



欢迎来到洪卫的博客

。衣带渐宽终不悔，为伊消得人憔悴。众里寻他千百度，蓦然回首，那人却在灯火阑珊处。人生三从境界：昨夜西风凋碧树，独上高楼，望尽天涯路

博客园首页 我的首页 全部分类 知识框架 Linux相关 Python基础 Python进阶 项目实战 人工智能 阅读人生

学习资源 娱乐休闲

导航

管 理

<	2019年4月							>
日	一	二	三	四	五	六		
31	1	2	3	4	5	6		
7	8	9	10	11	12	13		
14	15	16	17	18	19	20		
21	22	23	24	25	26	27		
28	29	30	1	2	3	4		
5	6	7	8	9	10	11		

公告



我的博客目录结构
一个不像技术博客的博客



Visitors

	CN 29,668		GB 63
	TW 1,037		CA 56
	US 624		DE 46
	HK 365		KR 41
	JP 155		MY 32
	SG 107		RU 27
	AU 87		FR 23

Pageviews: 52,935



总访客

39538

音乐1 - 広橋真紀子

00:00 / 00:00

Python GUI之tkinter窗口视窗教程大集合（看这篇就够了）

一、前言

由于本篇文章较长，所以下面给出内容目录方便跳转阅读，当然也可以用博客页面最右侧的文章目录导航栏进行跳转查阅。

- 一、前言
- 二、Tkinter 是什么
- 三、Tkinter 控件详细介绍
 - 1. Tkinter 模块元素简要说明
 - 2. 常用窗口部件及简要说明：
- 四、动手实践学习
 - 1. 创建主窗口及Label部件（标签）创建使用
 - 2. Button窗口部件
 - 3. Entry窗口部件
 - 4. Text窗口部件
 - 5. Listbox窗口部件
 - 6. Radiobutton窗口部件
 - 7. Checkbutton窗口部件
 - 8. Scale窗口部件
 - 9. Canvas窗口部件
 - 10. Menu窗口部件
 - 11. Frame 窗口部件
 - 12. messageBox窗口部件
 - 13. 窗口部件三种放置方式pack/grid/place
 - 14. 综合练习，用户登录窗口例子
 - 15. 其他部件后续再补充...

二、Tkinter是什么

Tkinter 是使用 python 进行窗口视窗设计的模块。Tkinter模块("Tk 接口")是Python的标准Tk GUI工具包的接口。作为 python 特定的GUI界面，是一个图像的窗口，tkinter是python 自带的，可以编辑的GUI界面，我们可以用GUI 实现很多直观的功能，比如想开发一个计算器，如果只是一个程序输入，输出窗口的话，是没用用户体验的。所有开发一个图像化的小窗口，就是必要的。

对于稍有GUI编程经验的人来说，Python的Tkinter界面库是非常简单的。python的GUI库非常多，选择

目录导航

扩大
缩小

1 音乐1 広橋真紀子

2 音乐2 Thomas

3 音乐3 林海

4 音乐4 赵海洋

昵称: 洪卫
园龄: 1年
粉丝: 169
关注: 11
+加关注

随笔分类

01. 博客目录(1)

02. 知识框架(1)

03. Linux相关(2)

04. Python基础(6)

05. Python进阶

06. 项目实战

07. 人工智能(13)

08. 机器学习笔记(8)

09. 学习资源(1)

10. 阅读人生(6)

11. 生活随笔(1)

12. 计算机视觉(4)

13. 博客美化(2)

14. Python学习笔记(2)

15. 其他随笔(1)

随笔档案

2019年1月 (1)

2018年9月 (1)

2018年8月 (3)

2018年6月 (3)

2018年5月 (19)

2018年4月 (6)

最新评论

1. Re:博客园美化教程第二篇----极致个性化你的专属博客（为博客添加背景音乐插件，调整页面布局等）
@BKKITO我用的一个开源音乐播放插件，对应把下面的配置参数改成true就是自动播放了...
--洪卫

2. Re:博客园美化教程第二篇----极致个性化你的专属博客（为博客添加背景音乐插件，调整页面布局等）
自动播放咋改
--BKKITO

3. Re:博客园美化教程第二篇----极致个性化你的专属博客（为博客添加背景音乐插件，调整页面布局等）
@BKKITO不用谢，欢迎一起交流。...
--洪卫

Tkinter，一是最为简单，二是自带库，不需下载安装，随时使用，三则是从需求出发，Python作为一种脚本语言，一种胶水语言，一般不会用它来开发复杂的桌面应用，它并不具备这方面的优势，使用Python，可以把它作为一个灵活的工具，而不是作为主要开发语言，那么在工作中，需要制作一个小工具，肯定是需要有界面的，不仅自己用，也能分享别人使用，在这种需求下，Tkinter是足够胜任的！

这篇文章主要做一个简单概述和实践编程，对于从没有接触过GUI的新手，在脑中树立一个基本的界面编程概念，同时自己也能学会如何简单的实现一些小的图形窗口功能。

对于Tkinter编程，可以用两个比喻来理解：

- 第一个，作画。我们都见过美术生写生的情景，先支一个画架，放上画板，蒙上画布，构思内容，用铅笔画草图，组织结构和比例，调色板调色，最后画笔勾勒。相应的，对应到tkinter编程，那么我们的显示屏就是支起来的画架，根窗体就是画板，在tkinter中则是Toplevel，画布就是tkinter中的容器（Frame），画板上可以放很多张画布（Convas），tkinter中的容器中也可以放很多个容器，绘画中的构图布局则是tkinter中的布局管理器（几何管理器），绘画的内容就是tkinter中的一个小组件，一幅画由许多元素构成，而我们的GUI界面，就是有一个个组件拼装起来的，它们就是widget。
- 第二个，我们小时候都玩过积木，只要发挥创意，相同的积木可以堆出各种造型。tkinter的组件也可以看做一个个积木，形状或许不同，其本质都是一样的，就是一个积木，不管它长什么样子，它始终就是积木！所以这些小组件都有许多共性，另外，个人认为，学习界面编程，最重要的不是一开始学习每个积木的样子，不是学习每个组件怎么用，而是这些组件该怎么放。初始学习中，怎么放远远比怎么用重要的多。网上有大量的文章资料，基本全是介绍组件怎么用的，对于怎么放，也就是tkinter中的布局管理器，都是一笔带过，这对初学者有点本末倒置，或许绝大部分是转载的原因吧，极少是自己真正写的。组件怎么用不是最迫切的，用到的时候再去了解也不迟，边用边学反而更好。因此我将专门写一章，详细介绍布局管理器的使用。

三、Tkinter 控件详细介绍

1. Tkinter 模块元素简述

4. Re:博客园美化教程第二篇---极致个性化你的专属博客（为博客添加背景音乐插件，调整页面布局等）
@洪卫神奇，谢谢...
--BKKITO
5. Re:博客园美化教程第二篇---极致个性化你的专属博客（为博客添加背景音乐插件，调整页面布局等）
@BKKITO直接把后缀名改成pdf就行了，还是音频编码格式，但是这样就可以上传博客园文件了，骚操作而已。...
--洪卫

阅读排行榜

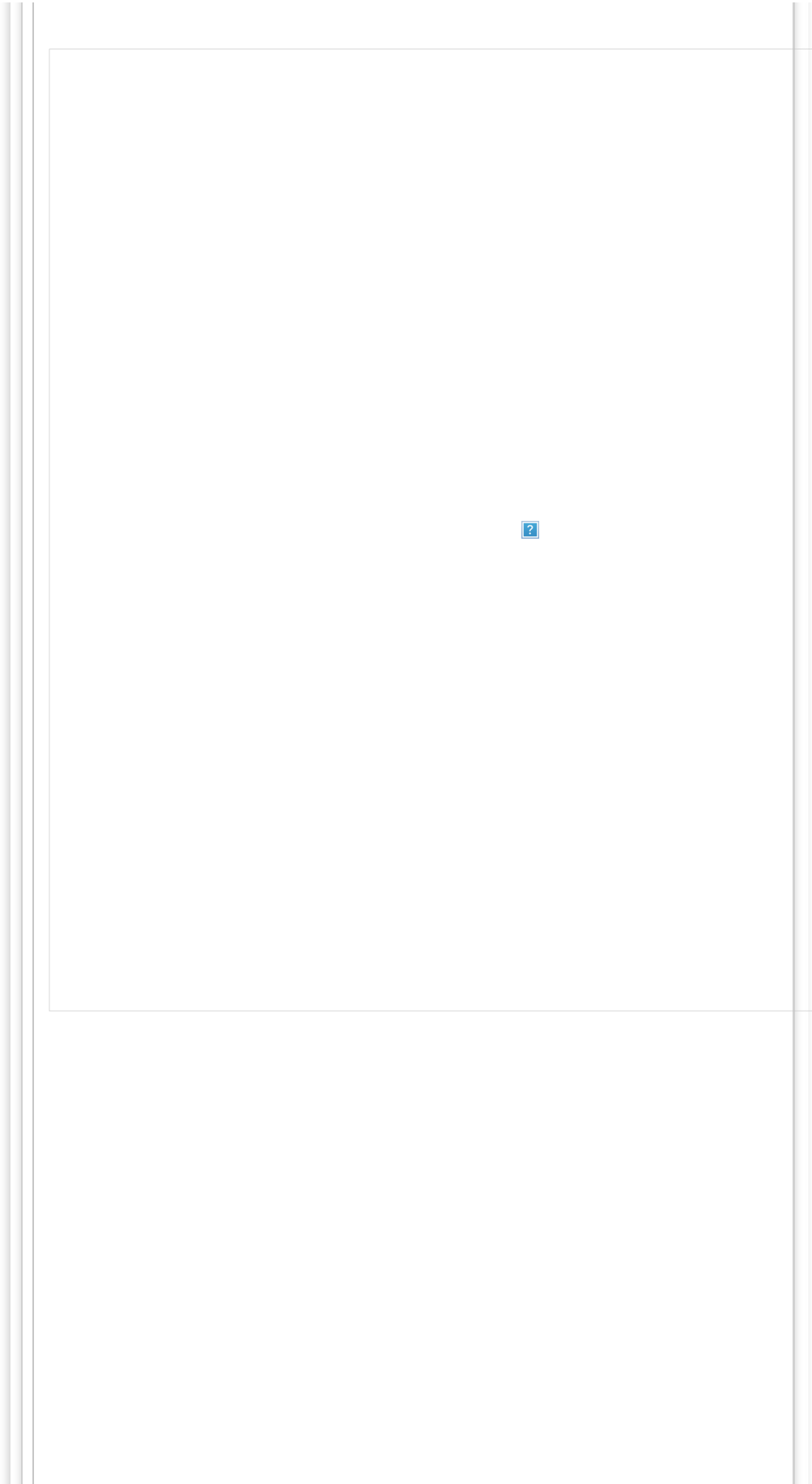
- 1. Python GUI之tkinter窗口视窗教程大集合（看这篇就够了）(18906)
- 2. 如何零基础开始自学Python编程(8544)
- 3. 博客园美化教程大集合---极致个性化你的专属博客（超详细，看这篇就够了）(4636)
- 4. 博客园美化教程第二篇---极致个性化你的专属博客（为博客添加背景音乐插件，调整页面布局等）(1956)
- 5. 约瑟夫坎贝尔 《千面英雄》阅读笔记(1743)

评论排行榜

- 1. 博客园美化教程大集合---极致个性化你的专属博客（超详细，看这篇就够了）(47)
- 2. 博客园美化教程第二篇---极致个性化你的专属博客（为博客添加背景音乐插件，调整页面布局等）(40)
- 3. Python GUI之tkinter窗口视窗教程大集合（看这篇就够了）(28)
- 4. 自控力极差的人如何自救？(2)
- 5. 万水千山总是情，看看这里行不行(2)

推荐排行榜

- 1. 博客园美化教程大集合---极致个性化你的专属博客（超详细，看这篇就够了）(118)
- 2. Python GUI之tkinter窗口视窗教程大集合（看这篇就够了）(39)
- 3. 博客园美化教程第二篇---极致个性化你的专属博客（为博客添加背景音乐插件，调整页面布局等）(26)
- 4. 非技术人员也能看懂云计算，大数据，人工智能(7)
- 5. 如何零基础开始自学Python编程(7)



2. 常用窗口部件简要描述:

Tkinter支持16个核心的窗口部件，这个16个核心窗口部件类简要描述如下：

Button：一个简单的按钮，用来执行一个命令或别的操作。

Canvas：组织图形。这个部件可以用来绘制图表和图，创建图形编辑器，实现定制窗口部件。

Checkbutton：代表一个变量，它有两个不同的值。点击这个按钮将会在这两个值间切换。

Entry：文本输入域。

Frame：一个容器窗口部件。帧可以有边框和背景，当创建一个应用程序或dialog(对话) 版面时，帧被用来组织其它的窗口部件。

Label：显示一个文本或图象。

Listbox：显示供选方案的一个列表。listbox能够被配置来得到radiobutton或checklist的行为。

Menu：菜单条。用来实现下拉和弹出式菜单。

Menubutton：菜单按钮。用来实现下拉式菜单。

Message：显示一文本。类似label窗口部件，但是能够自动地调整文本到给定的宽度或比率。

Radiobutton：代表一个变量，它可以有多个值中的一个。点击它将为这个变量设置值，并且清除与同一变量相关的其它radiobutton。

Scale：允许你通过滑块来设置一数字值。

Scrollbar：为配合使用canvas, entry, listbox, and text窗口部件的标准滚动条。

Text：格式化文本显示。允许你用不同的样式和属性来显示和编辑文本。同时支持内嵌图象和窗口。

Toplevel：一个容器窗口部件，作为一个单独的、最上面的窗口显示。

messageBox：消息框，用于显示你应用程序的消息框。(Python2中为tkMessageBox)

注意在Tkinter中窗口部件类没有分级；所有的窗口部件类在树中都是兄弟关系。

所有这些窗口部件提供了Misc和几何管理方法、配置管理方法和部件自己定义的另外的方法。此外，Toplevel类也提供窗口管理接口。这意味一个典型的窗口部件类提供了大约150种方法。

四、动手实践学习

1. 创建主窗口及Label部件（标签）创建使用

我们要学习使用上面提到的这些控件首先要创建一个主窗口，就像作画一样，先要架好架子和画板，然后才能在上面放画纸和各种绘画元素，创建好主窗口才能在上面放置各种控件元素。而创建过程是很简单的，如下：

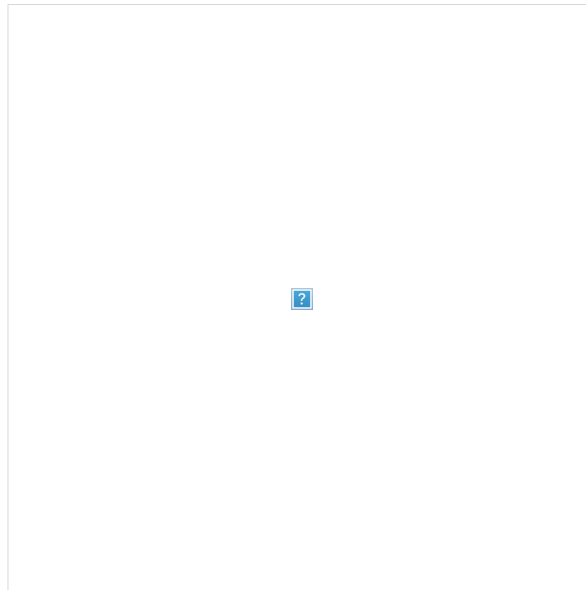
示例代码：

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



```
2  # -*- coding: utf-8 -*-
3  # author:洪卫
4
5  import tkinter as tk # 使用Tkinter前需要先导入
6
7  # 第1步, 实例化object, 建立窗口window
8  window = tk.Tk()
9
10 # 第2步, 给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第3步, 设定窗口的大小(长 * 宽)
14 window.geometry('500x300') # 这里的乘是小x
15
16 # 第4步, 在图形界面上设定标签
17 l = tk.Label(window, text='你好! this is Tkinter', bg='green', font=('Arial', 12), width=30,
18 # 说明: bg为背景, font为字体, width为长, height为高, 这里的长和高是字符的长和高, 比如height=2,就是标签有
19
20 # 第5步, 放置标签
21 l.pack() # Label内容content区域放置位置, 自动调节尺寸
22 # 放置lable的方法有: 1) l.pack(); 2)l.place();
23
24 # 第6步, 主窗口循环显示
25 window.mainloop()
26 # 注意, loop因为是循环的意思, window.mainloop就会让window不断的刷新, 如果没有mainloop, 就是一个静态的wi
27 # 所有的窗口文件都必须有类似的主mainloop函数, mainloop是窗口文件的关键的关键。
```

测试效果:



2. Button窗口部件

简单说明:

Button (按钮) 部件是一个标准的Tkinter窗口部件, 用来实现各种按钮。按钮能够包含文本或图象, 并且你能够将按钮与一个Python函数或方法相关联。当这个按钮被按下时, Tkinter自动调用相关联的函数或方法。

按钮仅能显示一种字体, 但是这个文本可以跨行。另外, 这个文本中的一个字母可以有下划线, 例如标明一个快捷键。默认情况, Tab键用于将焦点移动到一个按钮部件。

什么时候用按钮部件

简言之, 按钮部件用来让用户说“马上给我执行这个任务”, 通常我们用显示在按钮上的文本或图象来提示。按钮通常用在工具条中或应用程序窗口中, 并且用来接收或忽略输入在对话框中的数据。关于按钮和输入的数据的配合, 可以参看Checkbutton和Radiobutton部件。

如何创建：

普通的按钮很容易被创建，仅仅指定按钮的内容（文本、位图、图象）和一个当按钮被按下时的回调函数即可：

```
b = tk.Button(window, text="hit me", command=hit_me)
```

没有回调函数的按钮是没有用的，当你按下这个按钮时它什么也不做。你可能在开发一个应用程序的时候想实现这种按钮，比如为了不干扰你的beta版的测试者：

```
b = tk.Button(window, text="Help", command=DISABLED)
```

示例代码：

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # author:洪卫
4
5  import tkinter as tk # 使用Tkinter前需要先导入
6
7  # 第1步，实例化Object，建立窗口window
8  window = tk.Tk()
9
10 # 第2步，给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第3步，设定窗口的大小(长 * 宽)
14 window.geometry('500x300') # 这里的乘是小x
15
16 # 第4步，在图形界面上设定标签
17 var = tk.StringVar() # 将label标签的内容设置为字符类型，用var来接收hit_me函数的传出内容用以显示在标
18 l = tk.Label(window, textvariable=var, bg='green', fg='white', font=('Arial', 12), width=30,
19 # 说明： bg为背景，fg为字体颜色，font为字体，width为长，height为高，这里的长和高是字符的长和高，比如height
20 l.pack()
21
22 # 定义一个函数功能（内容自己自由编写），供点击Button按键时调用，调用命令参数command=函数名
23 on_hit = False
24 def hit_me():
25     global on_hit
26     if on_hit == False:
27         on_hit = True
28         var.set('you hit me')
29     else:
30         on_hit = False
31         var.set('')
32
33 # 第5步，在窗口界面设置放置Button按键
34 b = tk.Button(window, text='hit me', font=('Arial', 12), width=10, height=1, command=hit_me)
35 b.pack()
36
37 # 第6步，主窗口循环显示
38 window.mainloop()
```

测试效果：



3. Entry窗口部件

简单说明:

Entry是tkinter类中提供的的一个单行文本输入域, 用来输入显示一行文本, 收集键盘输入(类似 HTML 中的 text)。

什么时候用:

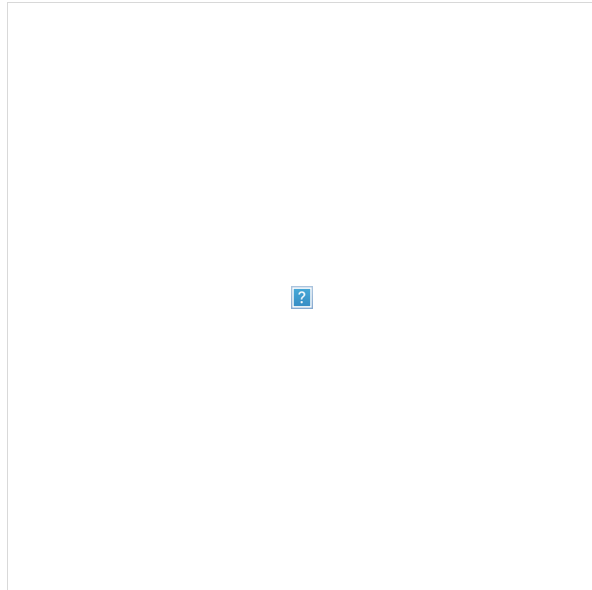
需要用户输入用户信息时, 比如我们平时使用软件、登录网页时, 用户交互界面让我们登录账户信息等时候可以用到。

示例代码:

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # author:洪卫
4
5  import tkinter as tk # 使用Tkinter前需要先导入
6
7  # 第1步, 实例化object, 建立窗口window
8  window = tk.Tk()
9
```

```
10 # 第二步, 给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第三步, 设定窗口的大小(长 * 宽)
14 window.geometry('500x300') # 这里的乘是小x
15
16 # 第四步, 在图形界面上设定输入框控件entry并放置控件
17 e1 = tk.Entry(window, show='*', font=('Arial', 14)) # 显示成密文形式
18 e2 = tk.Entry(window, show=None, font=('Arial', 14)) # 显示成明文形式
19 e1.pack()
20 e2.pack()
21
22 # 第五步, 主窗口循环显示
23 window.mainloop()
```

测试效果:



4 Text窗口部件

简单说明:

Text是tkinter类中提供的的一个多行文本区域, 显示多行文本, 可用来收集(或显示)用户输入的文字(类似HTML 中的 textarea), 格式化文本显示, 允许你用不同的样式和属性来显示和编辑文本, 同时支持内嵌图象和窗口。

什么时候用:

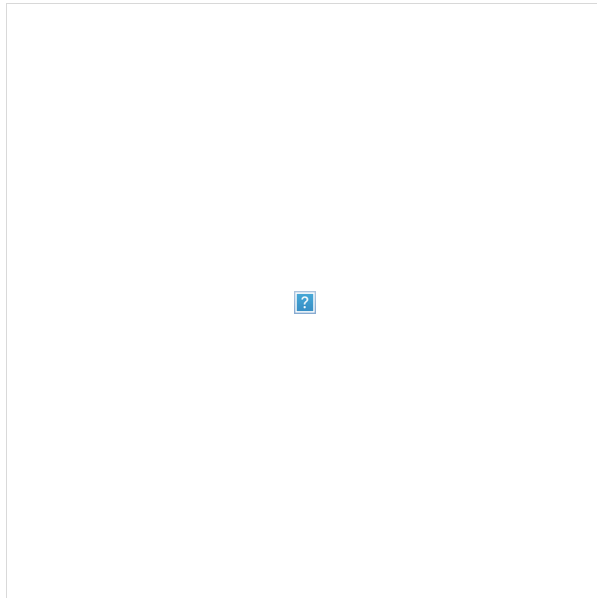
在需要显示编辑用户、产品多行信息时, 比如显示用户详细描述文字, 产品简介等等, 支持随时编辑。

示例代码:

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 # author:洪卫
4
5 import tkinter as tk # 使用Tkinter前需要先导入
6
7 # 第一步, 实例化Object, 建立窗口window
8 window = tk.Tk()
9
10 # 第二步, 给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第三步, 设定窗口的大小(长 * 宽)
```

```
14 window.geometry('500x300') # 这里的乘是小x
15
16 # 第4步, 在图形界面上设定输入框控件entry框并放置
17 e = tk.Entry(window, show = None) #显示成明文形式
18 e.pack()
19
20 # 第5步, 定义两个触发事件时的函数insert_point和insert_end (注意: 因为Python的执行顺序是从上往下, 所以函数
21 def insert_point(): # 在鼠标焦点处插入输入内容
22     var = e.get()
23     t.insert('insert', var)
24 def insert_end(): # 在文本框内容最后接着插入输入内容
25     var = e.get()
26     t.insert('end', var)
27
28 # 第6步, 创建并放置两个按钮分别触发两种情况
29 b1 = tk.Button(window, text='insert point', width=10,
30                height=2, command=insert_point)
31 b1.pack()
32 b2 = tk.Button(window, text='insert end', width=10,
33                height=2, command=insert_end)
34 b2.pack()
35
36 # 第7步, 创建并放置一个多行文本框text用以显示, 指定height=3为文本框是三个字符高度
37 t = tk.Text(window, height=3)
38 t.pack()
39
40 # 第8步, 主窗口循环显示
41 window.mainloop()
```

测试效果:



5. Listbox窗口部件

简单说明:

Text是tkinter类中提供的列表框部件, 显示供选方案的一个列表。listbox能够被配置来得到radiobutton或checklist的行为。

什么时候用:

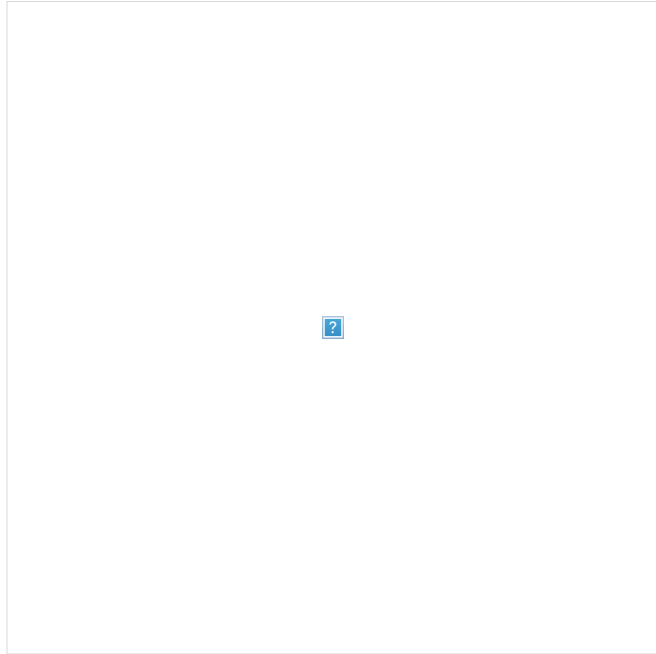
在有一个很多内容选项组成的列表提供用户选择时会用到。

示例代码:

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

```
2  # -*- coding: utf-8 -*-
3  # author:洪卫
4
5  import tkinter as tk # 使用Tkinter前需要先导入
6
7  # 第1步, 实例化object, 建立窗口window
8  window = tk.Tk()
9
10 # 第2步, 给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第3步, 设定窗口的大小(长 * 宽)
14 window.geometry('500x300') # 这里的乘是小x
15
16 # 第4步, 在图形界面上创建一个标签label用以显示并放置
17 var1 = tk.StringVar() # 创建变量, 用var1用来接收鼠标点击具体选项的内容
18 l = tk.Label(window, bg='green', fg='yellow', font=('Arial', 12), width=10, textvariable=var1)
19 l.pack()
20
21 # 第6步, 创建一个方法用于按钮的点击事件
22 def print_selection():
23     value = lb.get(lb.curselection()) # 获取当前选中的文本
24     var1.set(value) # 为label设置值
25
26 # 第5步, 创建一个按钮并放置, 点击按钮调用print_selection函数
27 b1 = tk.Button(window, text='print selection', width=15, height=2, command=print_selection)
28 b1.pack()
29
30 # 第7步, 创建Listbox并为其添加内容
31 var2 = tk.StringVar()
32 var2.set((1,2,3,4)) # 为变量var2设置值
33 # 创建Listbox
34 lb = tk.Listbox(window, listvariable=var2) # 将var2的值赋给Listbox
35 # 创建一个list并将值循环添加到Listbox控件中
36 list_items = [11,22,33,44]
37 for item in list_items:
38     lb.insert('end', item) # 从最后一个位置开始加入值
39 lb.insert(1, 'first') # 在第一个位置加入'first'字符
40 lb.insert(2, 'second') # 在第二个位置加入'second'字符
41 lb.delete(2) # 删除第二个位置的字符
42 lb.pack()
43
44 # 第8步, 主窗口循环显示
45 window.mainloop()
```

测试效果:



9. Radiobutton窗口部件

简单说明:

Radiobutton: 代表一个变量，它可以有多个值中的一个。点击它将为这个变量设置值，并且清除与这同一变量相关的其它radiobutton。

什么时候用:

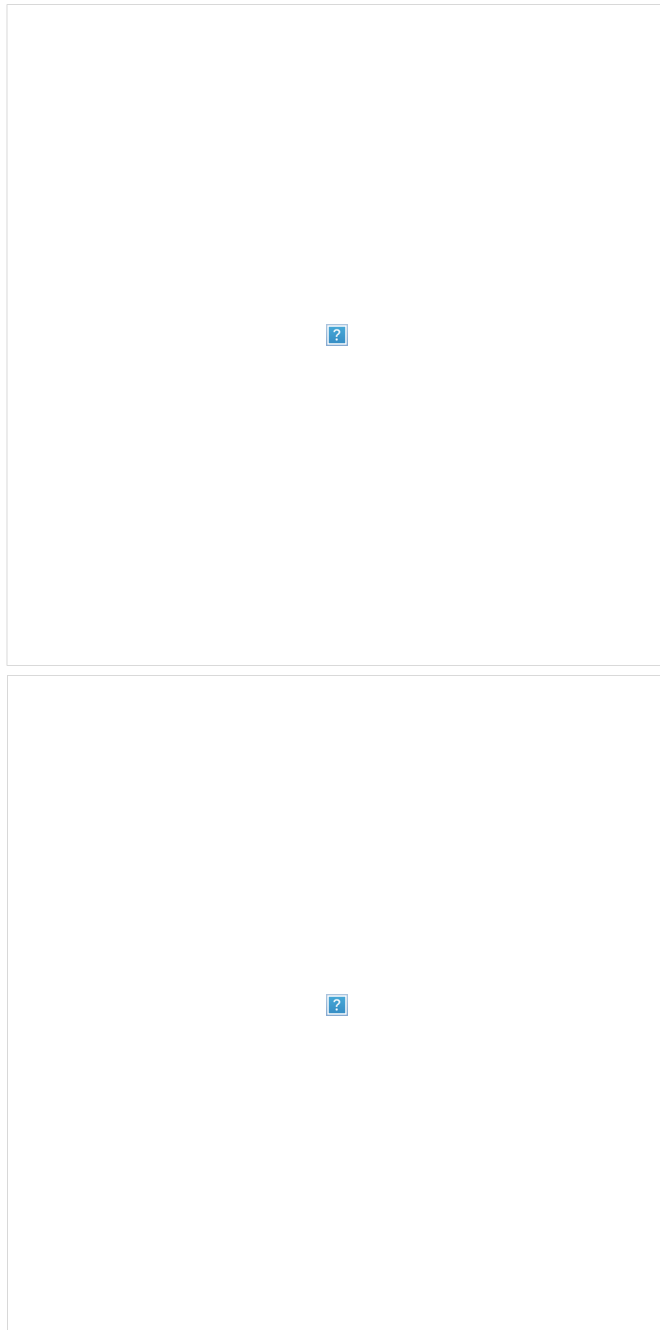
在有一个很多内容选项组成的选项列表提供用户选择时会用到，用户一次只能选择其中一个，不能多选。

示例代码:

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # author:洪卫
4
5  import tkinter as tk # 使用Tkinter前需要先导入
6
7  # 第1步, 实例化object, 建立窗口window
8  window = tk.Tk()
9
10 # 第2步, 给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第3步, 设定窗口的大小(长 * 宽)
14 window.geometry('500x300') # 这里的乘是小x
15
16 # 第4步, 在图形界面上创建一个标签label用以显示并放置
17 var = tk.StringVar() # 定义一个var用来将radiobutton的值和Label的值联系在一起.
18 l = tk.Label(window, bg='yellow', width=20, text='empty')
19 l.pack()
20
21 # 第6步, 定义选项触发函数功能
22 def print_selection():
23     l.config(text='you have selected ' + var.get())
24
25 # 第5步, 创建三个radiobutton选项, 其中variable=var, value='A'的意思就是, 当我们鼠标选中了其中一个选项,
26 r1 = tk.Radiobutton(window, text='Option A', variable=var, value='A', command=print_selection)
27 r1.pack()
28 r2 = tk.Radiobutton(window, text='Option B', variable=var, value='B', command=print_selection)
29 r2.pack()
```

```
30 r3 = tk.Radiobutton(window, text='Option C', variable=var, value='C', command=print_selectio
31 r3.pack()
32
33 # 第7步, 主窗口循环显示
34 window.mainloop()
```

测试效果:



7. Checkbutton窗口按钮

简单说明:

Checkbutton: 代表一个变量, 它有两个不同的值。点击这个按钮将会在这两个值间切换, 选择和取消选择。

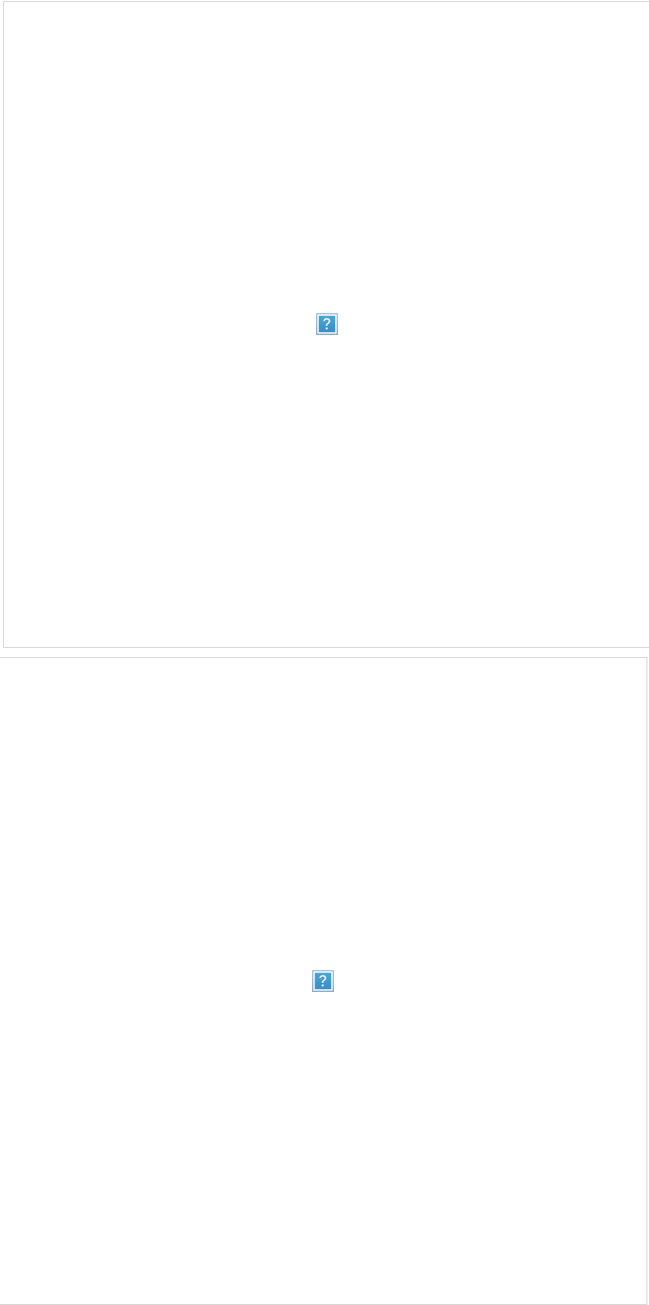
什么时候用:

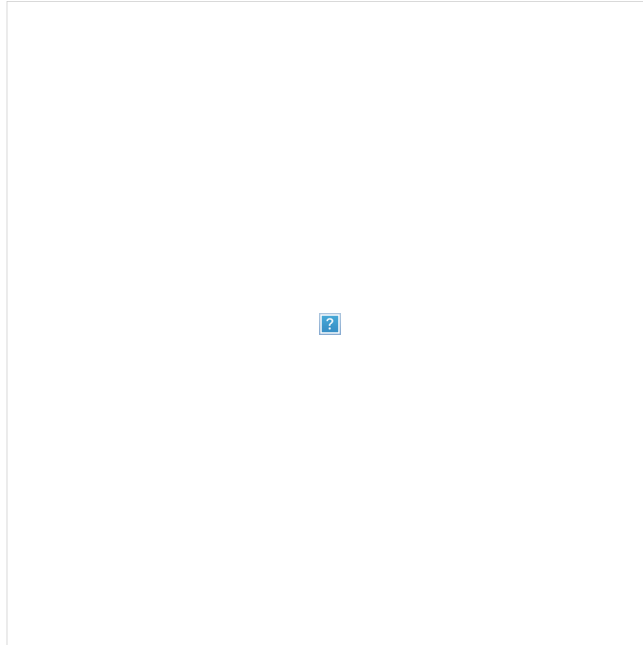
在有一个很多内容选项组成的选项列表提供用户选择时会用到，用户一次可以选择多个。

示例代码：

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # author:洪卫
4
5  import tkinter as tk # 使用Tkinter前需要先导入
6
7  # 第1步, 实例化object, 建立窗口window
8  window = tk.Tk()
9
10 # 第2步, 给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第3步, 设定窗口的大小(长 * 宽)
14 window.geometry('500x300') # 这里的乘是小x
15
16 # 第4步, 在图形界面上创建一个标签label用以显示并放置
17 l = tk.Label(window, bg='yellow', width=20, text='empty')
18 l.pack()
19
20 # 第6步, 定义触发函数功能
21 def print_selection():
22     if (var1.get() == 1) & (var2.get() == 0): # 如果选中第一个选项, 未选中第二个选项
23         l.config(text='I love only Python ')
24     elif (var1.get() == 0) & (var2.get() == 1): # 如果选中第二个选项, 未选中第一个选项
25         l.config(text='I love only C++')
26     elif (var1.get() == 0) & (var2.get() == 0): # 如果两个选项都未选中
27         l.config(text='I do not love either')
28     else:
29         l.config(text='I love both') # 如果两个选项都选中
30
31 # 第5步, 定义两个Checkbutton选项并放置
32 var1 = tk.IntVar() # 定义var1和var2整型变量用来存放选择行为返回值
33 var2 = tk.IntVar()
34 c1 = tk.Checkbutton(window, text='Python', variable=var1, onvalue=1, offvalue=0, command=print_selection)
35 c1.pack()
36 c2 = tk.Checkbutton(window, text='C++', variable=var2, onvalue=1, offvalue=0, command=print_selection)
37 c2.pack()
38
39 # 第7步, 主窗口循环显示
40 window.mainloop()
```

测试效果：





Scale窗口部件

简单说明:

Scale: 尺度 (拉动条), 允许你通过滑块来设置一数值。

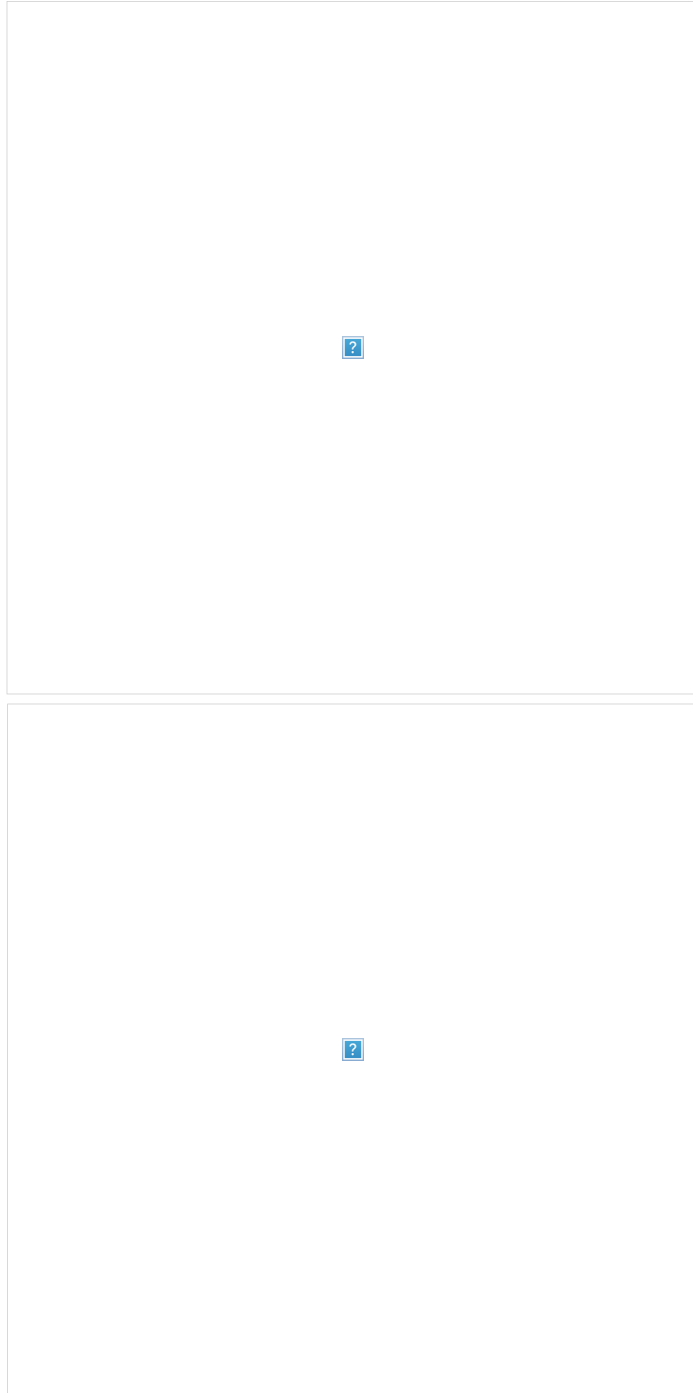
什么时候用:

在需要用户给出评价等级, 或者给出一个评价分数, 或者拉动滑动条提供一个具体的数值等等。

示例代码:

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # author:洪卫
4
5  import tkinter as tk # 使用Tkinter前需要先导入
6
7  # 第1步, 实例化Object, 建立窗口window
8  window = tk.Tk()
9
10 # 第2步, 给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第3步, 设定窗口的大小(长 * 宽)
14 window.geometry('500x300') # 这里的乘是小x
15
16 # 第4步, 在图形界面上创建一个标签label用以显示并放置
17 l = tk.Label(window, bg='green', fg='white', width=20, text='empty')
18 l.pack()
19
20 # 第5步, 定义一个触发函数功能
21 def print_selection(v):
22     l.config(text='you have selected ' + v)<br>
23 # 第6步, 创建一个尺度滑条, 长度200字符, 从0开始10结束, 以2为刻度, 精度为0.01, 触发调用print_selection函数
24 s = tk.Scale(window, label='try me', from_=0, to=10, orient=tk.HORIZONTAL, length=200, showv
25 s.pack()
26
27 # 第7步, 主窗口循环显示
28 window.mainloop()
```

测试效果:



9. Canvas窗口部件

简单说明：

Canvas：画布，提供绘图功能(直线、椭圆、多边形、矩形) 可以包含图形或位图，用来绘制图表和图，创建图形编辑器，实现定制窗口部件。

什么时候用：

在比如像用户交互界面等，需要提供设计的图标、图形、logo等信息是可以用到画布。

示例代码：

```
1 | #!/usr/bin/env python
2 | # -*- coding: utf-8 -*-
```

```
3 # author:洪卫
4
5 import tkinter as tk # 使用Tkinter前需要先导入
6
7 # 第1步, 实例化Object, 建立窗口window
8 window = tk.Tk()
9
10 # 第2步, 给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第3步, 设定窗口的大小(长 * 宽)
14 window.geometry('500x300') # 这里的乘是小x
15
16 # 第4步, 在图形界面上创建 500 * 200 大小的画布并放置各种元素
17 canvas = tk.Canvas(window, bg='green', height=200, width=500)
18 # 说明图片位置, 并导入图片到画布上
19 image_file = tk.PhotoImage(file='pic.gif') # 图片位置 (相对路径, 与.py文件同一文件夹下, 也可以用绝对
20 image = canvas.create_image(250, 0, anchor='n', image=image_file) # 图片锚定点 (n图片顶端
21 # 定义多边形参数, 然后在画布上画出指定图形
22 x0, y0, x1, y1 = 100, 100, 150, 150
23 line = canvas.create_line(x0-50, y0-50, x1-50, y1-50) # 画直线
24 oval = canvas.create_oval(x0+120, y0+50, x1+120, y1+50, fill='yellow') # 画圆 用黄色填充
25 arc = canvas.create_arc(x0, y0+50, x1, y1+50, start=0, extent=180) # 画扇形 从0度打开收到1
26 rect = canvas.create_rectangle(330, 30, 330+20, 30+20) # 画矩形正方形
27 canvas.pack()
28
29 # 第6步, 触发函数, 用来一定指定图形
30 def moveit():
31     canvas.move(rect, 2, 2) # 移动正方形rect (也可以改成其他图形名字用以移动一起图形、元素), 按每次 (x
32
33 # 第5步, 定义一个按钮用来移动指定图形的在画布上的位置
34 b = tk.Button(window, text='move item', command=moveit).pack()
35
36 # 第7步, 主窗口循环显示
37 window.mainloop()
```

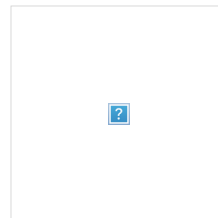
所用图片:

当然你可以随意用你的一张图片导入画布试一试效果, 图片可以用画图工具改一下像素大小, 以免图片太大, 导入画布显示不全, 当然你也可以用我提供的素材, 下面是链接:

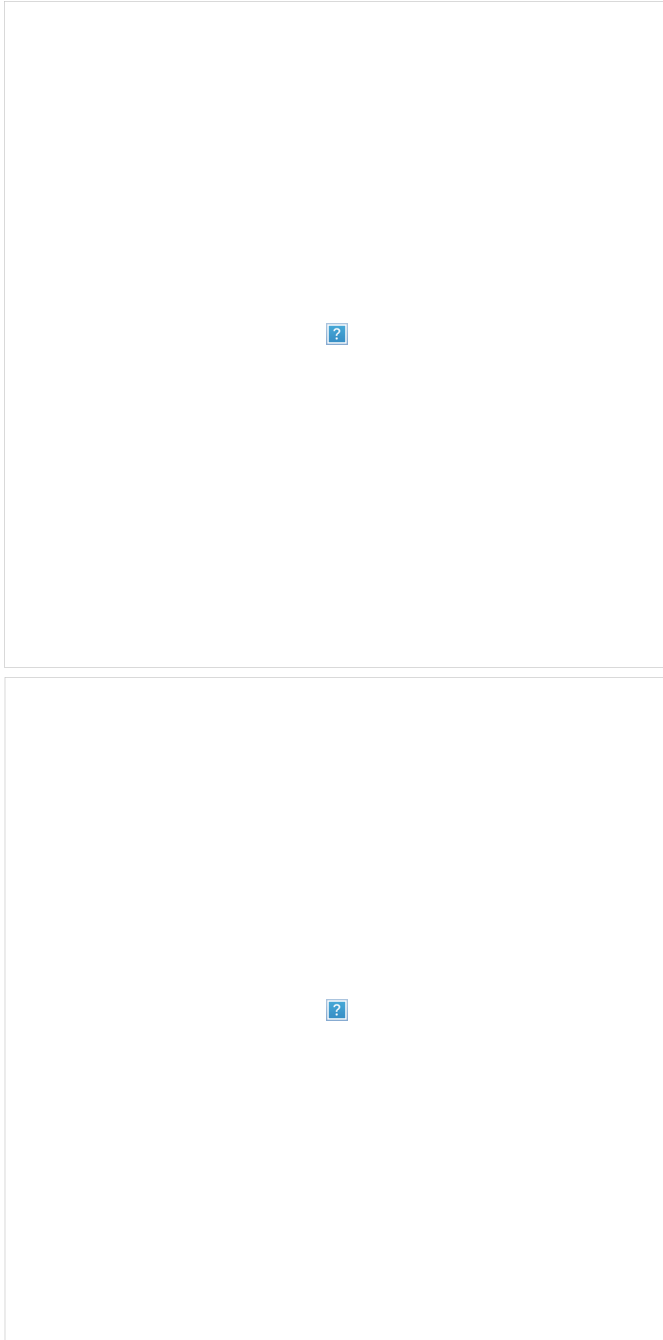
<https://files.cnblogs.com/files/shwee/pic.gif>



图片锚定点位置参数图:



测试效果:



18. Menu窗口部件

简单说明:

Menu: 菜单条, 用来实现下拉和弹出式菜单, 点下菜单后弹出的一个选项列表, 用户可以从中选择

什么时候用:

在比如像软件或网页交互界面等, 需要提供菜单选项功能提供用户选择菜单选项功能时用到。

示例代码:

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # author:洪卫
4
5  import tkinter as tk # 使用Tkinter前需要先导入
```

```
6
7 # 第1步, 实例化object, 建立窗口window
8 window = tk.Tk()
9
10 # 第2步, 给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第3步, 设定窗口的大小(长 * 宽)
14 window.geometry('500x300') # 这里的乘是小x
15
16 # 第4步, 在图形界面上创建一个标签用以显示内容并放置
17 l = tk.Label(window, text='      ', bg='green')
18 l.pack()
19
20 # 第10步, 定义一个函数功能, 用来代表菜单选项的功能, 这里为了操作简单, 定义的功能比较简单
21 counter = 0
22 def do_job():
23     global counter
24     l.config(text='do ' + str(counter))
25     counter += 1
26
27 # 第5步, 创建一个菜单栏, 这里我们可以把他理解成一个容器, 在窗口的上方
28 menubar = tk.Menu(window)
29
30 # 第6步, 创建一个File菜单项 (默认不下拉, 下拉内容包括New, Open, Save, Exit功能项)
31 filemenu = tk.Menu(menubar, tearoff=0)
32 # 将上面定义的空菜单命名为File, 放在菜单栏中, 就是装入那个容器中
33 menubar.add_cascade(label='File', menu=filemenu)
34
35 # 在File中加入New、Open、Save等小菜单, 即我们平时看到的下拉菜单, 每一个小菜单对应命令操作。
36 filemenu.add_command(label='New', command=do_job)
37 filemenu.add_command(label='Open', command=do_job)
38 filemenu.add_command(label='Save', command=do_job)
39 filemenu.add_separator() # 添加一条分隔线
40 filemenu.add_command(label='Exit', command=window.quit) # 用tkinter里面自带的quit()函数
41
42 # 第7步, 创建一个Edit菜单项 (默认不下拉, 下拉内容包括Cut, Copy, Paste功能项)
43 editmenu = tk.Menu(menubar, tearoff=0)
44 # 将上面定义的空菜单命名为 Edit, 放在菜单栏中, 就是装入那个容器中
45 menubar.add_cascade(label='Edit', menu=editmenu)
46
47 # 同样的在 Edit 中加入Cut、Copy、Paste等小命令功能单元, 如果点击这些单元, 就会触发do_job的功能
48 editmenu.add_command(label='Cut', command=do_job)
49 editmenu.add_command(label='Copy', command=do_job)
50 editmenu.add_command(label='Paste', command=do_job)
51
52 # 第8步, 创建第二级菜单, 即菜单项里面的菜单
53 submenu = tk.Menu(filemenu) # 和上面定义菜单一样, 不过此处实在File上创建一个空的菜单
54 filemenu.add_cascade(label='Import', menu=submenu, underline=0) # 给放入的菜单submenu命名为Import
55
56 # 第9步, 创建第三级菜单命令, 即菜单项里面的菜单项里面的菜单命令 (有点拗口, 笑~~~)
57 submenu.add_command(label='Submenu_1', command=do_job) # 这里和上面创建原理也一样, 在Import菜单
58
59 # 第11步, 创建菜单栏完成后, 配置让菜单栏menubar显示出来
60 window.config(menu=menubar)
61
62 # 第12步, 主窗口循环显示
63 window.mainloop()
```

测试效果:



14. Frame 窗口部件

简单说明:

Frame: 框架, 用来承载放置其他GUI元素, 就是一个容器, 是一个在 Windows 上分离小区域的部件, 它能够将 Windows 分成不同的区, 然后存放不同的其他部件. 同时一个 Frame 上也能再分成两个 Frame, Frame 可以认为是一种容器.

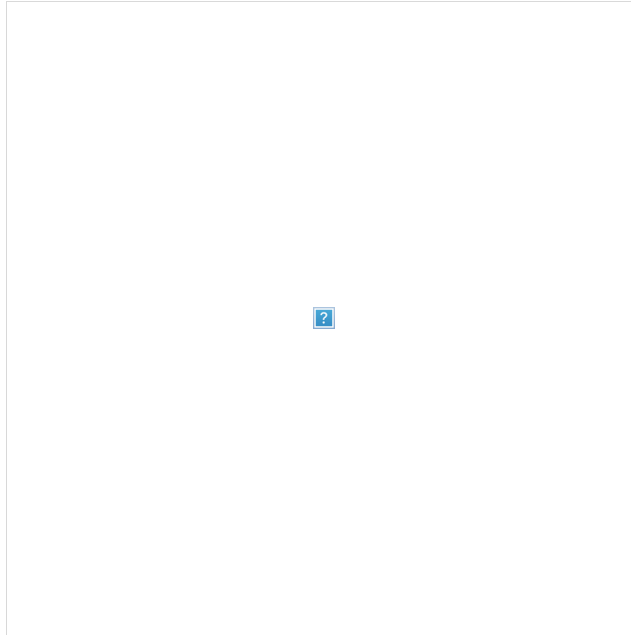
什么时候用:

在比如像软件或网页交互界面等, 有不同的界面逻辑层级和功能区域划分时可以用到, 让交互界面逻辑更加清晰.

示例代码:

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # author:洪卫
4
5  import tkinter as tk # 使用Tkinter前需要先导入
6
7  # 第1步, 实例化Object, 建立窗口window
8  window = tk.Tk()
9
10 # 第2步, 给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第3步, 设定窗口的大小(长 * 宽)
14 window.geometry('500x300') # 这里的乘是小x
15
16 # 第4步, 在图形界面上创建一个标签用以显示内容并放置
17 tk.Label(window, text='on the window', bg='red', font=('Arial', 16)).pack() # 和前面部件分开
18
19 # 第5步, 创建一个主frame, 长在主window窗口上
20 frame = tk.Frame(window)
21 frame.pack()
22
23 # 第6步, 创建第二层框架frame, 长在主框架frame上面
24 frame_l = tk.Frame(frame) # 第二层frame, 左frame, 长在主frame上
25 frame_r = tk.Frame(frame) # 第二层frame, 右frame, 长在主frame上
26 frame_l.pack(side='left')
27 frame_r.pack(side='right')
28
29 # 第7步, 创建三组标签, 为第二层frame上面的内容, 分为左区域和右区域, 用不同颜色标识
30 tk.Label(frame_l, text='on the frame_l1', bg='green').pack()
31 tk.Label(frame_l, text='on the frame_l2', bg='green').pack()
32 tk.Label(frame_l, text='on the frame_l3', bg='green').pack()
33 tk.Label(frame_r, text='on the frame_r1', bg='yellow').pack()
34 tk.Label(frame_r, text='on the frame_r2', bg='yellow').pack()
35 tk.Label(frame_r, text='on the frame_r3', bg='yellow').pack()
36
37 # 第8步, 主窗口循环显示
38 window.mainloop()
```

测试效果:



18. messagebox窗口部件

简单说明:

messageBox: 消息框, 用于显示你应用程序的消息框。(Python2中为tkMessageBox), 其实这里的messageBox就是我们平时看到的弹窗。我们首先需要定义一个触发功能, 来触发这个弹窗, 这里我们就放上以前学过的button按钮, 通过触发功能, 调用messagebox吧, 点击button按钮就会弹出提示对话框。下面给出messagebox提示信息的几种形式:

```
1 tkinter.messagebox.showinfo(title='Hi', message='你好! ') # 提示信息对话框
2 tkinter.messagebox.showwarning(title='Hi', message='有警告! ') # 提出警告对话框
3 tkinter.messagebox.showerror(title='Hi', message='出错了! ') # 提出错误对话框
4 print(tkinter.messagebox.askquestion(title='Hi', message='你好! ')) # 询问选择对话框return 'yes'
5 print(tkinter.messagebox.askyesno(title='Hi', message='你好! ')) # return 'True', 'False'
6 print(tkinter.messagebox.askokcancel(title='Hi', message='你好! ')) # return 'True', 'False'
```

什么时候用:

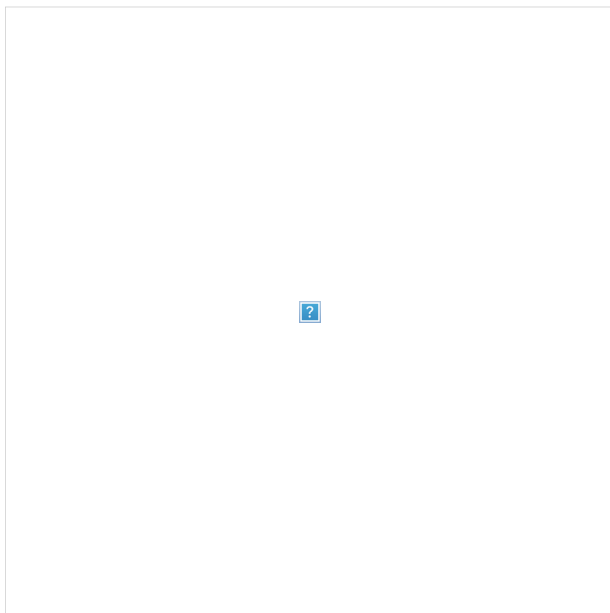
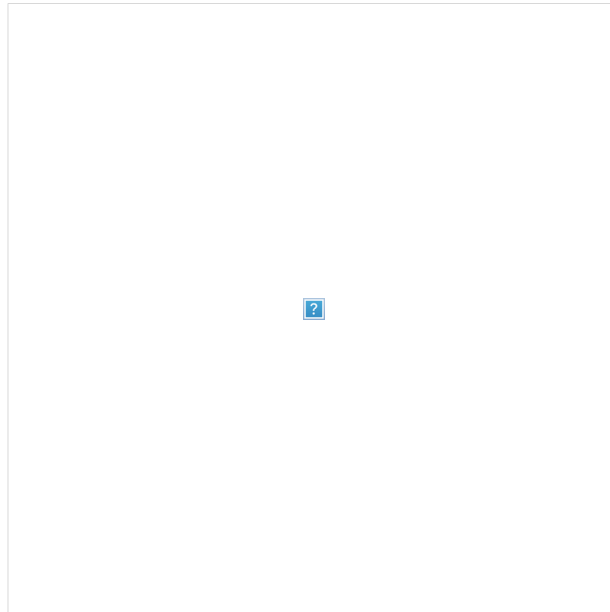
在比如像软件或网页交互界面等, 有不同的界面逻辑层级和功能区域划分时可以用到, 让交互界面逻辑更加清晰。

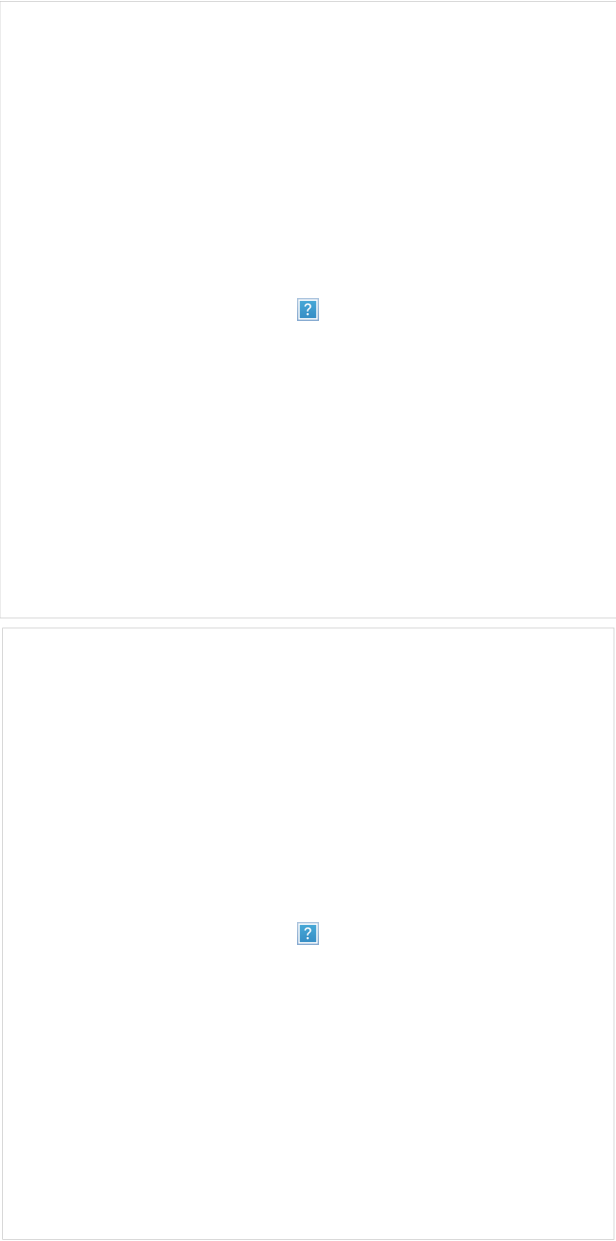
示例代码:

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 # author:洪卫
4
5 import tkinter as tk # 使用Tkinter前需要先导入
6 import tkinter.messagebox # 要使用messagebox先要导入模块
7
8 # 第1步, 实例化object, 建立窗口window
9 window = tk.Tk()
10
11 # 第2步, 给窗口的可视化起名字
12 window.title('My Window')
13
14 # 第3步, 设定窗口的大小(长 * 宽)
15 window.geometry('500x300') # 这里的乘是小x
16
17 # 第5步, 定义触发函数功能
18 def hit_me():
19     tkinter.messagebox.showinfo(title='Hi', message='你好! ') # 提示信息对话框
```

```
20     # tkinter.messagebox.showwarning(title='Hi', message='有警告! ')      # 提出警告对话框
21     # tkinter.messagebox.showerror(title='Hi', message='出错了! ')      # 提出错误对话框
22     # print(tkinter.messagebox.askquestion(title='Hi', message='你好! ')) # 询问选择对话框retu
23     # print(tkinter.messagebox.askyesno(title='Hi', message='你好! '))   # return 'True', '
24     # print(tkinter.messagebox.askokcancel(title='Hi', message='你好! ')) # return 'True', '
25
26     # 第4步, 在图形界面上创建一个标签用以显示内容并放置
27     tk.Button(window, text='hit me', bg='green', font=('Arial', 14), command=hit_me).pack()
28
29     # 第6步, 主窗口循环显示
30     window.mainloop()
```

测试效果:





19. 窗口部件三种放置方式 pack/grid/place

参考来源：

- [The Grid Geometry Manager](#)
- [The Pack Geometry Manager](#)
- [The Place Geometry Manager](#)

1. Grid: The Grid Geometry Manager

grid 是方格，所以所有的内容会被放在这些规律的方格中。例如：

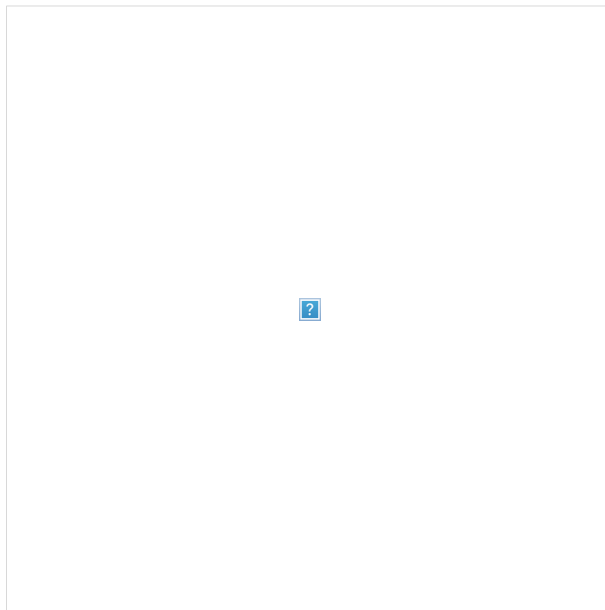
```
1 for i in range(3):
2     for j in range(3):
3         tk.Label(window, text=1).grid(row=i, column=j, padx=10, pady=10, ipadx=10, ipady=10)
```

以上的代码就是创建一个三行三列的表格，其实 grid 就是用表格的形式定位的。这里的参数 row 为行，column 为列，padx 就是单元格左右间距，pady 就是单元格上下间距，ipadx是单元格内部元素与单元格的左右间距，ipady是单元格内部元素与单元格的上下间距。

示例代码:

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # author:洪卫
4
5  import tkinter as tk # 使用Tkinter前需要先导入
6
7  # 第1步, 实例化object, 建立窗口window
8  window = tk.Tk()
9
10 # 第2步, 给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第3步, 设定窗口的大小(长 * 宽)
14 window.geometry('500x300') # 这里的乘是小x
15
16 # 第4步, grid 放置方法
17 for i in range(3):
18     for j in range(3):
19         tk.Label(window, text=1).grid(row=i, column=j, padx=10, pady=10, ipadx=10, ipady=10)
20
21 # 第5步, 主窗口循环显示
22 window.mainloop()
```

测试效果:



2. Pack: The Pack Geometry Manager

我们常用的pack(), 他会按照上下左右的方式排列.例如:

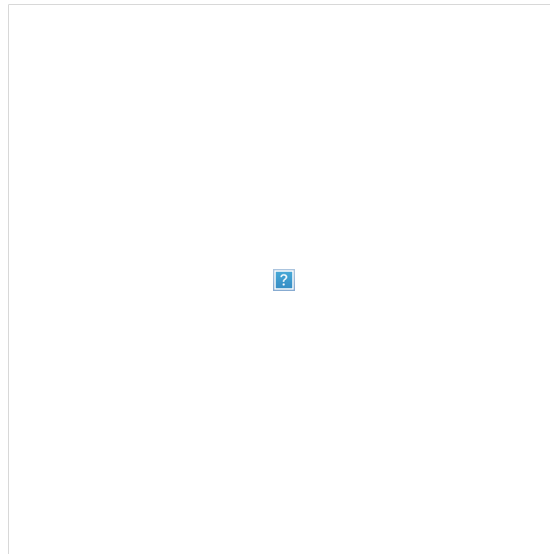
```
1  tk.Label(window, text='P', fg='red').pack(side='top')    # 上
2  tk.Label(window, text='P', fg='red').pack(side='bottom') # 下
3  tk.Label(window, text='P', fg='red').pack(side='left')   # 左
4  tk.Label(window, text='P', fg='red').pack(side='right')  # 右
```

示例代码:

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  # author:洪卫
4
5  import tkinter as tk # 使用Tkinter前需要先导入
```

```
6
7 # 第1步, 实例化object, 建立窗口window
8 window = tk.Tk()
9
10 # 第2步, 给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第3步, 设定窗口的大小(长 * 宽)
14 window.geometry('500x300') # 这里的乘是小x
15
16 # 第4步, pack 放置方法
17 tk.Label(window, text='P', fg='red').pack(side='top') # 上
18 tk.Label(window, text='P', fg='red').pack(side='bottom') # 下
19 tk.Label(window, text='P', fg='red').pack(side='left') # 左
20 tk.Label(window, text='P', fg='red').pack(side='right') # 右
21
22 # 第5步, 主窗口循环显示
23 window.mainloop()
```

测试效果:



3. Place: The Place Geometry Manager

再接下来我们来看place(), 这个比较容易理解, 就是给精确的坐标来定位, 如此处给的(50, 100), 就是将这个部件放在坐标为(x=50, y=100)的这个位置, 后面的参数 anchor='nw', 就是前面所讲的锚定点是西北角。例如:

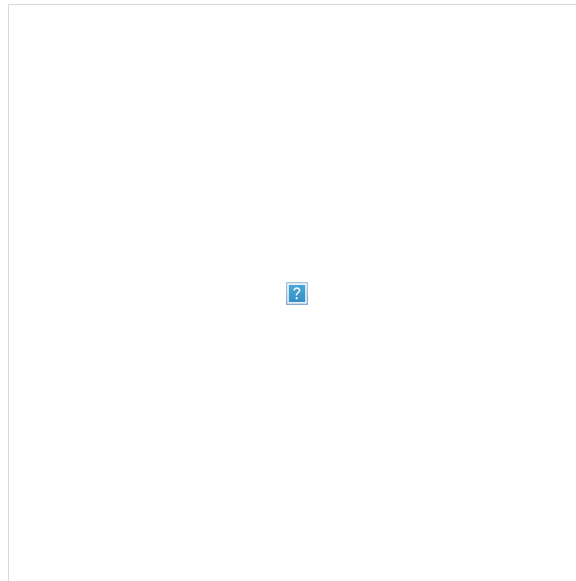
```
1 tk.Label(window, text='Pl', font=('Arial', 20), ).place(x=50, y=100, anchor='nw')
```

示例代码:

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 # author:洪卫
4
5 import tkinter as tk # 使用Tkinter前需要先导入
6
7 # 第1步, 实例化object, 建立窗口window
8 window = tk.Tk()
9
10 # 第2步, 给窗口的可视化起名字
11 window.title('My Window')
12
13 # 第3步, 设定窗口的大小(长 * 宽)
14 window.geometry('500x300') # 这里的乘是小x
15
```

```
16 # 第4步, place 放置方法 (精准的放置到指定坐标点的位置上)
17 tk.Label(window, text='Pl', font=('Arial', 20), ).place(x=50, y=100, anchor='nw')
18
19 # 第5步, 主窗口循环显示
20 window.mainloop()
```

测试效果:



14. 综合练习, 用户登录窗口例子

编写一个用户登录界面, 用户可以登录账户信息, 如果账户已经存在, 可以直接登录, 登录名或者登录密码输入错误会提示, 如果账户不存在, 提示用户注册, 点击注册进去注册页面, 输入注册信息, 确定后便可以返回登录界面进行登录。

示例代码:

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 # author:洪卫
4
5 import tkinter as tk # 使用Tkinter前需要先导入
6 import tkinter.messagebox
7 import pickle
8
9 # 第1步, 实例化object, 建立窗口window
10 window = tk.Tk()
11
12 # 第2步, 给窗口的可视化起名字
13 window.title('Wellcome to Hongwei Website')
14
15 # 第3步, 设定窗口的大小(长 * 宽)
16 window.geometry('400x300') # 这里的乘是小x
17
18 # 第4步, 加载 wellcome image
19 canvas = tk.Canvas(window, width=400, height=135, bg='green')
20 image_file = tk.PhotoImage(file='pic.gif')
21 image = canvas.create_image(200, 0, anchor='n', image=image_file)
22 canvas.pack(side='top')
23 tk.Label(window, text='Wellcome', font=('Arial', 16)).pack()
24
25 # 第5步, 用户信息
26 tk.Label(window, text='User name:', font=('Arial', 14)).place(x=10, y=170)
27 tk.Label(window, text='Password:', font=('Arial', 14)).place(x=10, y=210)
28
```

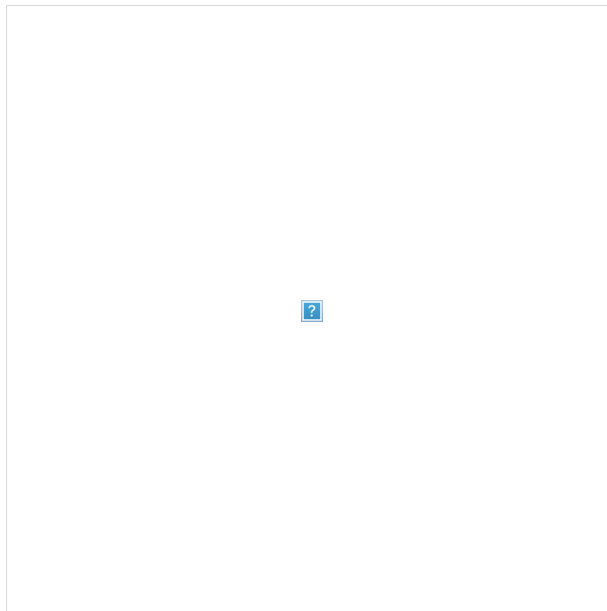
```

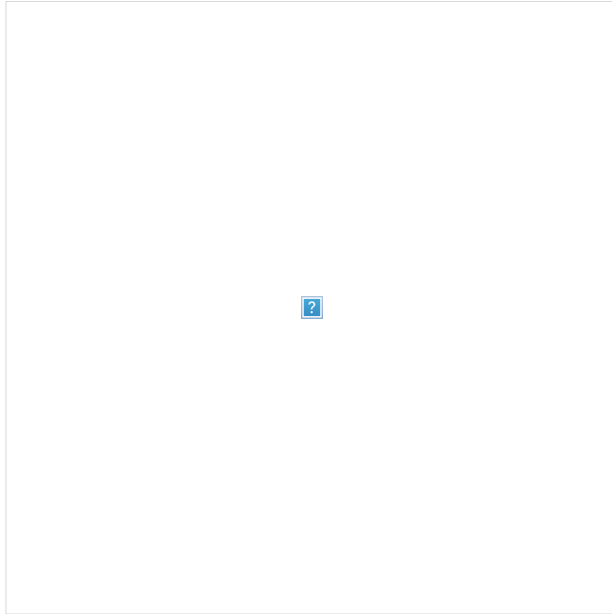
29 # 第6步, 用户登录输入框entry
30 # 用户名
31 var_usr_name = tk.StringVar()
32 var_usr_name.set('example@python.com')
33 entry_usr_name = tk.Entry(window, textvariable=var_usr_name, font=('Arial', 14))
34 entry_usr_name.place(x=120,y=175)
35 # 用户密码
36 var_usr_pwd = tk.StringVar()
37 entry_usr_pwd = tk.Entry(window, textvariable=var_usr_pwd, font=('Arial', 14), show='*')
38 entry_usr_pwd.place(x=120,y=215)
39
40 # 第8步, 定义用户登录功能
41 def usr_login():
42     # 这两行代码就是获取用户输入的usr_name和usr_pwd
43     usr_name = var_usr_name.get()
44     usr_pwd = var_usr_pwd.get()
45
46     # 这里设置异常捕获, 当我们第一次访问用户信息文件时是不存在的, 所以这里设置异常捕获。
47     # 中间的两行就是我们的匹配, 即程序将输入的信息和文件中的信息匹配。
48     try:
49         with open('usrs_info.pickle', 'rb') as usr_file:
50             usrs_info = pickle.load(usr_file)
51     except FileNotFoundError:
52         # 这里就是我们在没有读取到`usr_file`的时候, 程序会创建一个`usr_file`这个文件, 并将管理员
53         # 的用户和密码写入, 即用户名为`admin`密码为`admin`。
54         with open('usrs_info.pickle', 'wb') as usr_file:
55             usrs_info = {'admin': 'admin'}
56             pickle.dump(usrs_info, usr_file)
57             usr_file.close() # 必须先关闭, 否则pickle.load()会出现EOFError: Ran out of input
58
59     # 如果用户名和密码与文件中的匹配成功, 则会登录成功, 并跳出弹窗how are you? 加上你的用户名。
60     if usr_name in usrs_info:
61         if usr_pwd == usrs_info[usr_name]:
62             tkinter.messagebox.showinfo(title='Welcome', message='How are you? ' + usr_name)
63             # 如果用户名匹配成功, 而密码输入错误, 则会弹出'Error, your password is wrong, try again.'
64         else:
65             tkinter.messagebox.showerror(message='Error, your password is wrong, try again.')
66     else: # 如果发现用户名不存在
67         is_sign_up = tkinter.messagebox.askyesno('Welcome! ', 'You have not sign up yet. S
68         # 提示需不需要注册新用户
69         if is_sign_up:
70             usr_sign_up()
71
72 # 第9步, 定义用户注册功能
73 def usr_sign_up():
74     def sign_to_Hongwei_Website():
75         # 以下三行就是获取我们注册时所输入的信息
76         np = new_pwd.get()
77         npf = new_pwd_confirm.get()
78         nn = new_name.get()
79
80         # 这里是打开我们记录数据的文件, 将注册信息读出
81         with open('usrs_info.pickle', 'rb') as usr_file:
82             exist_usr_info = pickle.load(usr_file)
83         # 这里就是判断, 如果两次密码输入不一致, 则提示Error, Password and confirm password must be the same
84         if np != npf:
85             tkinter.messagebox.showerror('Error', 'Password and confirm password must be the same')
86
87         # 如果用户名已经在我们的数据文件中, 则提示Error, The user has already signed up!
88         elif nn in exist_usr_info:
89             tkinter.messagebox.showerror('Error', 'The user has already signed up!')
90
91         # 最后如果输入无以上错误, 则将注册输入的信息记录到文件当中, 并提示注册成功Welcome! ,You have successfully signed up!
92         else:
93             exist_usr_info[nn] = np

```

```
94         with open('usrs_info.pickle', 'wb') as usr_file:
95             pickle.dump(exist_usr_info, usr_file)
96             tkinter.messagebox.showinfo('Welcome', 'You have successfully signed up!')
97             # 然后销毁窗口。
98             window_sign_up.destroy()
99
100         # 定义长在窗口上的窗口
101         window_sign_up = tk.Toplevel(window)
102         window_sign_up.geometry('300x200')
103         window_sign_up.title('Sign up window')
104
105         new_name = tk.StringVar() # 将输入的注册名赋值给变量
106         new_name.set('example@python.com') # 将最初显示定为'example@python.com'
107         tk.Label(window_sign_up, text='User name: ').place(x=10, y=10) # 将`User name:`放置在坐标(10,10)
108         entry_new_name = tk.Entry(window_sign_up, textvariable=new_name) # 创建一个注册名的`entry`
109         entry_new_name.place(x=130, y=10) # `entry`放置在坐标 (150,10) .
110
111         new_pwd = tk.StringVar()
112         tk.Label(window_sign_up, text='Password: ').place(x=10, y=50)
113         entry_usr_pwd = tk.Entry(window_sign_up, textvariable=new_pwd, show='*')
114         entry_usr_pwd.place(x=130, y=50)
115
116         new_pwd_confirm = tk.StringVar()
117         tk.Label(window_sign_up, text='Confirm password: ').place(x=10, y=90)
118         entry_usr_pwd_confirm = tk.Entry(window_sign_up, textvariable=new_pwd_confirm, show='*')
119         entry_usr_pwd_confirm.place(x=130, y=90)
120
121         # 下面的 sign_to_Hongwei_Website
122         btn_confirm_sign_up = tk.Button(window_sign_up, text='Sign up', command=sign_to_Hongwei_Website)
123         btn_confirm_sign_up.place(x=180, y=120)
124
125         # 第7步, login and sign up 按钮
126         btn_login = tk.Button(window, text='Login', command=usr_login)
127         btn_login.place(x=120, y=240)
128         btn_sign_up = tk.Button(window, text='Sign up', command=usr_sign_up)
129         btn_sign_up.place(x=200, y=240)
130
131         # 第10步, 主窗口循环显示
132         window.mainloop()
```

测试效果:





您, 其他部分可能还需要补充...

注：不同电脑可能配置环境略有不同，如有小错误可以自己调试一下。

作者：洪卫

出处：<http://www.cnblogs.com/shwee/>

个性签名：独学而无友，则孤陋而寡闻。做一个灵魂有趣的人！

如果觉得这篇文章对你有小小的帮助的话，记得在右下角点个 **[推荐]** 哦，博主在此感谢！

万水千山总是情，打赏一分行不行，所以如果你心情还比较高兴，也是可以扫码打赏博主，哈哈(づ•ω•)づ~)！



分类：04. Python基础

标签：Python, tkinter, tk, GUI, 用户界面, 登录界面

好文要顶

关注我

收藏该文



洪卫

关注 - 11

粉丝 - 169

+ 加关注

« 上一篇：自控力极差的人如何自救？

» 下一篇：戴尔·卡耐基《人性的弱点》阅读笔记（1）

posted on 2018-08-09 08:04 洪卫 阅读(18906) 评论(28) 编辑 收藏

评论

#1楼

这个东西和winform真的不是一个量级的东西,作出来的UI像java的awt

支持(0) 反对(0)

2018-08-09 14:10 | 码农之一

#2楼

怎么看起来像winform？

支持(1) 反对(0)

2018-08-09 14:16 | CalvinR

#3楼

用python写UI，难受吗，哈哈

支持(0) 反对(0)

2018-08-09 14:36 | 陌上不开花

#4楼[楼主]

@ 韩之一

嗯嗯，本来就是很简单很轻量的一个模块吧，简单学一下，平时做点小插件，小应用，够用就行。大型业务软件一般不会用这个开发。

支持(1) 反对(0)

2018-08-09 14:37 | 洪卫

#5楼

@ 陌上不开花

我觉得写一些小工具挺方便的啊。

支持(0) 反对(0)

2018-08-09 14:37 | IclodQ

#6楼[楼主]

@ CalvinR

哈哈，不过不是winform。比较简单和轻量的模块。

支持(0) 反对(0)

2018-08-09 14:38 | 洪卫

#7楼[楼主]

@ IclodQ

对的，就是用来写一些小工具，小插件，小界面还是比较方便的(◡‿◡)✧

支持(0) 反对(0)

2018-08-09 14:40 | 洪卫

#8楼[楼主]

@ 陌上不开花

谈不上难受吧，挺简单的，还是挺有意思的，本身就是一个比较轻量化的模块都嘛(◡‿◡)ʕ，哈哈

支持(0) 反对(0)

2018-08-09 14:43 | 洪卫

#9楼

@ 洪卫

我只用C#做Winform，

支持(0) 反对(0)

2018-08-09 16:22 | zhang_jun_hong

#10楼

powershell图形界面组件:

字符界面的进度条:

Install-Module psInlineProgress

图片:

<https://communnary.files.wordpress.com/2016/04/inlineprogressbartests.png>

powershell脚本, 图形界面的进度条。

<https://gallery.technet.microsoft.com/Progress-Bar-With-d3924344>

图示:

https://gallery.technet.microsoft.com/site/view/file/143776/1/PWShell_PB.jpg

wpf界面的进度条:

<https://gallery.technet.microsoft.com/scriptcenter/PoshProgressBar-a72dc1d8>

win下ps可调用的, 图形组件, 输入框, 多选框, 等。

<https://github.com/My-Random-Thoughts/Various-Code/blob/master/Show-InputForm.ps1>

=====

藏脚阁今日分享:

1脚本中的ps代码, 用多线程开一个【win消息输出窗口】message-box。

2从脚本中, 发送消息给窗口。实现多线程消息传送。

即脚本中的代码块自己运行, 随时输出消息给窗口, 让窗口显示。

<https://github.com/oze4/ThreadedMessageBox>

3多线程的好处就是传值容易, 坏处就是编写难。

不过有了别人写好的模块, 你用起来就简单了。

支持(0) 反对(0)

2018-08-09 16:25 | PowerShell免费软件

#11楼[楼主]

@ zhang_jun_hong

嗯嗯, ·Net开发平台, C#还是很强大的。(๑•̀•́)。

这篇给的这个还是很轻量的模块, 做做小工具还是可以的, 大型的业务交互界面一般也不会用这个。

支持(0) 反对(0)

2018-08-09 16:36 | 洪卫

#12楼

我曾经用它写过文本编辑器.....

支持(0) 反对(0)

2018-08-09 16:50 | 从零开始的程序员生活

#13楼[楼主]

@ 从零开始的程序员生活

嗯嗯, good job! O(∩_∩)O哈哈~

	支持(0) 反对(0)
2018-08-09 16:51 洪卫	
<div>#14楼</div> <div>这python的UI看上去很难受呀，太low了点</div> <div>支持(0) 反对(0)</div>	
2018-08-09 17:53 markwu	
<div>#15楼[楼主]</div> <div>@ markwu 这看需求吧，对这方面有一定基础的来说，当然比较简单喽，哈哈。</div> <div>支持(0) 反对(0)</div>	
2018-08-09 17:57 洪卫	
<div>#16楼</div> <div>python 大佬</div> <div>支持(0) 反对(0)</div>	
2018-08-09 22:28 ~雨落忧伤~	
<div>#17楼</div> <div>Tkinter很糟糕，PyQt写界面，那才叫舒服</div> <div>支持(1) 反对(0)</div>	
2018-08-09 22:40 nutix	
<div>#18楼[楼主]</div> <div>@ ~雨落忧伤~ 大佬不敢当，只是学习学习，哈哈O(∩_∩)O~</div> <div>支持(0) 反对(0)</div>	
2018-08-09 23:16 洪卫	
<div>#19楼[楼主]</div> <div>@ nutix 嗯嗯，还行吧，后面应该会写一篇PyQt的学习笔记</div> <div>支持(0) 反对(0)</div>	
2018-08-09 23:29 洪卫	
<div>#20楼</div> <div>浏览这么多关于 tkinter 的学习笔记，楼主的这篇是最全，最详细，最好的</div> <div>支持(0) 反对(0)</div>	
2018-08-29 14:24 柳色青	
<div>#21楼[楼主]</div> <div>@ 柳色青 谢谢支持！一起交流O(∩_∩)O</div> <div>支持(0) 反对(0)</div>	
2018-08-29 14:30 洪卫	
<div>#22楼</div>	

楼主，这个教程跟莫凡python那个教程几乎一样，如果是转载的话，尽量表明来源吧。

支持(0) 反对(0)

2018-11-08 18:07 | kazuma

#23楼[楼主]

@ kazuma

嗯嗯，不是转载，学了教程自己敲的，应该可以加一个参考来源，谢谢你的建议。ㄟㄏㄩ

支持(0) 反对(0)

2018-11-17 11:07 | 洪卫

#24楼

感谢分享，还没看，先回复感谢楼主分享，嘿嘿~~

支持(0) 反对(0)

2019-01-08 15:26 | 5055555

#25楼

建议楼主介绍每个部件的时候，相对应的构造方法最好也讲下，这样子会更容易理解些。

支持(0) 反对(0)

2019-01-08 17:58 | 5055555

#26楼[楼主]

@ 5055555

嗯嗯，好的，谢谢你的建议！

支持(0) 反对(0)

2019-01-21 17:03 | 洪卫

#27楼

请问楼主我想在界面上加一个小控制灯，应该用什么控件呢？

支持(0) 反对(0)

2019-02-04 12:59 | leifeng1

#28楼

这有个python tkinter做的豆瓣电影助手项目，我看了下，几乎把大部分tkinter的控件给使用了，值得大家学习。

github地址是<https://github.com/shengqiangzhang/doubanMovieTool>

支持(0) 反对(0)

2019-02-22 22:52 | 云外孤鸟

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。