辈组件的布局。

组件独占空间，是固有的，其它后辈组件绝对不可能进入该空间，后辈组件最多只能紧靠它。

4. 组件可扩展空间

组件独占空间的另一个方向，是组件的可扩展空间方向。如上图的浅蓝色区域。

side = top / bottom 时，组件独占整行，并可向整列方向进行扩展。

side = left / right 时，组件独占整列，并可向整行方向进行扩展。

组件可扩展空间，没有进行填充(fill)时，也是不可见的，但它同样会直接影响自身和后辈组件的布局，必须掌握它的空间存在。

若设置了 expand = yes, 则启用可扩展空间；若 expand = no, 则不启用可扩展空间。

组件可扩展空间，是动态的，是可以被后辈组件的独占空间占据压缩的，后辈组件可以从前辈可扩展空间的边缘开始入侵，直至压缩到前辈组件自身的独占空间为止。

5. 组件占用总空间

组件独占空间 + 组件可扩展空间，就是当前组件的总占用空间，若没有启用组件扩展空间（expand = no），则只包含组件独占空间。

接下来的 锚定位(anchor)和填充(fill)，都是基于组件的占用总空间来进行处理的。

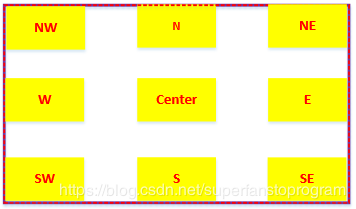
6. 空间定位和填充

锚定位（anchor ），加 padx, pady , 可将组件安排在指定位置

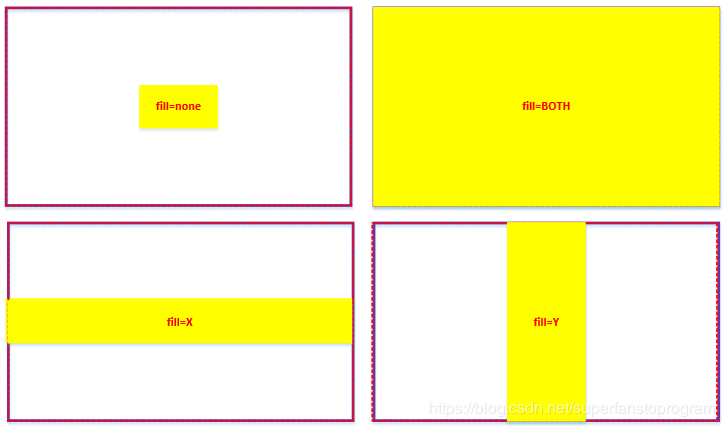
锚定位有9个方向，缺省为 center, 居中

若 side = top / bottom ,  expand=no, 则合并到一行，即NW, W, SW的效果一样，其他同理。

若 side = left / right,  expand = no, 则合并到一列，即NW, N, NE的效果一样，其他同理



填充（fill）组件，可以将组件的可见，可操作范围充满指定区域

填充的方式有4种，缺省的方式是（none）,不填充。

## 二. 单组件布局的分析步骤

以上描述的pack布局原理，需要仔细理解和掌握，这是pack布局的基础，当我们理解这些原理后，就可以用来控制单个组件的位置了。

基本分析步骤为：

1. 明确当前容器的可用空间范围

2.分析是用上下排列方式（side=top / bottom）,还是左右排列方式（side=left/right）

3.分析是否需要启用扩展空间，若需要则设置 expand = yes, 否则保持缺省值 expand = no

4.利用锚定位（anchor），确定组件的具体位置

5.利用填充（fill）, 对组件的可见区域进行填充

举例分析：



设置代码

布局效果

说明

.pack(side=‘top‘)

独占整行，缺省居中

.pack(side='top', anchor='sw')

没有启用可扩展空间，只独占行，锚定位 nw, w, sw 都一样，定位在西边

.pack(side='top', expand='yes', anchor='sw')

启用可扩展空间，除了独占的行空间，还扩展到了容器可用空间的所有列空间，锚定位'sw' ，定位到了容器的西南边

.pack(side='top', expand='yes')

启动扩展空间，没有设置anchor，则缺省居中（整个组件占用空间的中间，不同于第1个例子）

.pack(side='top', expand='yes', fill='y')

启用扩展空间，定位完后，进行“y”轴方向的填充，填满整个占用空间的列

.pack(side='top', fill='y')

没有启用扩展空间，当选择进行“y”方向填充时，也只是在独占的行空间内进行“y”方向填充

## 三.  多组件布局的分析

我们现在基本掌握了单个组件的布局定位分析方法，而实际的软件界面，一定是多组件的综合布局，涉及多组件之间的相互关系和影响。以上所有讲到的单组件布局分析，都有一个前提，就是在已知容器可用空间的前提下， 进行当前组件的布局定位。

整理分析多组件布局的要点：

1. 所有组件按代码的前后组织次序，依次进行布局定位

2. 设置side = top / bottom 的组件，为上下排列的组件，独占当时可用空间的整行

3. 设置side = left / right 的组件，为左右排列的组件，独占当时可用空间的整列

4. 后辈组件不可进入任何前辈组件的独占空间，不论 side的设置如何，是否相同，一律都是前辈

5. 组件可扩展空间，是动态的，是可以被后辈组件的独占空间占据压缩的，后辈组件可以从前辈可扩展空间的边缘开始入侵，直至压缩到前辈组件自身的独占空间为止

6.后辈组件的扩展空间，不能侵占前辈组件的扩展空间；既若扩展空间重叠，前辈享用扩展空间

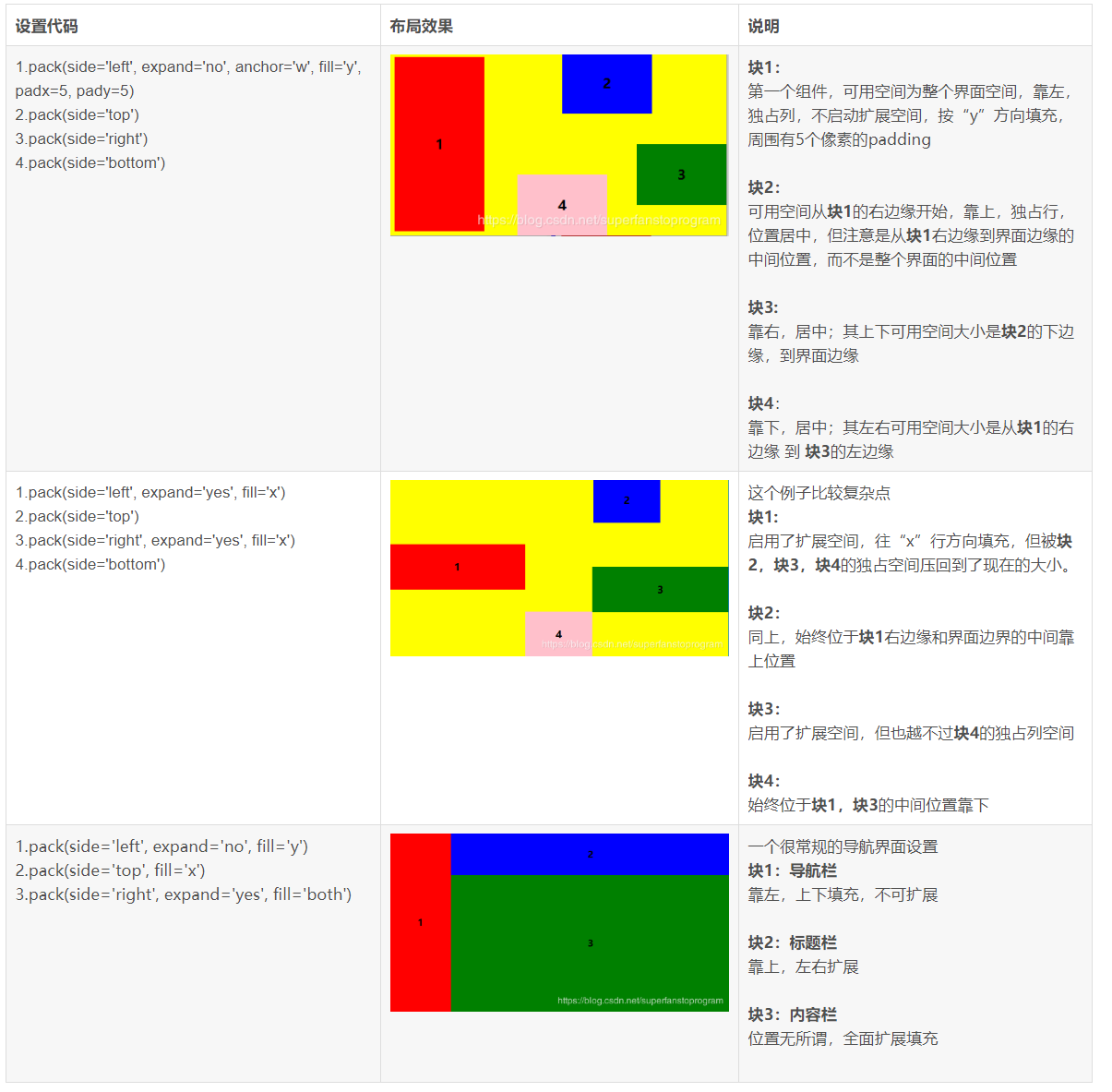
7. 若整个界面空间缩小，出现后辈组件独占空间跟前排组件独占空间重叠时，后辈组件将被压缩，直至消失。

8. 若界面空间继续缩小，小到比组件独占空间小时，组件将被压缩，直至消失。

9. 空间优先级别为：界面空间 > 前辈组件独占空间 > 后辈组件独占空间 > 前辈组件可扩展空间 > 后辈组件可扩展空间

10.可利用frame的多层嵌套，进行灵活布局

## 举例分析：



设置代码

布局效果

说明

1.pack(side='left', expand='no', anchor='w', fill='y', padx=5, pady=5)

2.pack(side='top')

3.pack(side='right')

4.pack(side='bottom')

块1：

第一个组件，可用空间为整个界面空间，靠左，独占列，不启动扩展空间，按“y”方向填充，周围有5个像素的padding

块2：

可用空间从块1的右边缘开始，靠上，独占行，位置居中，但注意是从块1右边缘到界面边缘的中间位置，而不是整个界面的中间位置

块3:

靠右，居中；其上下可用空间大小是块2的下边缘，到界面边缘

块4：

靠下，居中；其左右可用空间大小是从块1的右边缘 到 块3的左边缘

1.pack(side='left', expand='yes', fill='x')

2.pack(side='top')

3.pack(side='right', expand='yes', fill='x')

4.pack(side='bottom')

这个例子比较复杂点

块1:

启用了扩展空间，往“x”行方向填充，但被块2，块3，块4的独占空间压回到了现在的大小。

块2：

同上，始终位于块1右边缘和界面边界的中间靠上位置

块3：

启用了扩展空间，但也越不过块4的独占列空间

块4：

始终位于块1，块3的中间位置靠下

1.pack(side='left', expand='no', fill='y')

2.pack(side='top', fill='x')

3.pack(side='right', expand='yes', fill='both')

一个很常规的导航界面设置

块1：导航栏

靠左，上下填充，不可扩展

块2：标题栏

靠上，左右扩展

块3：内容栏

位置无所谓，全面扩展填充

四. 代码演示

我写了一个布局演示程序，包含文章开头展示的界面，可以对照界面实际效果和具体代码，来了解实现的细节。另外提供了单组件和多组件的演示工具，可以调整设置，观察效果，来理解和掌握pack布局的基本方法。演示程序中还包括一些其它窗体布局和实现的方法，希望对大家有帮助。

源代码位置：

https://github.com/EdwinZhang1970/Python/tree/master/tkinter-pack%20Demo



————————————————

版权声明：本文为CSDN博主「快乐清风客」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/superfanstoprogram/java/article/details/83713196>

| **Type** | **Name** | **Latest commit message** | **Commit time** |
| --- | --- | --- | --- |
|  | [**..**](https://github.com/EdwinZhang1970/Python) |  |  |
|  | [.idea](https://github.com/EdwinZhang1970/Python/tree/master/tkinter-pack%20Demo/.idea) | [Add Boxman game](https://github.com/EdwinZhang1970/Python/commit/b872abf8ea36a67f8c6aa413dfddfed466d7f16c) | 8 months ago |
|  | [\_\_pycache\_\_](https://github.com/EdwinZhang1970/Python/tree/master/tkinter-pack%20Demo/__pycache__) | [add tkinter pack demo source code](https://github.com/EdwinZhang1970/Python/commit/0caebb986f4e227ddac79e53a347b3334f9e51de) | 2 years ago |
|  | [images](https://github.com/EdwinZhang1970/Python/tree/master/tkinter-pack%20Demo/images) | [add tkinter pack demo source code](https://github.com/EdwinZhang1970/Python/commit/0caebb986f4e227ddac79e53a347b3334f9e51de) | 2 years ago |
|  | [app.bat](https://github.com/EdwinZhang1970/Python/blob/master/tkinter-pack%20Demo/app.bat) | [add tkinter pack demo source code](https://github.com/EdwinZhang1970/Python/commit/0caebb986f4e227ddac79e53a347b3334f9e51de) | 2 years ago |
|  | [main.py](https://github.com/EdwinZhang1970/Python/blob/master/tkinter-pack%20Demo/main.py) | [add tkinter pack demo source code](https://github.com/EdwinZhang1970/Python/commit/0caebb986f4e227ddac79e53a347b3334f9e51de) | 2 years ago |
|  | [tkutils.py](https://github.com/EdwinZhang1970/Python/blob/master/tkinter-pack%20Demo/tkutils.py) | [add tkinter pack demo source code](https://github.com/EdwinZhang1970/Python/commit/0caebb986f4e227ddac79e53a347b3334f9e51de) | 2 years ago |
|  | [win\_ai.py](https://github.com/EdwinZhang1970/Python/blob/master/tkinter-pack%20Demo/win_ai.py) | [add tkinter pack demo source code](https://github.com/EdwinZhang1970/Python/commit/0caebb986f4e227ddac79e53a347b3334f9e51de) | 2 years ago |
|  | [win\_multi.py](https://github.com/EdwinZhang1970/Python/blob/master/tkinter-pack%20Demo/win_multi.py) | [add tkinter pack demo source code](https://github.com/EdwinZhang1970/Python/commit/0caebb986f4e227ddac79e53a347b3334f9e51de) | 2 years ago |
|  | [win\_multi2.py](https://github.com/EdwinZhang1970/Python/blob/master/tkinter-pack%20Demo/win_multi2.py) | [add tkinter pack demo source code](https://github.com/EdwinZhang1970/Python/commit/0caebb986f4e227ddac79e53a347b3334f9e51de) | 2 years ago |
|  | [win\_single.py](https://github.com/EdwinZhang1970/Python/blob/master/tkinter-pack%20Demo/win_single.py) | [add tkinter pack demo source code](https://github.com/EdwinZhang1970/Python/commit/0caebb986f4e227ddac79e53a347b3334f9e51de) | 2 years ago |

## [main.py](https://github.com/EdwinZhang1970/Python/blob/master/tkinter-pack%20Demo/main.py)

|  |
| --- |
| # coding:utf-8 |
|  |  |
|  | import tkinter as tk |
|  | import tkinter.font as tkFont |
|  | from PIL import Image, ImageTk |
|  | import tkutils as tku |
|  | import win\_ai |
|  | import win\_single |
|  | import win\_multi |
|  |  |
|  |  |
|  | class App: |
|  | def \_\_init\_\_(self): |
|  | self.root = tk.Tk() |
|  | self.root.geometry("%dx%d" % (700, 400)) # 窗体尺寸 |
|  | tku.center\_window(self.root) # 将窗体移动到屏幕中央 |
|  | self.root.iconbitmap("images\\Money.ico") # 窗体图标 |
|  | self.root.title("Python pack 布局演示") |
|  | self.root.resizable(False, False) # 设置窗体不可改变大小 |
|  | self.no\_title = True |
|  | self.show\_title() |
|  | self.body() |
|  |  |
|  | def body(self): |
|  | # --------------------------------------------------------------------- |
|  | # 背景图片 |
|  | # --------------------------------------------------------------------- |
|  | self.img = ImageTk.PhotoImage(file="images\\bg1.png") |
|  | canvas = tk.Canvas(self.root, width=720, height=420) |
|  | canvas.create\_image(300, 200, image=self.img) |
|  | canvas.pack(expand=tk.YES, fill=tk.BOTH) |
|  |  |
|  | # --------------------------------------------------------------------- |
|  | # 标题栏 |
|  | # --------------------------------------------------------------------- |
|  | f1 = tk.Frame(canvas) |
|  |  |
|  | im1 = tku.image\_label(f1, "images\\python.png", 86, 86, False) |
|  | im1.configure(bg="Teal") |
|  | im1.bind('<Button-1>', self.show\_title) |
|  | im1.pack(side=tk.LEFT, anchor=tk.NW, fill=tk.Y) |
|  |  |
|  | ft1 = tkFont.Font(family="微软雅黑", size=24, weight=tkFont.BOLD) |
|  | tk.Label(f1, text="Pack 布局演示", height=2, fg="white", font=ft1, bg="Teal")\ |
|  | .pack(side=tk.LEFT, expand=tk.YES, fill=tk.X) |
|  |  |
|  | im2 = tku.image\_label(f1, "images\\close.png", 86, 86, False) |
|  | im2.configure(bg="Teal") |
|  | im2.bind('<Button-1>', self.close) |
|  | im2.pack(side=tk.RIGHT, anchor=tk.NW, fill=tk.Y) |
|  |  |
|  | f1.pack(fill=tk.X) |
|  |  |
|  | # --------------------------------------------------------------------- |
|  | # 功能按钮组 |
|  | # --------------------------------------------------------------------- |
|  | ft2 = tkFont.Font(family="微软雅黑", size=14, weight=tkFont.BOLD) |
|  | tk.Button(canvas, text="单组件演示", bg="cadetblue", command=self.show\_single, font=ft2, height=2, fg="white", width=15)\ |
|  | .pack(side=tk.LEFT, expand=tk.YES, anchor=tk.CENTER, padx=5) |
|  | tk.Button(canvas, text="多组件演示", bg="cadetblue", command=self.show\_multi, font=ft2, height=2, fg="white", width=15)\ |
|  | .pack(side=tk.LEFT, expand=tk.YES, anchor=tk.CENTER, padx=5) |
|  | tk.Button(canvas, text="应用平台界面", bg="cadetblue", command=self.show\_ai, font=ft2, height=2, fg="white", width=15)\ |
|  | .pack(side=tk.RIGHT, expand=tk.YES, anchor=tk.CENTER, padx=5) |
|  |  |
|  | def show\_title(self, \*args): |
|  | self.root.overrideredirect(self.no\_title) |
|  | self.no\_title = not self.no\_title |
|  |  |
|  | def show\_single(self): |
|  | win\_single.Window(self.root) |
|  |  |
|  | def show\_multi(self): |
|  | win\_multi.Window(self.root) |
|  |  |
|  | def show\_ai(self): |
|  | win\_ai.Window(self.root) |
|  |  |
|  | def close(self, \*arg): |
|  | if tku.show\_confirm("确认退出吗 ?"): |
|  | self.root.destroy() |
|  |  |
|  |  |
|  | if \_\_name\_\_ == "\_\_main\_\_": |
|  | app = App() |
|  | app.root.mainloop() |

## [tkutils.py](https://github.com/EdwinZhang1970/Python/blob/master/tkinter-pack%20Demo/tkutils.py)

|  |
| --- |
| # coding: utf-8 |
|  |  |
|  | import tkinter as tk |
|  | import tkinter.font as tkFont |
|  | from PIL import Image, ImageTk |
|  | from tkinter import messagebox |
|  |  |
|  |  |
|  | def show\_info(message=""): |
|  | messagebox.showinfo("提示框", message) |
|  |  |
|  |  |
|  | def show\_confirm(message=""): |
|  | """ |
|  | True : yes |
|  | False : no |
|  | """ |
|  | return messagebox.askyesno("确认框", message) |
|  |  |
|  |  |
|  | def center\_window(win, width=None, height=None): |
|  | """ 将窗口屏幕居中 """ |
|  | screenwidth = win.winfo\_screenwidth() |
|  | screenheight = win.winfo\_screenheight() |
|  | if width is None: |
|  | width, height = get\_window\_size(win)[:2] |
|  | size = '%dx%d+%d+%d' % (width, height, (screenwidth - width)/2, (screenheight - height)/3) |
|  | win.geometry(size) |
|  |  |
|  |  |
|  | def get\_window\_size(win, update=True): |
|  | """ 获得窗体的尺寸 """ |
|  | if update: |
|  | win.update() |
|  | return win.winfo\_width(), win.winfo\_height(), win.winfo\_x(), win.winfo\_y() |
|  |  |
|  |  |
|  | def tkimg\_resized(img, w\_box, h\_box, keep\_ratio=True): |
|  | """对图片进行按比例缩放处理""" |
|  | w, h = img.size |
|  |  |
|  | if keep\_ratio: |
|  | if w > h: |
|  | width = w\_box |
|  | height = int(h\_box \* (1.0 \* h / w)) |
|  |  |
|  | if h >= w: |
|  | height = h\_box |
|  | width = int(w\_box \* (1.0 \* w / h)) |
|  | else: |
|  | width = w\_box |
|  | height = h\_box |
|  |  |
|  | img1 = img.resize((width, height), Image.ANTIALIAS) |
|  | tkimg = ImageTk.PhotoImage(img1) |
|  | return tkimg |
|  |  |
|  |  |
|  | def image\_label(frame, img, width, height, keep\_ratio=True): |
|  | """输入图片信息，及尺寸，返回界面组件""" |
|  | if isinstance(img, str): |
|  | \_img = Image.open(img) |
|  | else: |
|  | \_img = img |
|  | lbl\_image = tk.Label(frame, width=width, height=height) |
|  |  |
|  | tk\_img = tkimg\_resized(\_img, width, height, keep\_ratio) |
|  | lbl\_image.image = tk\_img |
|  | lbl\_image.config(image=tk\_img) |
|  | return lbl\_image |
|  |  |
|  |  |
|  | def \_font(fname="微软雅黑", size=12, bold=tkFont.NORMAL): |
|  | """设置字体""" |
|  | ft = tkFont.Font(family=fname, size=size, weight=bold) |
|  | return ft |
|  |  |
|  |  |
|  | def \_ft(size=12, bold=False): |
|  | """极简字体设置函数""" |
|  | if bold: |
|  | return \_font(size=size, bold=tkFont.BOLD) |
|  | else: |
|  | return \_font(size=size, bold=tkFont.NORMAL) |
|  |  |
|  |  |
|  | def h\_seperator(parent, height=2): # height 单位为像素值 |
|  | """水平分割线, 水平填充 """ |
|  | tk.Frame(parent, height=height, bg="whitesmoke").pack(fill=tk.X) |
|  |  |
|  |  |
|  | def v\_seperator(parent, width, bg="whitesmoke"): # width 单位为像素值 |
|  | """垂直分割线 , fill=tk.Y, 但如何定位不确定，直接返回对象，由容器决定 """ |
|  | frame = tk.Frame(parent, width=width, bg=bg) |
|  | return frame |

## [win\_ai.py](https://github.com/EdwinZhang1970/Python/blob/master/tkinter-pack%20Demo/win_ai.py)

|  |
| --- |
| # coding:utf-8 |
|  |  |
|  | # \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | # 模块说明：利用 pack 方法，模拟构建人工智能的定制平台界面，只用于界面制作的学习参考 |
|  | # 界面代码进行了分层编写，在容器层，设置子组件的pack()， |
|  | # 在子组件层， 创建自身框架，布局自己的子组件， 并返回自身，供容器布局 |
|  | # 界面的结构层次比较清晰，修改调整会方便些 |
|  | # 开发人员: Edwin.Zhang |
|  | # 开发时间: 2018-09-28 |
|  | # \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  | import tkinter as tk |
|  | from tkinter import ttk |
|  | import tkinter.font as tkFont |
|  | from PIL import Image, ImageTk |
|  |  |
|  |  |
|  | def center\_window(win, width=None, height=None): |
|  | """ 将窗口屏幕居中 """ |
|  | screenwidth = win.winfo\_screenwidth() |
|  | screenheight = win.winfo\_screenheight() |
|  | if width is None: |
|  | width, height = get\_window\_size(win)[:2] |
|  | size = '%dx%d+%d+%d' % (width, height, (screenwidth - width)/2, (screenheight - height)/3) |
|  | win.geometry(size) |
|  |  |
|  |  |
|  | def get\_window\_size(win, update=True): |
|  | """ 获得窗体的尺寸 """ |
|  | if update: |
|  | win.update() |
|  | return win.winfo\_width(), win.winfo\_height(), win.winfo\_x(), win.winfo\_y() |
|  |  |
|  |  |
|  | def tkimg\_resized(img, w\_box, h\_box, keep\_ratio=True): |
|  | """对图片进行按比例缩放处理""" |
|  | w, h = img.size |
|  |  |
|  | if keep\_ratio: |
|  | if w > h: |
|  | width = w\_box |
|  | height = int(h\_box \* (1.0 \* h / w)) |
|  |  |
|  | if h >= w: |
|  | height = h\_box |
|  | width = int(w\_box \* (1.0 \* w / h)) |
|  | else: |
|  | width = w\_box |
|  | height = h\_box |
|  |  |
|  | img1 = img.resize((width, height), Image.ANTIALIAS) |
|  | tkimg = ImageTk.PhotoImage(img1) |
|  | return tkimg |
|  |  |
|  |  |
|  | def image\_label(frame, img, width, height, keep\_ratio=True): |
|  | """输入图片信息，及尺寸，返回界面组件""" |
|  | if isinstance(img, str): |
|  | \_img = Image.open(img) |
|  | else: |
|  | \_img = img |
|  | lbl\_image = tk.Label(frame, width=width, height=height) |
|  |  |
|  | tk\_img = tkimg\_resized(\_img, width, height, keep\_ratio) |
|  | lbl\_image.image = tk\_img |
|  | lbl\_image.config(image=tk\_img) |
|  | return lbl\_image |
|  |  |
|  |  |
|  | def \_font(fname="微软雅黑", size=12, bold=tkFont.NORMAL): |
|  | """设置字体""" |
|  | ft = tkFont.Font(family=fname, size=size, weight=bold) |
|  | return ft |
|  |  |
|  |  |
|  | def \_ft(size=12, bold=False): |
|  | """极简字体设置函数""" |
|  | if bold: |
|  | return \_font(size=size, bold=tkFont.BOLD) |
|  | else: |
|  | return \_font(size=size, bold=tkFont.NORMAL) |
|  |  |
|  |  |
|  | def h\_seperator(parent, height=2): # height 单位为像素值 |
|  | """水平分割线, 水平填充 """ |
|  | tk.Frame(parent, height=height, bg="whitesmoke").pack(fill=tk.X) |
|  |  |
|  |  |
|  | def v\_seperator(parent, width, bg="whitesmoke"): # width 单位为像素值 |
|  | """垂直分割线 , fill=tk.Y, 但如何定位不确定，直接返回对象，由容器决定 """ |
|  | frame = tk.Frame(parent, width=width, bg=bg) |
|  | return frame |
|  |  |
|  |  |
|  | class Window: |
|  | def \_\_init\_\_(self, parent): |
|  | self.root = tk.Toplevel() |
|  | self.parent = parent |
|  | self.root.geometry("%dx%d" % (1200, 800)) # 窗体尺寸 |
|  | center\_window(self.root) # 将窗体移动到屏幕中央 |
|  | self.root.title("人工智能") # 窗体标题 |
|  | self.root.iconbitmap("images\\Money.ico") # 窗体图标 |
|  | self.root.grab\_set() |
|  | self.body() # 绘制窗体组件 |
|  |  |
|  | # 绘制窗体组件 |
|  | def body(self): |
|  | self.title(self.root).pack(fill=tk.X) |
|  |  |
|  | self.main(self.root).pack(expand=tk.YES, fill=tk.BOTH) |
|  |  |
|  | self.bottom(self.root).pack(fill=tk.X) |
|  |  |
|  | def title(self, parent): |
|  | """ 标题栏 """ |
|  |  |
|  | def label(frame, text, size, bold=False): |
|  | return tk.Label(frame, text=text, bg="black", fg="white", height=2, font=\_ft(size, bold)) |
|  |  |
|  | frame = tk.Frame(parent, bg="black") |
|  |  |
|  | label(frame, "人工智能应用平台", 16, True).pack(side=tk.LEFT, padx=10) |
|  | label(frame, "图像模型定制", 12).pack(side=tk.LEFT, padx=100) |
|  | label(frame, "声音模型定制", 12).pack(side=tk.LEFT, padx=0) |
|  | label(frame, "定制模型", 12).pack(side=tk.LEFT, padx=100) |
|  | label(frame, "", 12).pack(side=tk.RIGHT, padx=20) |
|  | label(frame, "登录用户", 12).pack(side=tk.RIGHT, padx=20) |
|  | image\_label(frame, "images\\user.png", 40, 40, False).pack(side=tk.RIGHT) |
|  |  |
|  | return frame |
|  |  |
|  | def bottom(self, parent): |
|  | """ 窗体最下面留空白 """ |
|  |  |
|  | frame = tk.Frame(parent, height=10, bg="whitesmoke") |
|  | frame.propagate(True) |
|  | return frame |
|  |  |
|  | def main(self, parent): |
|  | """ 窗体主体 """ |
|  |  |
|  | frame = tk.Frame(parent, bg="whitesmoke") |
|  |  |
|  | self.main\_top(frame).pack(fill=tk.X, padx=30, pady=15) |
|  | self.main\_left(frame).pack(side=tk.LEFT, fill=tk.Y, padx=30) |
|  | v\_seperator(frame, 30).pack(side=tk.RIGHT, fill=tk.Y) |
|  | self.main\_right(frame).pack(side=tk.RIGHT, expand=tk.YES, fill=tk.BOTH) |
|  |  |
|  | return frame |
|  |  |
|  | def main\_top(self, parent): |
|  | def label(frame, text, size=12): |
|  | return tk.Label(frame, bg="white", fg="gray", text=text, font=\_ft(size)) |
|  |  |
|  | frame = tk.Frame(parent, bg="white", height=150) |
|  |  |
|  | image\_label(frame, "images\\img\_title.png", width=240, height=120, keep\_ratio=False) \ |
|  | .pack(side=tk.LEFT, padx=10, pady=10) |
|  |  |
|  | self.main\_top\_middle(frame).pack(side=tk.LEFT) |
|  |  |
|  | label(frame, "收起^").pack(side=tk.RIGHT, padx=10) |
|  |  |
|  | frame.propagate(False) |
|  | return frame |
|  |  |
|  | def main\_top\_middle(self, parent): |
|  | str1 = "定制图像分类模型，可以识别一张图整体是什么物体/状态/场景。" |
|  | str2 = "在各分类图片之间差异明显的情况下，训练数据每类仅需20-100张，最快10分钟可训练完毕" |
|  |  |
|  | def label(frame, text): |
|  | return tk.Label(frame, bg="white", fg="gray", text=text, font=\_ft(12)) |
|  |  |
|  | frame = tk.Frame(parent, bg="white") |
|  |  |
|  | self.main\_top\_middle\_top(frame).pack(anchor=tk.NW) |
|  |  |
|  | label(frame, str1).pack(anchor=tk.W, padx=10, pady=2) |
|  | label(frame, str2).pack(anchor=tk.W, padx=10) |
|  |  |
|  | return frame |
|  |  |
|  | def main\_top\_middle\_top(self, parent): |
|  | def label(frame, text, size=12, bold=True, fg="blue"): |
|  | return tk.Label(frame, text=text, bg="white", fg=fg, font=\_ft(size, bold)) |
|  |  |
|  | frame = tk.Frame(parent, bg="white") |
|  |  |
|  | label(frame, "图像分类模型", 20, True, "black").pack(side=tk.LEFT, padx=10) |
|  | label(frame, "操作文档").pack(side=tk.LEFT, padx=10) |
|  | label(frame, "教学视频").pack(side=tk.LEFT, padx=10) |
|  | label(frame, "常见问题").pack(side=tk.LEFT, padx=10) |
|  |  |
|  | return frame |
|  |  |
|  | def main\_left(self, parent): |
|  | def label(frame, text, size=10, bold=False, bg="white"): |
|  | return tk.Label(frame, text=text, bg=bg, font=\_ft(size, bold)) |
|  |  |
|  | frame = tk.Frame(parent, width=180, bg="white") |
|  |  |
|  | label(frame, "模型中心", 12, True).pack(anchor=tk.W, padx=20, pady=10) |
|  | label(frame, "我的模型").pack(anchor=tk.W, padx=40, pady=5) |
|  |  |
|  | f1 = tk.Frame(frame, bg="whitesmoke") |
|  | v\_seperator(f1, width=5, bg="blue").pack(side=tk.LEFT, fill=tk.Y) |
|  | label(f1, "创建模型", bg="whitesmoke").pack(side=tk.LEFT, anchor=tk.W, padx=35, pady=5) |
|  | f1.pack(fill=tk.X) |
|  |  |
|  | label(frame, "训练模型").pack(anchor=tk.W, padx=40, pady=5) |
|  | label(frame, "校验模型").pack(anchor=tk.W, padx=40, pady=5) |
|  | label(frame, "发布模型").pack(anchor=tk.W, padx=40, pady=5) |
|  |  |
|  | h\_seperator(frame, 10) |
|  |  |
|  | label(frame, "数据中心", 12, True).pack(anchor=tk.W, padx=20, pady=10) |
|  | label(frame, "数据集管理").pack(anchor=tk.W, padx=40, pady=5) |
|  | label(frame, "创建数据集").pack(anchor=tk.W, padx=40, pady=5) |
|  |  |
|  | frame.propagate(False) |
|  | return frame |
|  |  |
|  | def main\_right(self, parent): |
|  | def label(frame, text, size=10, bold=False, fg="black"): |
|  | return tk.Label(frame, text=text, bg="white", fg=fg, font=\_ft(size, bold)) |
|  |  |
|  | def space(n): |
|  | s = " " |
|  | r = "" |
|  | for i in range(n): |
|  | r += s |
|  | return r |
|  |  |
|  | frame = tk.Frame(parent, width=200, bg="white") |
|  |  |
|  | label(frame, "创建模型", 12, True).pack(anchor=tk.W, padx=20, pady=5) |
|  |  |
|  | h\_seperator(frame) |
|  |  |
|  | f1 = tk.Frame(frame, bg="white") |
|  | label(f1, space(8) + "模型类别:").pack(side=tk.LEFT, pady=5) |
|  | label(f1, "图像分类").pack(side=tk.LEFT, padx=20) |
|  | f1.pack(fill=tk.X) |
|  |  |
|  | f2 = tk.Frame(frame, bg="white") |
|  | label(f2, space(5) + "\*", fg="red").pack(side=tk.LEFT, pady=5) |
|  | label(f2, "模型名称:").pack(side=tk.LEFT) |
|  | tk.Entry(f2, bg="white", font=\_ft(10), width=25).pack(side=tk.LEFT, padx=20) |
|  | f2.pack(fill=tk.X) |
|  |  |
|  | f3 = tk.Frame(frame, bg="white") |
|  | label(f3, space(5) + "\*", fg="red").pack(side=tk.LEFT, pady=5) |
|  | label(f3, "联系方式:").pack(side=tk.LEFT) |
|  | tk.Entry(f3, bg="white", font=\_ft(10), width=25).pack(side=tk.LEFT, padx=20) |
|  | f3.pack(fill=tk.X) |
|  |  |
|  | f4 = tk.Frame(frame, bg="white") |
|  | label(f4, space(5) + "\*", fg="red").pack(side=tk.LEFT, anchor=tk.N, pady=5) |
|  | label(f4, "功能描述:").pack(side=tk.LEFT, anchor=tk.N, pady=5) |
|  | tk.Text(f4, bg="white", font=\_ft(10), height=10, width=40).pack(side=tk.LEFT, padx=20, pady=5) |
|  | f4.pack(fill=tk.X) |
|  |  |
|  | ttk.Button(frame, text="下一步", width=12).pack(anchor=tk.W, padx=112, pady=5) |
|  |  |
|  | return frame |

## [win\_multi.py](https://github.com/EdwinZhang1970/Python/blob/master/tkinter-pack%20Demo/win_multi.py)

|  |
| --- |
| # coding:utf-8 |
|  |  |
|  | import tkinter as tk |
|  | import tkinter.font as tkFont |
|  | import tkutils as tku |
|  | import win\_multi2 |
|  |  |
|  | colors = ['red', 'yellow', 'blue', 'green', 'pink', 'slateblue', 'lawngreen', 'orange', 'gold', 'cyan', |
|  | 'cyan', 'brown', 'gray', 'royalblue', 'magenta', 'olive', 'black'] |
|  |  |
|  |  |
|  | class Window: |
|  | def \_\_init\_\_(self, parent): |
|  | self.root = tk.Toplevel() |
|  | self.parent = parent |
|  | self.root.geometry("%dx%d" % (1000, 700)) # 窗体尺寸 |
|  | tku.center\_window(self.root) # 将窗体移动到屏幕中央 |
|  | self.root.title("多组件的pack布局演示") # 窗体标题 |
|  | self.root.grab\_set() |
|  | self.root.resizable(False, False) |
|  | self.default\_code = "1.pack(side='left', expand='no', anchor='w', fill='y', padx=5, pady=5)\n" |
|  | self.default\_code += "2.pack(side='top')\n" |
|  | self.default\_code += "3.pack(side='right')\n" |
|  | self.default\_code += "4.pack(side='bottom')\n" |
|  | self.body() # 绘制窗体组件 |
|  | self.reset() |
|  |  |
|  | # 绘制窗体组件 |
|  | def body(self): |
|  | self.title(self.root).pack(fill=tk.X) |
|  | tk.Frame(self.root, height=2).pack(fill=tk.X) |
|  |  |
|  | self.parameters(self.root).pack(side=tk.LEFT, fill=tk.Y) |
|  |  |
|  | tk.Frame(self.root, width=5, bg="blue").pack(side=tk.LEFT, fill=tk.Y) |
|  | tk.Frame(self.root, height=5, bg="blue").pack(side=tk.TOP, fill=tk.X) |
|  | self.box = tk.Frame(self.root, height=5, bg="blue") |
|  | self.box.pack(side=tk.BOTTOM, fill=tk.X) |
|  |  |
|  | self.frame\_main = tk.Frame(self.root, bg="white", bd=2) |
|  | self.frame\_main.pack(side=tk.LEFT, expand=tk.YES, fill=tk.BOTH) |
|  |  |
|  | self.bottom = tk.Frame(self.root, width=5, bg="blue") |
|  | self.bottom.pack(side=tk.LEFT, fill=tk.Y) |
|  |  |
|  | def title(self, parent): |
|  | frame = tk.Frame(parent, height=50, bg="black") |
|  |  |
|  | ft0 = tkFont.Font(family="微软雅黑", size=14, weight=tkFont.BOLD) |
|  | tk.Label(frame, font=ft0, bg="black", fg="white", text="设置范例：")\ |
|  | .pack(side=tk.LEFT, padx=10) |
|  |  |
|  | ft1 = tkFont.Font(family="微软雅黑", size=12, weight=tkFont.BOLD) |
|  | tk.Label(frame, height=2, font=ft1, bg="black", fg="white", |
|  | text="n.pack(side='left', expand='no', anchor='w', fill='y', padx=5, pady=5) (全小写字母 + 单引号)")\ |
|  | .pack(side=tk.LEFT, padx=5) |
|  |  |
|  | frame.propagate = False |
|  | return frame |
|  |  |
|  | def parameters(self, parent): |
|  | frame = tk.Frame(parent, width=500) |
|  |  |
|  | ft1 = tkFont.Font(family="微软雅黑", size=12, weight=tkFont.BOLD) |
|  | ft2 = tkFont.Font(family="微软雅黑", size=10, weight=tkFont.BOLD) |
|  |  |
|  | f\_side = tk.Frame(frame) |
|  | tk.Button(f\_side, text="复位", width=10, height=1, bg="cadetblue", font=ft1, command=self.reset)\ |
|  | .pack(side=tk.LEFT, anchor=tk.W, padx=5, pady=5) |
|  |  |
|  | tk.Button(f\_side, text="刷新", width=10, height=1, bg="cadetblue", font=ft1, command=self.refresh)\ |
|  | .pack(side=tk.LEFT, anchor=tk.W, padx=10) |
|  |  |
|  | tk.Button(f\_side, text="弹窗显示", width=10, height=1, bg="cadetblue", font=ft1, command=self.popup)\ |
|  | .pack(side=tk.LEFT, anchor=tk.W, padx=10) |
|  |  |
|  | f\_side.pack(fill=tk.X, anchor=tk.W) |
|  |  |
|  | self.controls = tk.Text(frame, font=ft2) |
|  | self.controls.pack(expand='yes', fill='both', padx=5, pady='10') |
|  | self.controls.insert(tk.END, self.default\_code) |
|  |  |
|  | frame.propagate(False) |
|  | return frame |
|  |  |
|  | def reset(self): |
|  | self.controls.delete(0.0, tk.END) |
|  | self.controls.insert(tk.END, self.default\_code) |
|  | self.refresh() |
|  |  |
|  | def refresh(self): |
|  | self.frame\_main.forget() |
|  | del self.frame\_main |
|  |  |
|  | self.bottom.forget() |
|  | del self.bottom |
|  |  |
|  | self.frame\_main = tk.Frame(self.root, bg="white", bd=2) |
|  | self.frame\_main.pack(side=tk.LEFT, expand=tk.YES, fill=tk.BOTH) |
|  |  |
|  | self.bottom = tk.Frame(self.root, width=5, bg="blue") |
|  | self.bottom.pack(side=tk.LEFT, fill=tk.Y) |
|  |  |
|  | ft = tkFont.Font(family="微软雅黑", size=12, weight=tkFont.BOLD) |
|  | all\_code = self.controls.get(0.0, tk.END) |
|  | len\_color = len(colors) - 1 |
|  | all\_error = "" |
|  | for i, one in enumerate(all\_code.split("\n")): |
|  | if one is not None and one.strip() != "": |
|  | color = colors[i % len\_color] |
|  | desc = one.split(".")[0] |
|  | pack = one.split(".")[1:][0] |
|  | pack\_code = "tk.Label(self.frame\_main, font=ft, text='{}', width=10, height=3, bg='{}')".format(desc, color) |
|  | pack\_code += "." + pack |
|  | try: |
|  | exec(pack\_code) |
|  | except Exception as err: |
|  | print(pack\_code) |
|  | all\_error += one + "\n" |
|  | if all\_error != "": |
|  | message = "以下布局设置语法有错误，请检查:\n\n" |
|  | message += all\_error |
|  | tku.show\_info(message) |
|  |  |
|  | def popup(self): |
|  | all\_code = self.controls.get(0.0, tk.END) |
|  | win\_multi2.Window(self.root, all\_code) |

## [win\_multi2.py](https://github.com/EdwinZhang1970/Python/blob/master/tkinter-pack%20Demo/win_multi2.py)

|  |
| --- |
| # coding:utf-8 |
|  |  |
|  | import tkinter as tk |
|  | import tkinter.font as tkFont |
|  | import tkutils as tku |
|  |  |
|  | colors = ['red', 'yellow', 'blue', 'green', 'pink', 'slateblue', 'lawngreen', 'orange', 'gold', 'cyan', |
|  | 'cyan', 'brown', 'gray', 'royalblue', 'magenta', 'olive', 'black'] |
|  |  |
|  |  |
|  | class Window: |
|  | def \_\_init\_\_(self, parent, pack\_code): |
|  | self.root = tk.Toplevel() |
|  | self.parent = parent |
|  | self.root.geometry("%dx%d" % (600, 400)) |
|  | tku.center\_window(self.root) |
|  | self.root.title("多组件的pack布局, 改变窗体大小，观察组件的变化") |
|  | self.root.grab\_set() |
|  | self.root.resizable(True, True) |
|  | self.pack\_code = pack\_code |
|  | self.body() |
|  |  |
|  | # 绘制窗体组件 |
|  | def body(self): |
|  | all\_code = self.pack\_code |
|  | len\_color = len(colors) - 1 |
|  | all\_error = "" |
|  | ft = tkFont.Font(family="微软雅黑", size=12, weight=tkFont.BOLD) |
|  | for i, one in enumerate(all\_code.split("\n")): |
|  | if one is not None and one.strip() != "": |
|  | color = colors[i % len\_color] |
|  | desc = one.split(".")[0] |
|  | pack = one.split(".")[1:][0] |
|  | pack\_code = "tk.Label(self.root, font=ft, text='{}', width=10, height=3, bg='{}')".format(desc, color) |
|  | pack\_code += "." + pack |
|  | try: |
|  | exec(pack\_code) |
|  | except Exception as err: |
|  | all\_error += one + "\n" |
|  | if all\_error != "": |
|  | message = "以下布局设置语法有错误，请检查:\n\n" |
|  | message += all\_error |
|  | tku.show\_info(message) |

## [win\_single.py](https://github.com/EdwinZhang1970/Python/blob/master/tkinter-pack%20Demo/win_single.py)

|  |
| --- |
| # coding:utf-8 |
|  |  |
|  | import tkinter as tk |
|  | import tkinter.font as tkFont |
|  | import tkutils as tku |
|  |  |
|  |  |
|  | class Window: |
|  | def \_\_init\_\_(self, parent): |
|  | self.root = tk.Toplevel() |
|  | self.parent = parent |
|  | self.root.geometry("%dx%d" % (1000, 700)) # 窗体尺寸 |
|  | tku.center\_window(self.root) # 将窗体移动到屏幕中央 |
|  | self.root.title("单组件的pack布局演示") # 窗体标题 |
|  | self.root.grab\_set() |
|  | self.root.resizable(False, False) |
|  |  |
|  | self.v\_code\_pack = tk.StringVar() |
|  | self.v\_code\_pack.set(".pack()") |
|  |  |
|  | self.v\_side = tk.StringVar() |
|  | self.v\_side.set("") |
|  | self.side\_name = ("RIGHT", "LEFT", "TOP", "BOTTOM", "NONE") |
|  |  |
|  | self.v\_expand = tk.StringVar() |
|  | self.v\_expand.set("") |
|  | self.expand\_name = ("YES", "NO", "NONE") |
|  |  |
|  | self.v\_anchor = tk.StringVar() |
|  | self.v\_anchor.set("") |
|  | self.anchor\_name = ("W", "E", "N", "S", "CENTER", "NW", "NE", "SW", "SE", "NONE") |
|  |  |
|  | self.v\_fill = tk.StringVar() |
|  | self.v\_fill.set("") |
|  | self.fill\_name = ("X", "Y", "BOTH", "NONE") |
|  |  |
|  | self.v\_padx = tk.StringVar() |
|  | self.v\_padx.set("0") |
|  |  |
|  | self.v\_pady = tk.StringVar() |
|  | self.v\_pady.set("0") |
|  |  |
|  | self.body() # 绘制窗体组件 |
|  |  |
|  | # 绘制窗体组件 |
|  | def body(self): |
|  | self.title(self.root).pack(fill=tk.X) |
|  | tk.Frame(self.root, height=2).pack(fill=tk.X) |
|  |  |
|  | self.parameters(self.root).pack(side=tk.LEFT, fill=tk.Y) |
|  |  |
|  | tk.Frame(self.root, width=5, bg="blue").pack(side=tk.LEFT, fill=tk.Y) |
|  | tk.Frame(self.root, height=5, bg="blue").pack(side=tk.TOP, fill=tk.X) |
|  | tk.Frame(self.root, height=5, bg="blue").pack(side=tk.BOTTOM, fill=tk.X) |
|  | self.main(self.root).pack(side=tk.LEFT, expand=tk.YES, fill=tk.BOTH) |
|  | tk.Frame(self.root, width=5, bg="blue").pack(side=tk.LEFT, fill=tk.Y) |
|  |  |
|  | def title(self, parent): |
|  | frame = tk.Frame(parent, height=50, bg="black") |
|  |  |
|  | ft0 = tkFont.Font(family="微软雅黑", size=18, weight=tkFont.BOLD) |
|  | tk.Label(frame, font=ft0, bg="black", fg="white", text="当前设置：")\ |
|  | .pack(side=tk.LEFT, padx=20) |
|  |  |
|  | ft1 = tkFont.Font(family="微软雅黑", size=14, weight=tkFont.BOLD) |
|  | tk.Label(frame, height=2, font=ft1, bg="black", fg="white", textvariable=self.v\_code\_pack)\ |
|  | .pack(side=tk.LEFT, padx=5) |
|  |  |
|  | frame.propagate = False |
|  | return frame |
|  |  |
|  | def parameters(self, parent): |
|  | frame = tk.Frame(parent, width=400) |
|  |  |
|  | ft1 = tkFont.Font(family="微软雅黑", size=12, weight=tkFont.BOLD) |
|  |  |
|  | ft2 = tkFont.Font(family="微软雅黑", size=14, weight=tkFont.BOLD) |
|  |  |
|  | f\_side = tk.Frame(frame) |
|  | \_var\_side = tk.StringVar(value=self.side\_name) |
|  | self.p\_side = tk.Listbox(f\_side, listvariable=\_var\_side, selectmode='single', width=20, height=5, font=ft1) |
|  | self.p\_side.pack(side=tk.RIGHT, padx=10, pady=5) |
|  | self.p\_side.bind('<<ListboxSelect>>', self.get\_side) |
|  | tk.Label(f\_side, font=ft2, text="side:").pack(side=tk.RIGHT, anchor=tk.W, padx=10) |
|  | tk.Button(f\_side, text="复位", width=10, height=1, bg="cadetblue", font=ft2, command=self.reset)\ |
|  | .pack(side=tk.RIGHT, anchor=tk.W, padx=10) |
|  | f\_side.pack(fill=tk.X, anchor=tk.W) |
|  |  |
|  | f\_expand = tk.Frame(frame) |
|  | \_var\_expand = tk.StringVar(value=self.expand\_name) |
|  | self.p\_expand = tk.Listbox(f\_expand, listvariable=\_var\_expand, selectmode='single', width=20, height=3, font=ft1) |
|  | self.p\_expand.pack(side=tk.RIGHT, padx=10) |
|  | self.p\_expand.bind('<<ListboxSelect>>', self.get\_expand) |
|  | tk.Label(f\_expand, font=ft2, text="expand:").pack(side=tk.RIGHT, anchor=tk.W, padx=10) |
|  | f\_expand.pack(fill=tk.X, anchor=tk.W,) |
|  |  |
|  | f\_anchor = tk.Frame(frame) |
|  | \_var\_anchor = tk.StringVar(value=self.anchor\_name) |
|  | self.p\_anchor = tk.Listbox(f\_anchor, listvariable=\_var\_anchor, selectmode='single', width=20, height=10, font=ft1) |
|  | self.p\_anchor.pack(side=tk.RIGHT, padx=10) |
|  | self.p\_anchor.bind('<<ListboxSelect>>', self.get\_anchor) |
|  | tk.Label(f\_anchor, font=ft2, text="anchor:").pack(side=tk.RIGHT, anchor=tk.W, padx=10) |
|  | f\_anchor.pack(fill=tk.X, anchor=tk.W, pady=5) |
|  |  |
|  | f\_fill = tk.Frame(frame) |
|  | \_var\_fill = tk.StringVar(value=self.fill\_name) |
|  | self.p\_fill = tk.Listbox(f\_fill, listvariable=\_var\_fill, selectmode='single', width=20, height=4, font=ft1) |
|  | self.p\_fill.pack(side=tk.RIGHT, padx=10) |
|  | self.p\_fill.bind('<<ListboxSelect>>', self.get\_fill) |
|  | tk.Label(f\_fill, font=ft2, text="fill:").pack(side=tk.RIGHT, anchor=tk.W, padx=10) |
|  | f\_fill.pack(fill=tk.X, anchor=tk.W) |
|  |  |
|  | f\_padx = tk.Frame(frame) |
|  | self.p\_padx = tk.Spinbox(f\_padx, from\_=0, to=300, textvariable=self.v\_padx, width=19, font=ft1, command=self.layout) |
|  | self.p\_padx.pack(side=tk.RIGHT, padx=10) |
|  | tk.Label(f\_padx, font=ft2, text="padx:").pack(side=tk.RIGHT, anchor=tk.W, padx=10) |
|  | f\_padx.pack(fill=tk.X, anchor=tk.W, pady=5) |
|  |  |
|  | f\_pady = tk.Frame(frame) |
|  | self.p\_pady = tk.Spinbox(f\_pady, from\_=0, to=300, textvariable=self.v\_pady, width=19, font=ft1, command=self.layout) |
|  | self.p\_pady.pack(side=tk.RIGHT, padx=10) |
|  | tk.Label(f\_pady, font=ft2, text="pady:").pack(side=tk.RIGHT, anchor=tk.W, padx=10) |
|  | f\_pady.pack(fill=tk.X, anchor=tk.W, pady=5) |
|  |  |
|  | frame.propagate(False) |
|  | return frame |
|  |  |
|  | def main(self, parent): |
|  | self.frame\_main = tk.Frame(parent, bg="lightgreen", bd=2) |
|  | self.element = tk.Label(self.frame\_main, width=15, height=3, bg="red") |
|  | self.element.pack() |
|  | return self.frame\_main |
|  |  |
|  | def get\_side(self, \*arg): |
|  | idxs = self.p\_side.curselection() |
|  | if idxs is not None and len(idxs) == 1: |
|  | idx = int(idxs[0]) |
|  | self.v\_side.set(self.side\_name[idx]) |
|  | self.layout() |
|  |  |
|  | def get\_expand(self, \*arg): |
|  | idxs = self.p\_expand.curselection() |
|  | if idxs is not None and len(idxs) == 1: |
|  | idx = int(idxs[0]) |
|  | self.v\_expand.set(self.expand\_name[idx]) |
|  | self.layout() |
|  |  |
|  | def get\_anchor(self, \*arg): |
|  | idxs = self.p\_anchor.curselection() |
|  | if idxs is not None and len(idxs) == 1: |
|  | idx = int(idxs[0]) |
|  | self.v\_anchor.set(self.anchor\_name[idx]) |
|  | self.layout() |
|  |  |
|  | def get\_fill(self, \*arg): |
|  | idxs = self.p\_fill.curselection() |
|  | if idxs is not None and len(idxs) == 1: |
|  | idx = int(idxs[0]) |
|  | self.v\_fill.set(self.fill\_name[idx]) |
|  | self.layout() |
|  |  |
|  | def reset(self): |
|  | self.v\_side.set("") |
|  | self.v\_expand.set("") |
|  | self.v\_anchor.set("") |
|  | self.v\_fill.set("") |
|  | self.v\_padx.set("0") |
|  | self.v\_pady.set("0") |
|  | self.layout() |
|  |  |
|  | def layout(self): |
|  | self.element.forget() |
|  | del self.element |
|  |  |
|  | self.element = tk.Label(self.frame\_main, width=15, height=3, bg="red") |
|  | self.element.pack() |
|  |  |
|  | code = ".pack(" |
|  | is\_add = False |
|  | if self.v\_side.get() != "": |
|  | if self.v\_side.get() != "NONE": |
|  | code += "side=" + self.v\_side.get() |
|  | is\_add = True |
|  | self.element.pack(side=self.v\_side.get().lower()) |
|  |  |
|  | if self.v\_expand.get() != "": |
|  | if self.v\_expand.get() != "NONE": |
|  | if is\_add: |
|  | code += ", " |
|  | code += "expand=" + self.v\_expand.get() |
|  | is\_add = True |
|  | self.element.pack(expand=self.v\_expand.get().lower()) |
|  |  |
|  | if self.v\_anchor.get() != "": |
|  | if self.v\_anchor.get() != "NONE": |
|  | if is\_add: |
|  | code += ", " |
|  | code += "anchor=" + self.v\_anchor.get() |
|  | is\_add = True |
|  | self.element.pack(anchor=self.v\_anchor.get().lower()) |
|  |  |
|  | if self.v\_fill.get() != "": |
|  | if self.v\_fill.get() != "NONE": |
|  | if is\_add: |
|  | code += ", " |
|  | code += "fill=" + self.v\_fill.get() |
|  | is\_add = True |
|  | self.element.pack(fill=self.v\_fill.get().lower()) |
|  |  |
|  | x = int(self.v\_padx.get()) |
|  | if x > 0: |
|  | if is\_add: |
|  | code += ", " |
|  | code += "padx= " + str(x) |
|  | is\_add = True |
|  | self.element.pack(padx=x) |
|  |  |
|  | y = int(self.v\_pady.get()) |
|  | if y > 0: |
|  | if is\_add: |
|  | code += ", " |
|  | code += "pady= " + str(y) |
|  | is\_add = True |
|  | self.element.pack(pady=y) |
|  |  |
|  | code += ")" |
|  | self.v\_code\_pack.set(code) |