一、前言

由于本篇文章较长，所以下面给出内容目录方便阅读

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 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 模块元素简要说明

tkinter 类	元素	简要说明
Button	按钮	类似标签,但提供额外的功能，点击时执行一个动作,例如鼠标掠过、按下、释放以及键盘操作/事件
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
Checkbutton	复选框	允许用户选择或反选一个选项，一组方框,可以选择其中的任意个(类似 HTML 中的 checkbox)
Entry	单行文本框	单行文字域，显示一行文本，用来收集键盘输入(类似 HTML 中的 text)
Frame	框架	用来承载放置其他GUI元素，就是一个容器
Label	标签	用于显示不可编辑的文本或图标
LabelFrame	容器控件	是一个简单的容器控件。常用与复杂的窗口布局
Listbox	列表框	一个选项列表,用户可以从中选择
Menu	菜单	点下菜单按钮后弹出的一个选项列表,用户可以从中选择
Menubutton	菜单按钮	用来包含菜单的组件(有下拉式、层叠式等等)
Message	消息框	类似于标签,但可以显示多行文本
OptionMenu	选择菜单	下拉菜单的一个改版，弥补了Listbox无法下拉列表框的遗憾
PanedWindow	窗口布局管理	是一个窗口布局管理的插件，可以包含一个或者多个子控件。
Radiobuttion	单选框	允许用户从多个选项中选取一个，一组按钮,其中只有一个可被“按下” (类似 HTML 中的 radio)
Scale	进度条	线性“滑块”组件,可设定起始值和结束值,会显示当前位置的精确值
Scrollbar	滚动条	对其支持的组件(文本域、画布、列表框、文本框)提供滚动功能
Spinbox	输入控件	与Entry类似，但是可以指定输入范围值
Text	多行文本框	多行文字区域，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)
Toplevel	顶层	类似框架,为其他的控件提供单独的容器
messageBox	消息框	用于显示你应用程序的消息框。(Python2中为tkMessagebox)

The Button Widget
The Canvas Widget
The Checkbutton Widget
The Entry Widget
The Frame Widget
The Label Widget
The LabelFrame Widget
The Listbox Widget
The Menu Widget
The Menubutton Widget
The Message Widget
The OptionMenu Widget
The PanedWindow Widget
The Radiobutton Widget
The Scale Widget
The Scrollbar Widget
The Spinbox Widget
The Text Widget
The Toplevel Widget
Basic Widget Methods
Toplevel Window Methods

2. 常用窗口部件及简要说明：

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，并且清除与这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：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')
```

```
# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 第 4 步，在图形界面上设定标签
l = tk.Label(window, text='你好！this is Tkinter', bg='green',
font=('Arial', 12), width=30, height=2)
# 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 2 个字符这么高

# 第 5 步，放置标签
l.pack()    # Label 内容 content 区域放置位置，自动调节尺寸
# 放置 lable 的方法有：1）l.pack(); 2)l.place();

# 第 6 步，主窗口循环显示
window.mainloop()
# 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 window,传入进去的值就不会有循环，
mainloop 就相当于一个很大的 while 循环，有个 while，每点击一次就会更新一
次，所以我们必须要有循环
# 所有的窗口文件都必须有类似的 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：

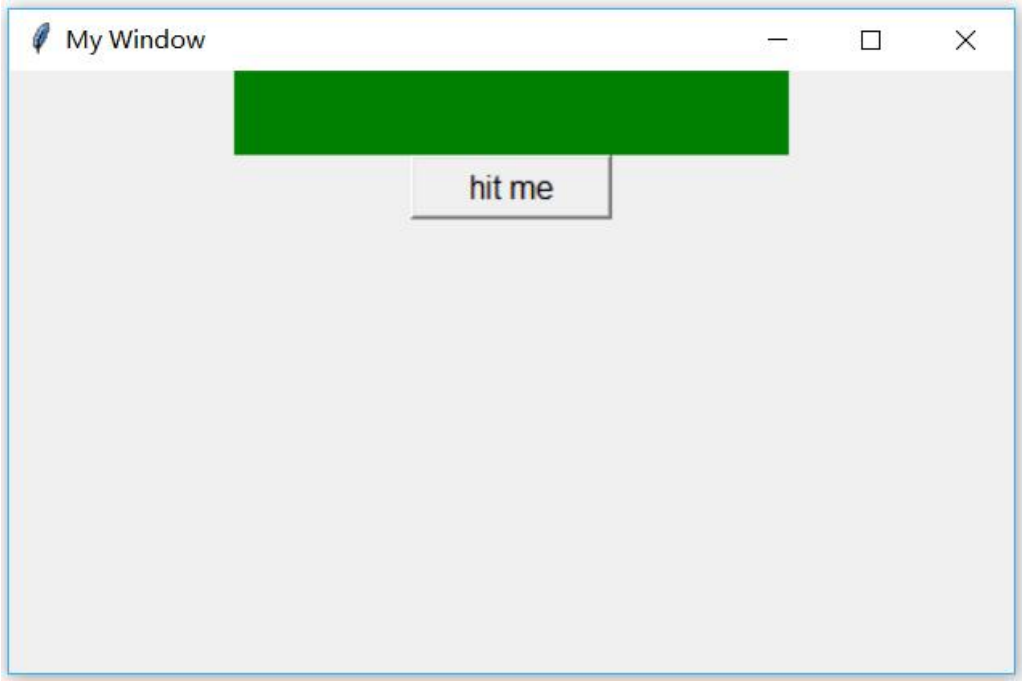
```
b = tk.Button(window, text="Help", command=DISABLED)
```

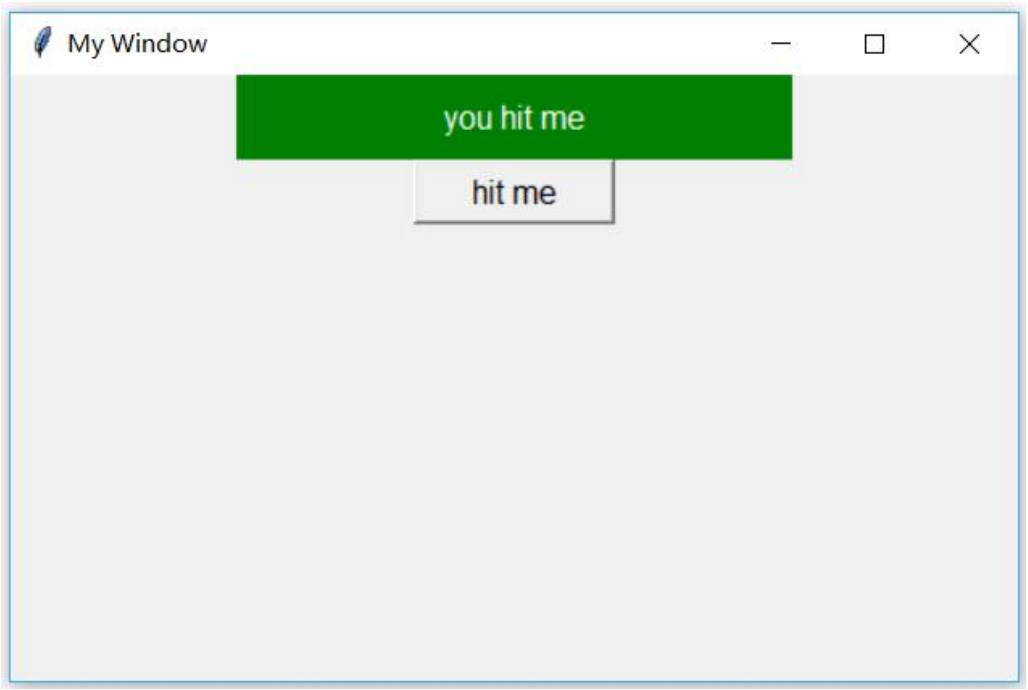
示例代码：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 第 4 步，在图形界面上设定标签
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签上
```

```
l = tk.Label(window, textvariable=var, bg='green', fg='white',
font=('Arial', 12), width=30, height=2)
# 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=2,就是标签有 2 个字符这么高
l.pack()

# 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
on_hit = False
def hit_me():
    global on_hit
    if on_hit == False:
        on_hit = True
        var.set('you hit me')
    else:
        on_hit = False
        var.set('')

# 第 5 步，在窗口界面设置放置 Button 按键
b = tk.Button(window, text='hit me', font=('Arial', 12), width=10,
height=1, command=hit_me)
b.pack()

# 第 6 步，主窗口循环显示
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：

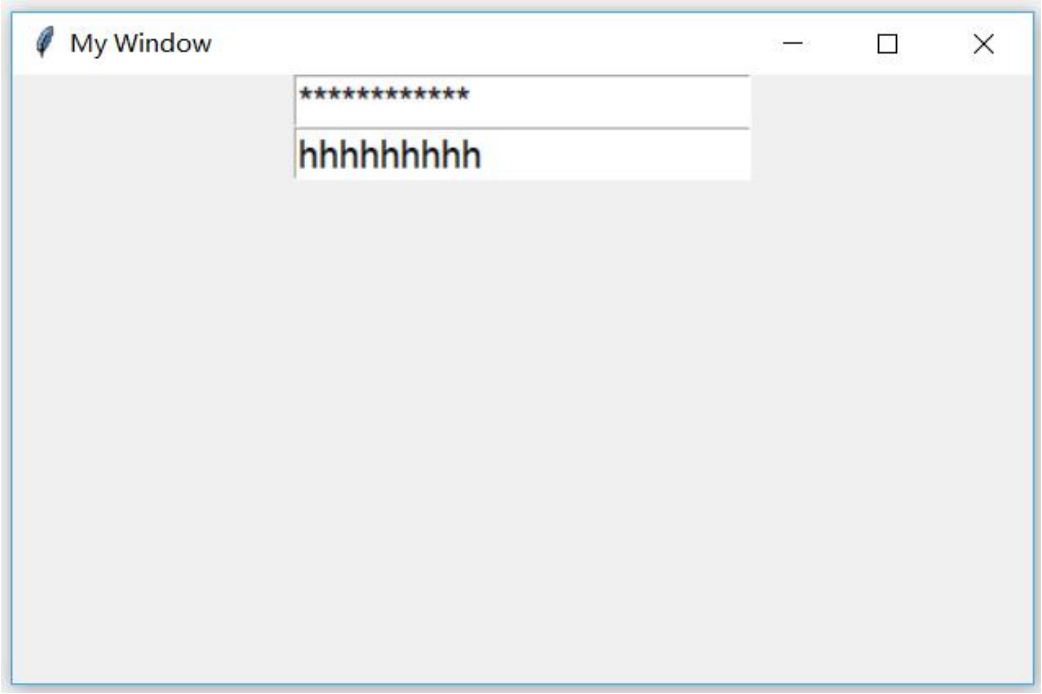
需要用户输入用户信息时，比如我们平时使用软件、登录网页时，用户交互界面让我们登录账户信息等时候可以用到。

示例代码：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x
```

```
# 第4步，在图形界面上设定输入框控件entry并放置控件
e1 = tk.Entry(window, show='*', font=('Arial', 14))    # 显示成密文形式
e2 = tk.Entry(window, show=None, font=('Arial', 14))   # 显示成明文形式
e1.pack()
e2.pack()

# 第5步，主窗口循环显示
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。

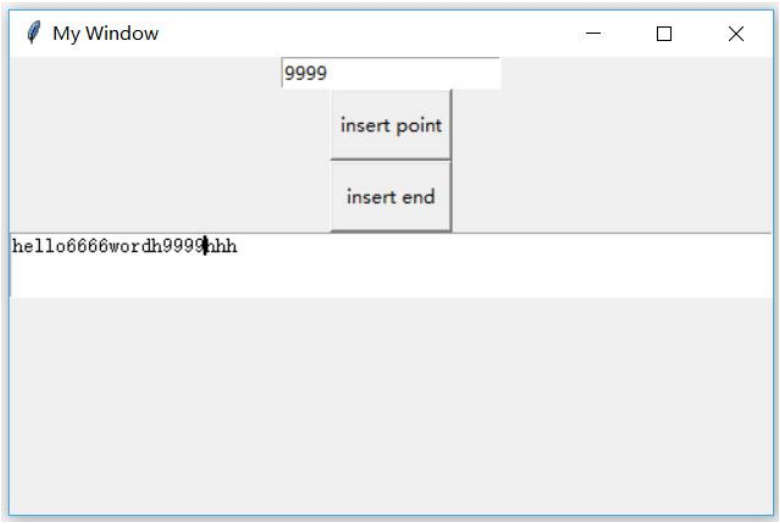
示例代码：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫
```

```
import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 第 4 步，在图形界面上设定输入框控件 entry 框并放置
e = tk.Entry(window, show = None)#显示成明文形式
e.pack()

# 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，所以函数一定要放在按钮的上面）
def insert_point(): # 在鼠标焦点处插入输入内容
    var = e.get()
    t.insert('insert', var)
def insert_end():    # 在文本框内容最后接着插入输入内容
    var = e.get()
    t.insert('end', var)

# 第 6 步，创建并放置两个按钮分别触发两种情况
b1 = tk.Button(window, text='insert point', width=10,
               height=2, command=insert_point)
b1.pack()
b2 = tk.Button(window, text='insert end', width=10,
               height=2, command=insert_end)
b2.pack()

# 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
t = tk.Text(window, height=3)
t.pack()

# 第 8 步，主窗口循环显示
window.mainloop()
```

.

测试效果：



5. Listbox 窗口部件

简单说明：

Text 是 tkinter 类中提供的的列表框部件，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：

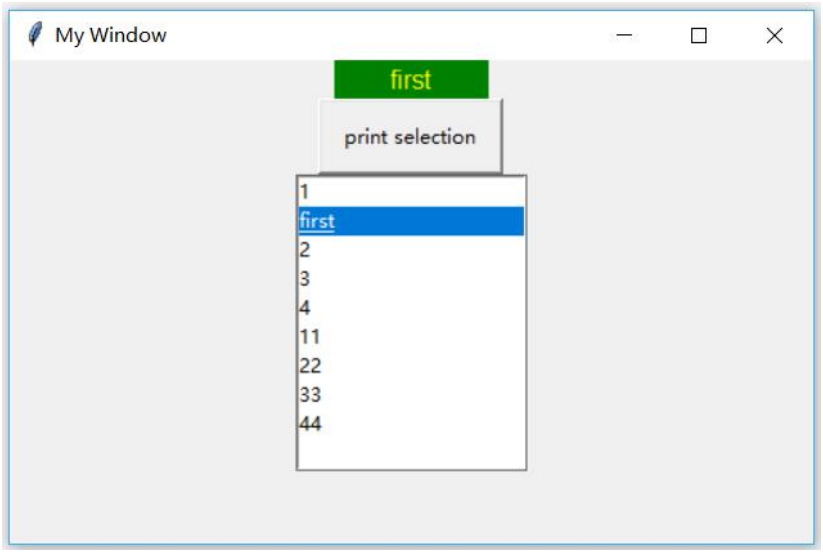
在有一个很多内容选项组成的列表提供用户选择时会用到。

示例代码：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 第 4 步，在图形界面上创建一个标签 label 用以显示并放置
var1 = tk.StringVar()  # 创建变量，用 var1 用来接收鼠标点击具体选项的内容
l = tk.Label(window, bg='green', fg='yellow',font=('Arial', 12),
width=10, textvariable=var1)
```

```
l.pack()

# 第 6 步，创建一个方法用于按钮的点击事件
def print_selection():
    value = lb.get(lb.curselection())    # 获取当前选中的文本
    var1.set(value)  # 为 label 设置值

# 第 5 步，创建一个按钮并放置，点击按钮调用 print_selection 函数
b1 = tk.Button(window, text='print selection', width=15, height=2,
command=print_selection)
b1.pack()

# 第 7 步，创建 Listbox 并为其添加内容
var2 = tk.StringVar()
var2.set((1,2,3,4)) # 为变量 var2 设置值
# 创建 Listbox
lb = tk.Listbox(window, listvariable=var2)  #将 var2 的值赋给 Listbox
# 创建一个 list 并将值循环添加到 Listbox 控件中
list_items = [11,22,33,44]
for item in list_items:
    lb.insert('end', item)  # 从最后一个位置开始加入值
lb.insert(1, 'first')       # 在第一个位置加入'first'字符
lb.insert(2, 'second')      # 在第二个位置加入'second'字符
lb.delete(2)                # 删除第二个位置的字符
lb.pack()

# 第 8 步，主窗口循环显示
window.mainloop()
```

- 测试效果：



6. 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：

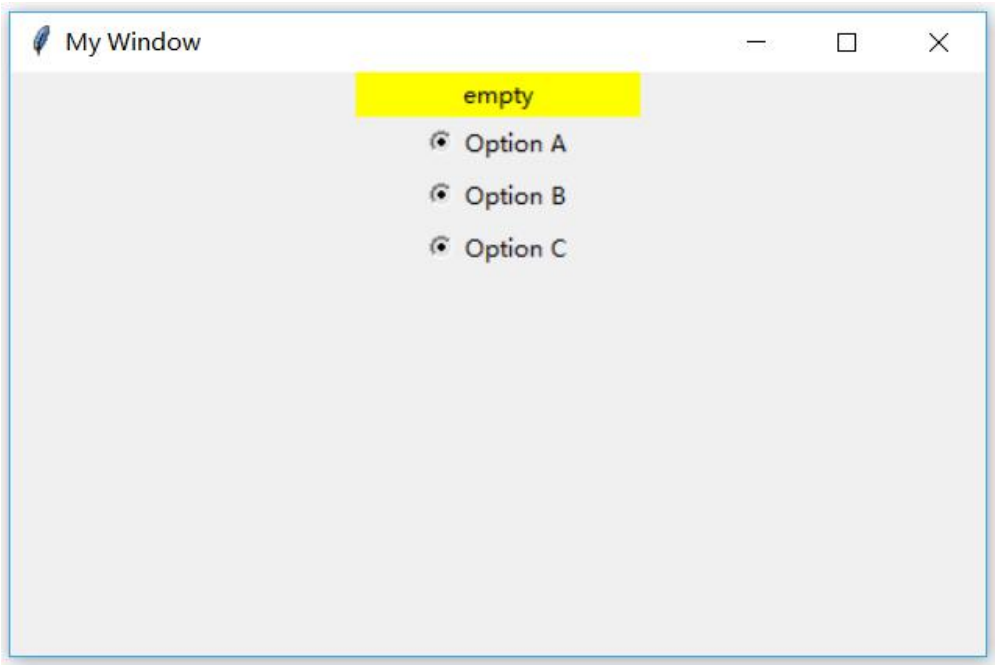
在有一个很多内容选项组成的选项列表提供用户选择时会用到，用户一次只能选择其中一个，不能多选。

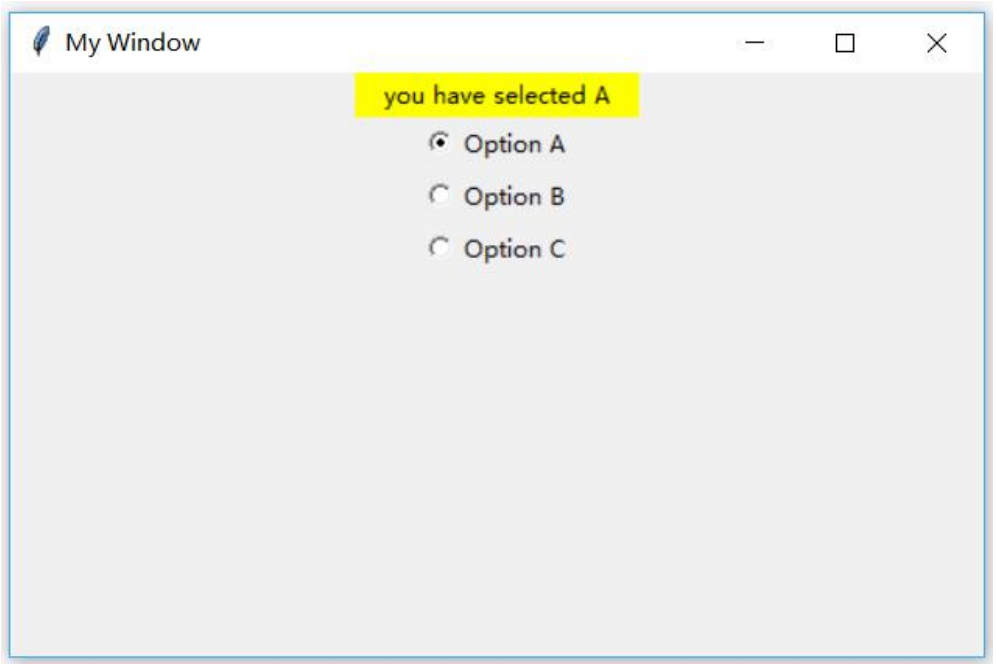
示例代码：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 第 4 步，在图形界面上创建一个标签 label 用以显示并放置
var = tk.StringVar()    # 定义一个 var 用来将 radiobutton 的值和 Label 的值联系在一起.
l = tk.Label(window, bg='yellow', width=20, text='empty')
l.pack()

# 第 6 步，定义选项触发函数功能
def print_selection():
    l.config(text='you have selected ' + var.get())
# 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，把 value 的值 A 放到变量 var 中，然后赋值给 variable
r1 = tk.Radiobutton(window, text='Option A', variable=var, value='A', command=print_selection)
r1.pack()
r2 = tk.Radiobutton(window, text='Option B', variable=var, value='B', command=print_selection)
r2.pack()
```

```
r3 = tk.Radiobutton(window, text='Option C', variable=var, value='C'
, command=print_selection)
r3.pack()

# 第 7 步，主窗口循环显示
window.mainloop()
```

测试效果：





7. Checkbutton 窗口部件

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：

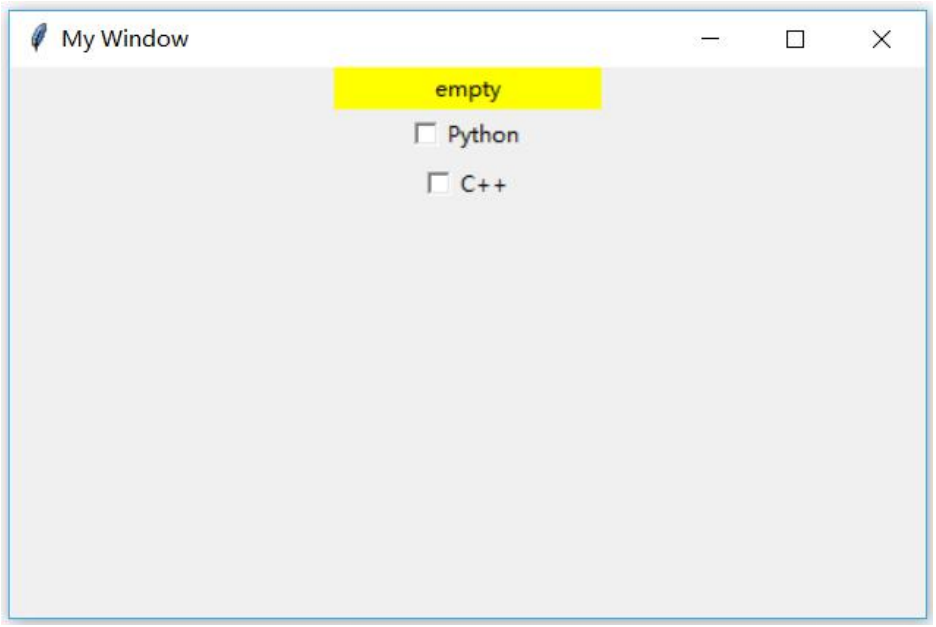
在有一个很多内容选项组成的选项列表提供用户选择时会用到，用户一次可以选择多个。

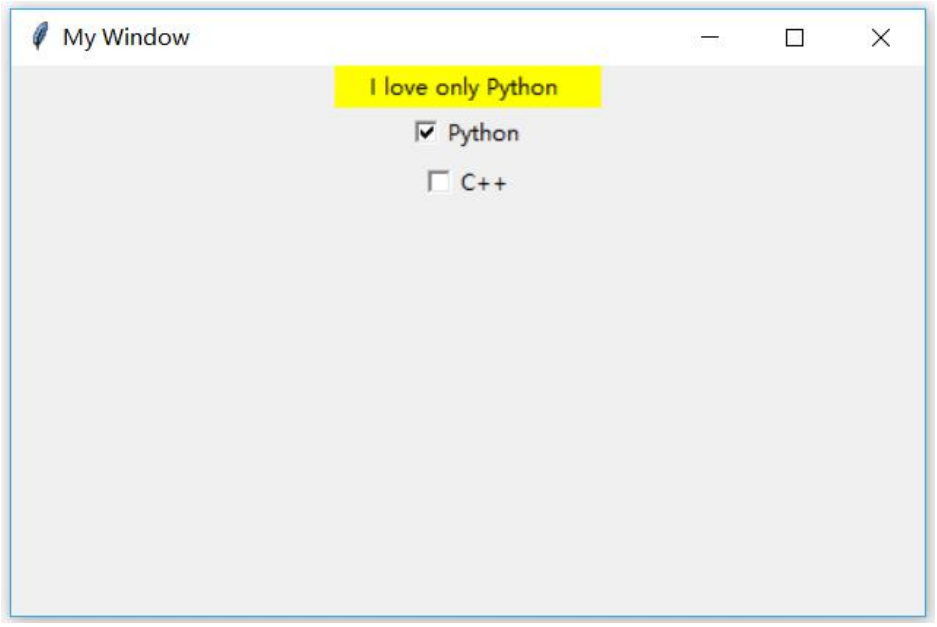
示例代码：

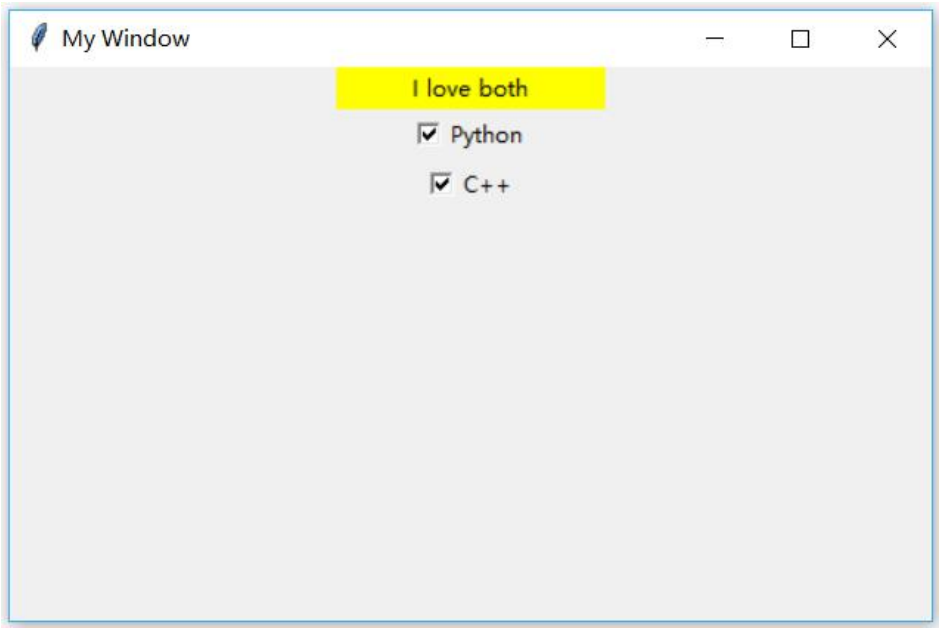
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 第 4 步，在图形界面上创建一个标签 label 用以显示并放置
l = tk.Label(window, bg='yellow', width=20, text='empty')
l.pack()

# 第 6 步，定义触发函数功能
def print_selection():
    if (var1.get() == 1) & (var2.get() == 0):      # 如果选中第一个选项，未选中第二个选项
        l.config(text='I love only Python ')
    elif (var1.get() == 0) & (var2.get() == 1):    # 如果选中第二个选项，未选中第一个选项
        l.config(text='I love only C++')
    elif (var1.get() == 0) & (var2.get() == 0):    # 如果两个选项都未选中
        l.config(text='I do not love either')
    else:
        l.config(text='I love both')               # 如果两个选项都选中

# 第 5 步，定义两个 Checkbutton 选项并放置
var1 = tk.IntVar()  # 定义 var1 和 var2 整型变量用来存放选择行为返回值
var2 = tk.IntVar()
```

```
c1 = tk.Checkbutton(window, text='Python',variable=var1, onvalue=1,
offvalue=0, command=print_selection)    # 传值原理类似于 radiobutton
部件
c1.pack()
c2 = tk.Checkbutton(window, text='C++',variable=var2, onvalue=1,
offvalue=0, command=print_selection)
c2.pack()

# 第 7 步，主窗口循环显示
window.mainloop()
```

- 测试效果：







8. 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字值。

什么时候用：

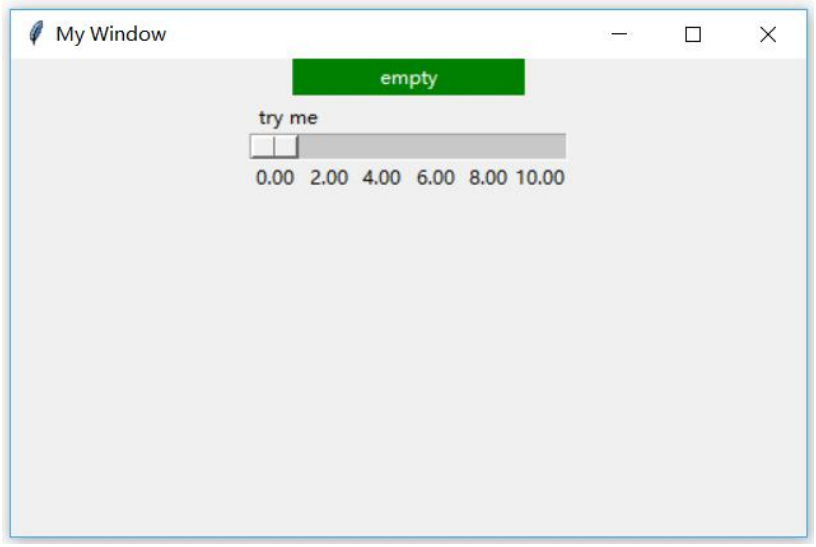
在需要用户给出评价等级，或者给出一个评价分数，或者拉动滑动条提供一个具体的数值等等。

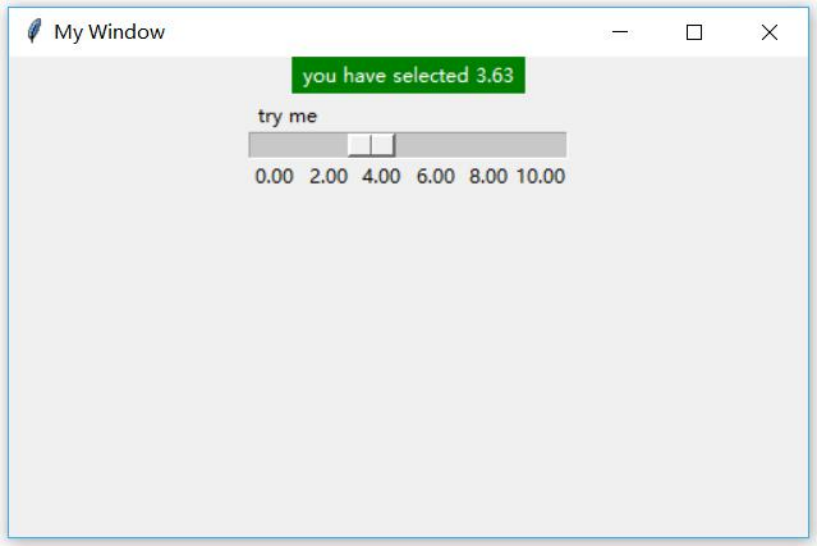
示例代码：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 第 4 步，在图形界面上创建一个标签 label 用以显示并放置
l = tk.Label(window, bg='green', fg='white', width=20, text='empty')
```

```
l.pack()

# 第6步，定义一个触发函数功能
def print_selection(v):
    l.config(text='you have selected ' + v)
# 第5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
s = tk.Scale(window, label='try me', from_=0, to=10,
orient=tk.HORIZONTAL, length=200, showvalue=0,tickinterval=2,
resolution=0.01, command=print_selection)
s.pack()

# 第7步，主窗口循环显示
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：

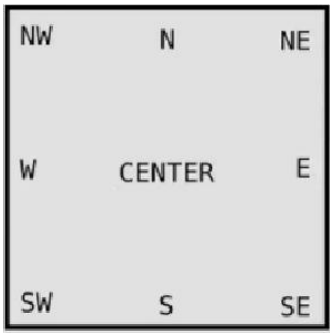
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 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
canvas = tk.Canvas(window, bg='green', height=200, width=500)
# 说明图片位置，并导入图片到画布上
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，也可以用绝对路径，需要给定图片具体绝对路径）
image = canvas.create_image(250, 0, anchor='n',image=image_file)
# 图片锚定点（n 图片顶端的中间点位置）放在画布（250,0）坐标处
# 定义多边形参数，然后在画布上画出指定图形
x0, y0, x1, y1 = 100, 100, 150, 150
line = canvas.create_line(x0-50, y0-50, x1-50, y1-50)
# 画直线
oval = canvas.create_oval(x0+120, y0+50, x1+120, y1+50, fill='yellow')
# 画圆 用黄色填充
arc = canvas.create_arc(x0, y0+50, x1, y1+50, start=0, extent=180)
# 画扇形 从 0 度打开收到 180 度结束
rect = canvas.create_rectangle(330, 30, 330+20, 30+20)
# 画矩形正方形
canvas.pack()
```

```
# 第6步，触发函数，用来一定指定图形
def moveit():
    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=2, y=2）步长进行移动

# 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
b = tk.Button(window, text='move item',command=moveit).pack()

# 第7步，主窗口循环显示
window.mainloop()
```

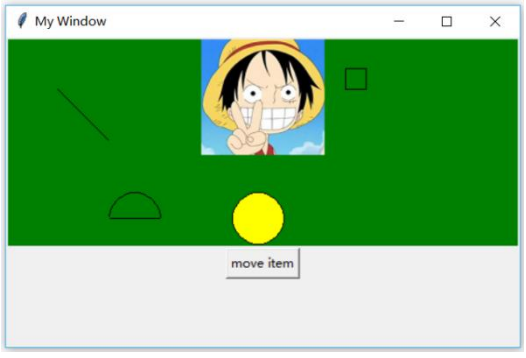
所用图片：

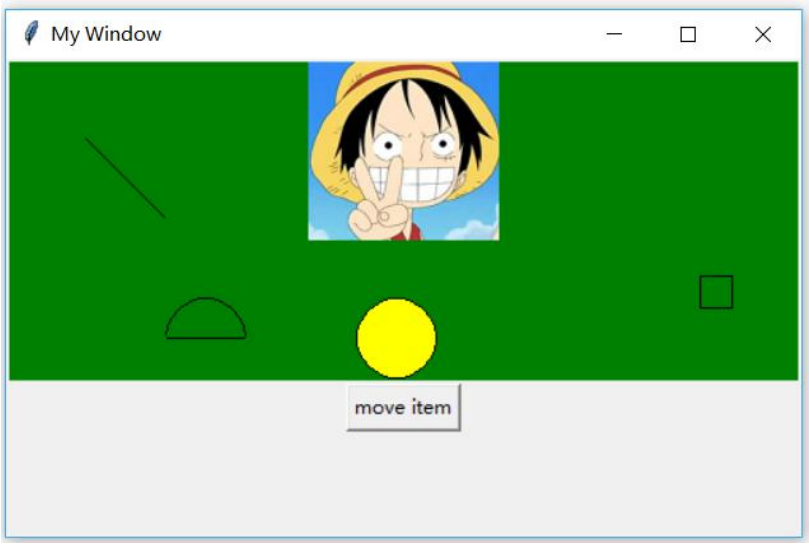
当然你可以随意用你的一张图片导入画布试一试效果，图片可以用画图工具改一下像素大小，以免图片太大，导入画布显示不全，当然你也可以用我提供的素材，下面是链接：https://files.cnblogs.com/files/shwee/pic.gif

图片锚定点位置参数图：



测试效果：





10. 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：

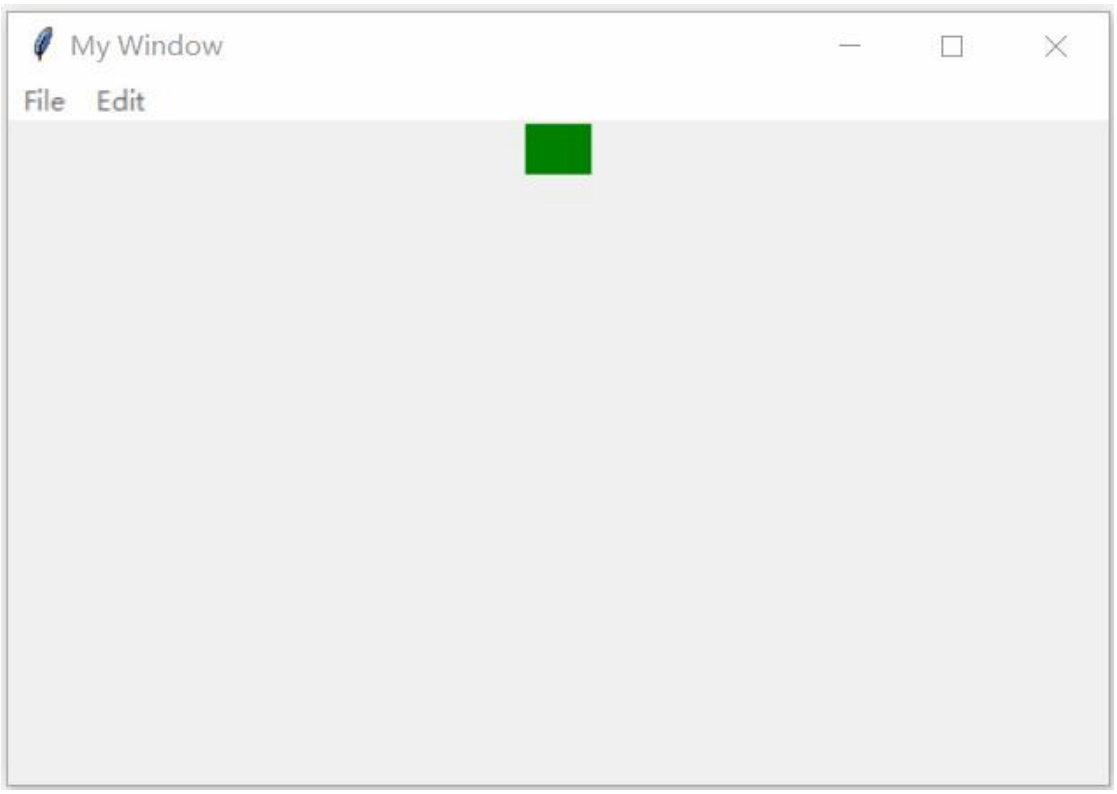
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 第 4 步，在图形界面上创建一个标签用以显示内容并放置
l = tk.Label(window, text='  ', bg='green')
l.pack()
```

```
# 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
counter = 0
def do_job():
    global counter
    l.config(text='do '+ str(counter))
    counter += 1

# 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
menubar = tk.Menu(window)

# 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）
filemenu = tk.Menu(menubar, tearoff=0)
# 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
menubar.add_cascade(label='File', menu=filemenu)

# 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。
filemenu.add_command(label='New', command=do_job)
filemenu.add_command(label='Open', command=do_job)
filemenu.add_command(label='Save', command=do_job)
filemenu.add_separator()     # 添加一条分隔线
filemenu.add_command(label='Exit', command=window.quit) # 用 tkinter 里面自带的 quit()函数

# 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）
editmenu = tk.Menu(menubar, tearoff=0)
# 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
menubar.add_cascade(label='Edit', menu=editmenu)

# 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
editmenu.add_command(label='Cut', command=do_job)
editmenu.add_command(label='Copy', command=do_job)
editmenu.add_command(label='Paste', command=do_job)

# 第 8 步，创建第二级菜单，即菜单项里面的菜单
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
filemenu.add_cascade(label='Import', menu=submenu, underline=0) # 给放入的菜单 submenu 命名为 Import
```

```
# 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~~~）
submenu.add_command(label='Submenu_1', command=do_job)    # 这里和上面创建原理也一样，在 Import 菜单项中加入一个小菜单命令 Submenu_1

# 第 11 步，创建菜单栏完成后，配置让菜单栏 menubar 显示出来
window.config(menu=menubar)

# 第 12 步，主窗口循环显示
window.mainloop()
```

-

测试效果：



11. 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：

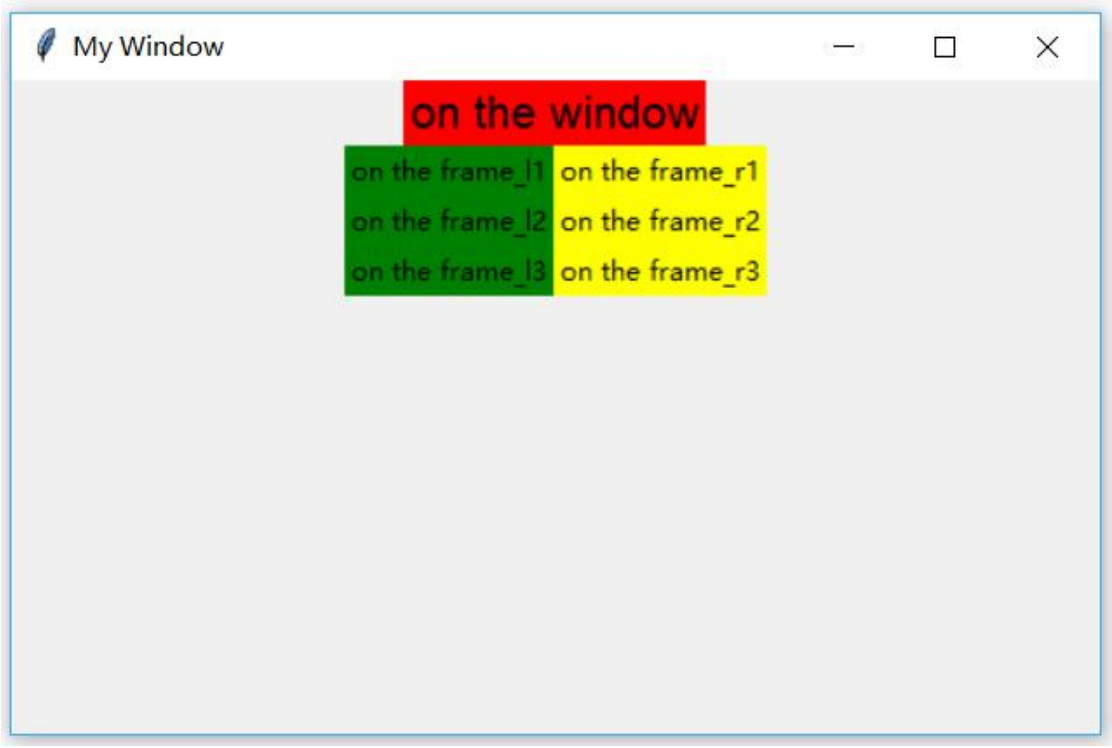
在比如像软件或网页交互界面等,有不同的界面逻辑层级和功能区域划分时可以用到，让交互界面逻辑更加清晰。

示例代码：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 第 4 步，在图形界面上创建一个标签用以显示内容并放置
tk.Label(window, text='on the window', bg='red', font=('Arial',
16)).pack()    # 和前面部件分开创建和放置不同,其实可以创建和放置一步完
成

# 第 5 步，创建一个主 frame，长在主 window 窗口上
frame = tk.Frame(window)
frame.pack()

# 第 6 步，创建第二层框架 frame，长在主框架 frame 上面
frame_l = tk.Frame(frame)# 第二层 frame，左 frame，长在主 frame 上
frame_r = tk.Frame(frame)# 第二层 frame，右 frame，长在主 frame 上
frame_l.pack(side='left')
frame_r.pack(side='right')

# 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
tk.Label(frame_l, text='on the frame_l1', bg='green').pack()
tk.Label(frame_l, text='on the frame_l2', bg='green').pack()
tk.Label(frame_l, text='on the frame_l3', bg='green').pack()
tk.Label(frame_r, text='on the frame_r1', bg='yellow').pack()
tk.Label(frame_r, text='on the frame_r2', bg='yellow').pack()
tk.Label(frame_r, text='on the frame_r3', bg='yellow').pack()

# 第 8 步，主窗口循环显示
window.mainloop()
```

.

测试效果：



12. 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：

```
tkinter.messagebox.showinfo(title='Hi', message='你好！')
# 提示信息对话窗
tkinter.messagebox.showwarning(title='Hi', message='有警告！')
# 提出警告对话窗
tkinter.messagebox.showerror(title='Hi', message='出错了！')
# 提出错误对话窗
print(tkinter.messagebox.askquestion(title='Hi', message='你好！'))
# 询问选择对话窗 return 'yes', 'no'
print(tkinter.messagebox.askyesno(title='Hi', message='你好！'))
# return 'True', 'False'
print(tkinter.messagebox.askokcancel(title='Hi', message='你好！'))
# return 'True', 'False'
```

.

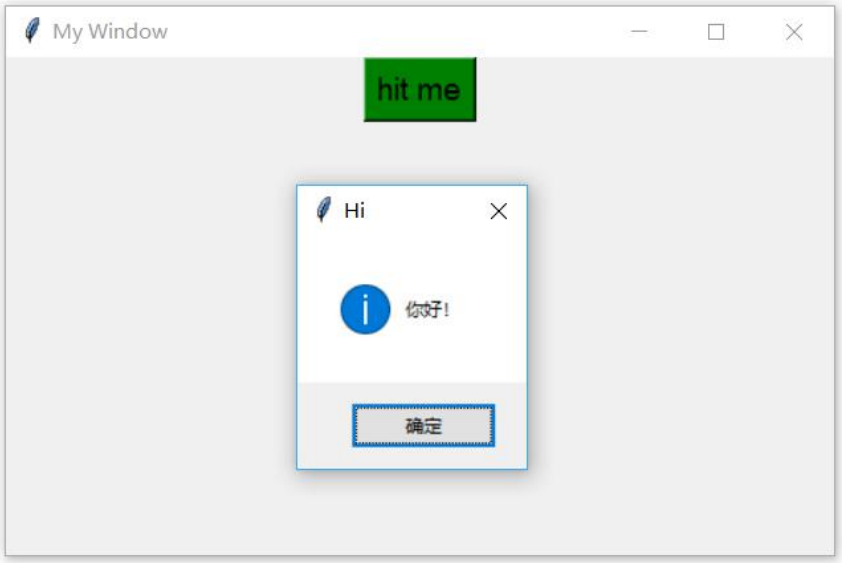
什么时候用：

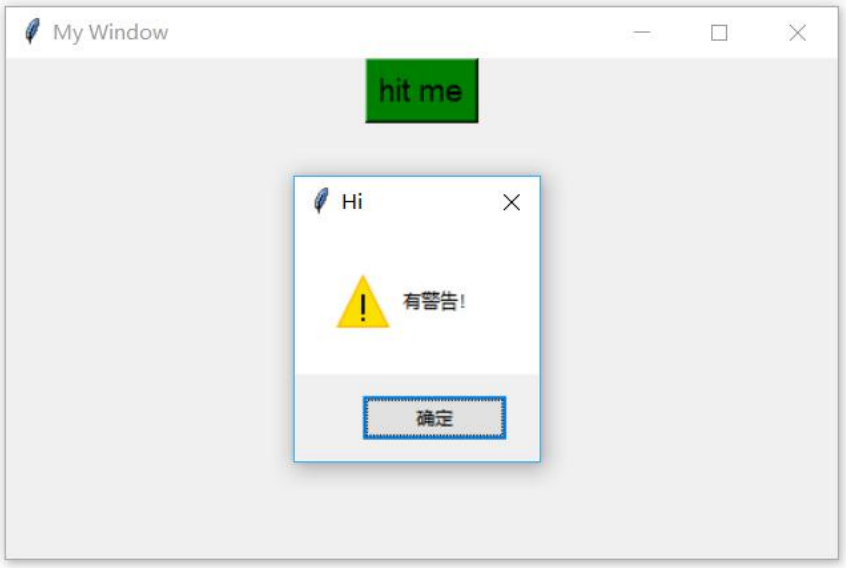
在比如像软件或网页交互界面等，有不同的界面逻辑层级和功能区域划分时可以用到，让交互界面逻辑更加清晰。

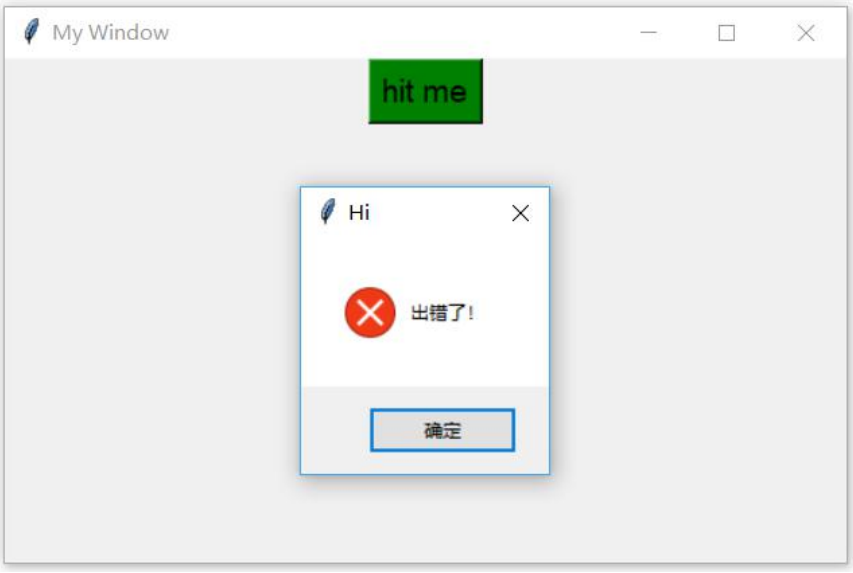
示例代码：

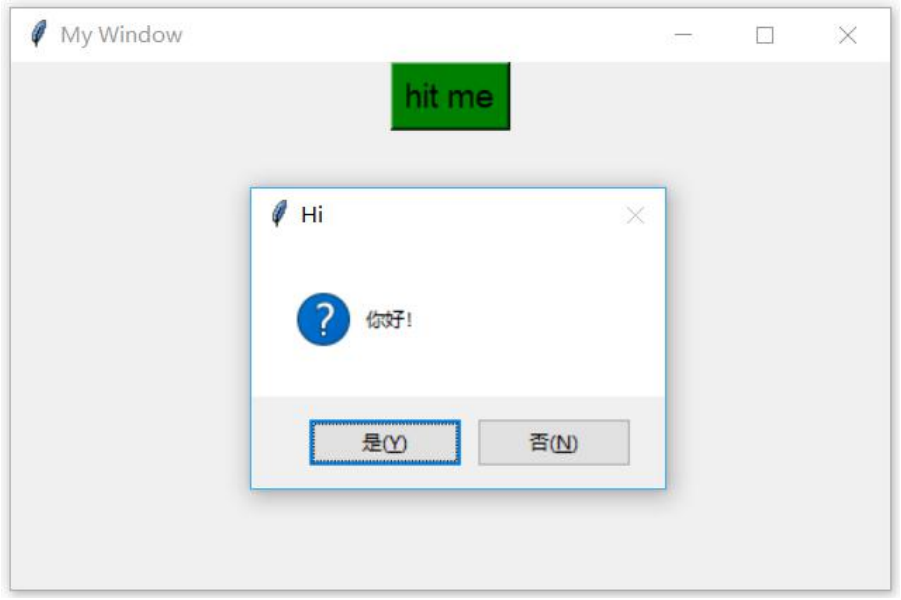
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入
import tkinter.messagebox  # 要使用 messagebox 先要导入模块

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 第 5 步，定义触发函数功能
def hit_me():
    tkinter.messagebox.showinfo(title='Hi', message='你好！')
    # 提示信息对话窗
    # tkinter.messagebox.showwarning(title='Hi', message='有警告！')
    # 提出警告对话窗
    # tkinter.messagebox.showerror(title='Hi', message='出错了！')
    # 提出错误对话窗
    # print(tkinter.messagebox.askquestion(title='Hi', message='你好！
'))  # 询问选择对话窗 return 'yes', 'no'
    # print(tkinter.messagebox.askyesno(title='Hi', message='你好！'))
    # return 'True', 'False'
    # print(tkinter.messagebox.askokcancel(title='Hi', message='你好！
'))  # return 'True', 'False'

# 第 4 步，在图形界面上创建一个标签用以显示内容并放置
tk.Button(window, text='hit me', bg='green', font=('Arial', 14),
command=hit_me).pack()

# 第 6 步，主窗口循环显示
window.mainloop()
```

测试效果：


My Window
hit me
Hi
你好!
确定


My Window
hit me
Hi
有警告!
确定


My Window
hit me
Hi
出错了!
确定



13. 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：

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

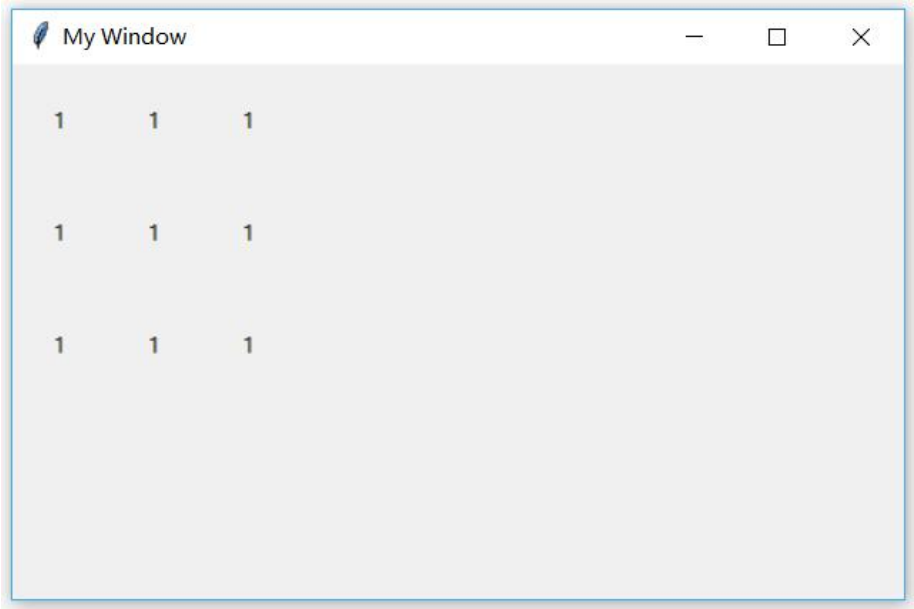
以上的代码就是创建一个三行三列的表格，其实 grid 就是用表格的形式定位的。这里的参数 row 为行，colum 为列，padx 就是单元格左右间距，pady 就是单元格上下间距，ipadx 是单元格内部元素与单元格的左右间距，ipady 是单元格内部元素与单元格的上下间距。

示例代码：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
```

```
window = tk.Tk()

# 第2步，给窗口的可视化起名字
window.title('My Window')

# 第3步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 第4步，grid 放置方法
for i in range(3):
    for j in range(3):
        tk.Label(window, text=1).grid(row=i, column=j, padx=10, pady=10
, ipadx=10, ipady=10)

# 第5步，主窗口循环显示
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：

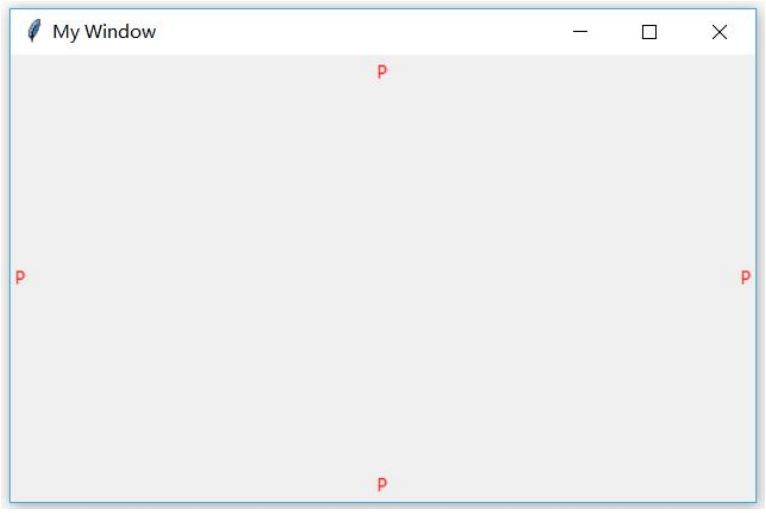
```
tk.Label(window, text='P', fg='red').pack(side='top')    # 上
tk.Label(window, text='P', fg='red').pack(side='bottom') # 下
tk.Label(window, text='P', fg='red').pack(side='left')   # 左
tk.Label(window, text='P', fg='red').pack(side='right')  # 右
```

示例代码：

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 第 4 步，pack 放置方法
tk.Label(window, text='P', fg='red').pack(side='top')     # 上
tk.Label(window, text='P', fg='red').pack(side='bottom') # 下
tk.Label(window, text='P', fg='red').pack(side='left')    # 左
tk.Label(window, text='P', fg='red').pack(side='right')   # 右

# 第 5 步，主窗口循环显示
window.mainloop()
```

.

测试效果：



3. Place：The Place Geometry Manager

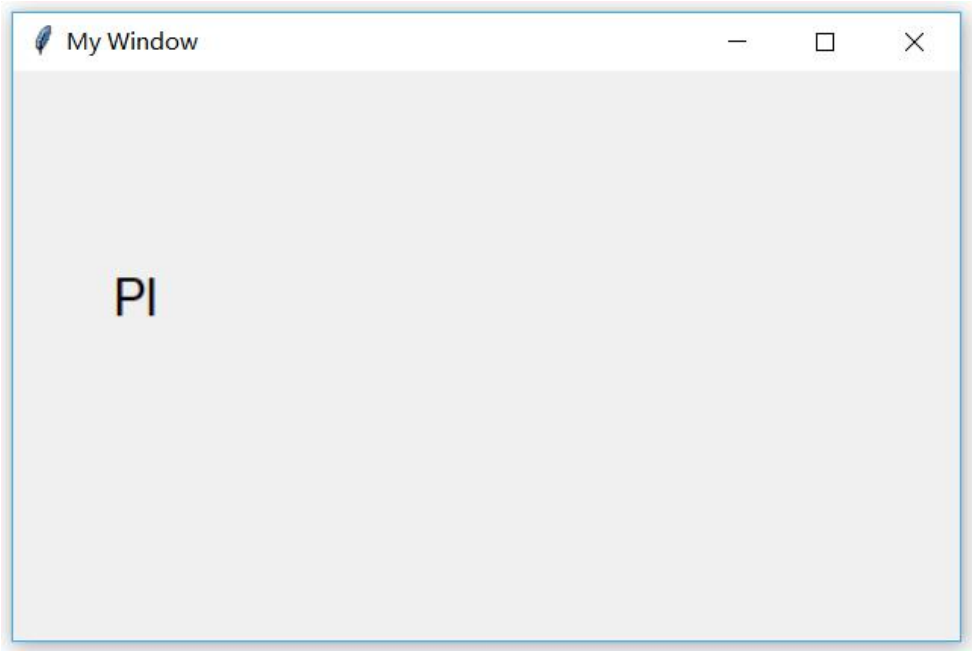
再接下来我们来看 place()，这个比较容易理解，就是给精确的坐标来定位，如此处给的(50, 100)，就是将这个部件放在坐标为(x=50, y=100)的这个位置，后面的参数 anchor='nw'，就是前面所讲的锚定点是西北角。例如：

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

示例代码：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('My Window')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('500x300')  # 这里的乘是小 x

# 第 4 步，place 放置方法（精准的放置到指定坐标点的位置上）
tk.Label(window, text='PI', font=('Arial', 20), ).place(x=50, y=100, anchor='nw')

# 第 5 步，主窗口循环显示
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：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author:洪卫

import tkinter as tk  # 使用 Tkinter 前需要先导入
import tkinter.messagebox
import pickle

# 第 1 步，实例化 object，建立窗口 window
window = tk.Tk()

# 第 2 步，给窗口的可视化起名字
window.title('Wellcome to Hongwei Website')

# 第 3 步，设定窗口的大小(长 * 宽)
window.geometry('400x300')  # 这里的乘是小 x

# 第 4 步，加载 wellcome image
canvas = tk.Canvas(window, width=400, height=135, bg='green')
image_file = tk.PhotoImage(file='pic.gif')
image = canvas.create_image(200, 0, anchor='n', image=image_file)
canvas.pack(side='top')
tk.Label(window, text='Wellcome',font=('Arial', 16)).pack()

# 第 5 步，用户信息
tk.Label(window, text='User name:', font=('Arial', 14)).place(x=10,
y=170)
tk.Label(window, text='Password:', font=('Arial', 14)).place(x=10,
y=210)

# 第 6 步，用户登录输入框 entry
# 用户名
var_usr_name = tk.StringVar()
var_usr_name.set('example@python.com')
entry_usr_name = tk.Entry(window, textvariable=var_usr_name,
font=('Arial', 14))
```

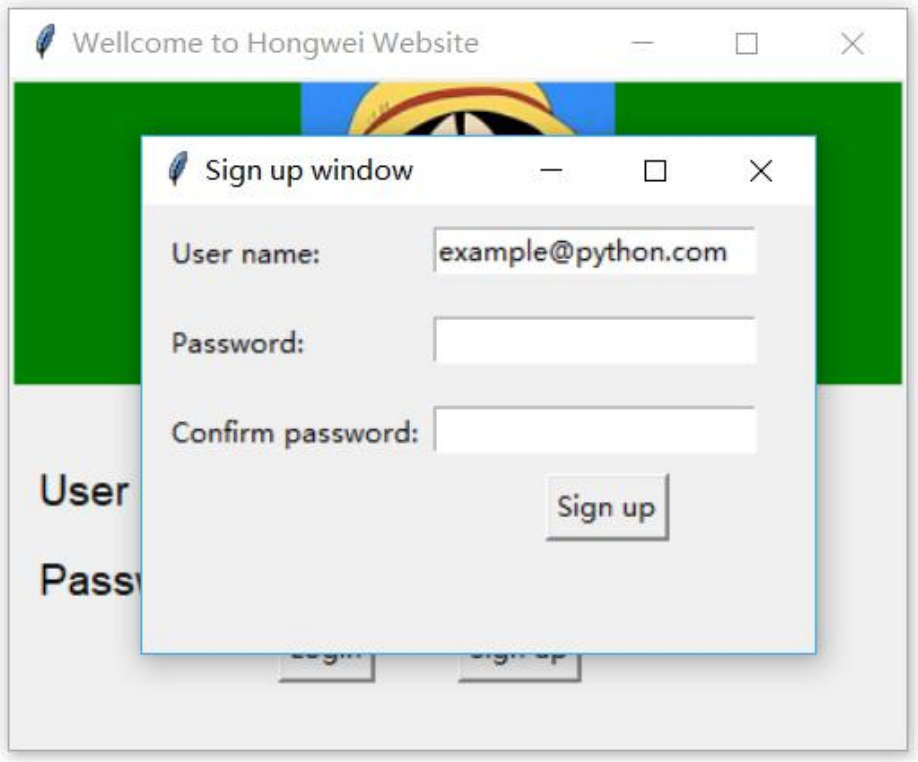
```
entry_usr_name.place(x=120,y=175)
# 用户密码
var_usr_pwd = tk.StringVar()
entry_usr_pwd = tk.Entry(window, textvariable=var_usr_pwd,
font=('Arial', 14), show='*')
entry_usr_pwd.place(x=120,y=215)

# 第8步，定义用户登录功能
def usr_login():
    # 这两行代码就是获取用户输入的usr_name和usr_pwd
    usr_name = var_usr_name.get()
    usr_pwd = var_usr_pwd.get()

    # 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。
    # 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。
    try:
        with open('usrs_info.pickle', 'rb') as usr_file:
            usrs_info = pickle.load(usr_file)
    except FileNotFoundError:
        # 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
        # 的用户和密码写入，即用户名为`admin`密码为`admin`。
        with open('usrs_info.pickle', 'wb') as usr_file:
            usrs_info = {'admin': 'admin'}
            pickle.dump(usrs_info, usr_file)
            usr_file.close()    # 必须先关闭，否则pickle.load()会出现
EOFError: Ran out of input

    # 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。
    if usr_name in usrs_info:
        if usr_pwd == usrs_info[usr_name]:
            tkinter.messagebox.showinfo(title='Welcome', message='How
are you? ' + usr_name)
        # 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.'
        else:
            tkinter.messagebox.showerror(message='Error, your password
is wrong, try again.')
    else:  # 如果发现用户名不存在
        is_sign_up = tkinter.messagebox.askyesno('Welcome! ', 'You have
not sign up yet. Sign up now?')
        # 提示需不需要注册新用户
```

```
        if is_sign_up:
            usr_sign_up()

# 第9步，定义用户注册功能
def usr_sign_up():
    def sign_to_Hongwei_Website():
        # 以下三行就是获取我们注册时所输入的信息
        np = new_pwd.get()
        npf = new_pwd_confirm.get()
        nn = new_name.get()

        # 这里是打开我们记录数据的文件，将注册信息读出
        with open('usrs_info.pickle', 'rb') as usr_file:
            exist_usr_info = pickle.load(usr_file)
        # 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!
        if np != npf:
            tkinter.messagebox.showerror('Error', 'Password and
confirm password must be the same!')

        # 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!
        elif nn in exist_usr_info:
            tkinter.messagebox.showerror('Error', 'The user has already
signed up!')

        # 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!，然后销毁窗口。
        else:
            exist_usr_info[nn] = np
            with open('usrs_info.pickle', 'wb') as usr_file:
                pickle.dump(exist_usr_info, usr_file)
            tkinter.messagebox.showinfo('Welcome', 'You have
successfully signed up!')
            # 然后销毁窗口。
            window_sign_up.destroy()

    # 定义长在窗口上的窗口
    window_sign_up = tk.Toplevel(window)
    window_sign_up.geometry('300x200')
    window_sign_up.title('Sign up window')

    new_name = tk.StringVar()  # 将输入的注册名赋值给变量
```

```
    new_name.set('example@python.com')  # 将最初显示定为
'example@python.com'
    tk.Label(window_sign_up, text='User name: ').place(x=10, y=10)
    #将`User name:`放置在坐标（10,10）。
    entry_new_name = tk.Entry(window_sign_up, textvariable=new_name)
    # 创建一个注册名的`entry`，变量为`new_name`
    entry_new_name.place(x=130, y=10)  # `entry`放置在坐标（150,10）.

    new_pwd = tk.StringVar()
    tk.Label(window_sign_up, text='Password: ').place(x=10, y=50)
    entry_usr_pwd = tk.Entry(window_sign_up, textvariable=new_pwd,
show='*')
    entry_usr_pwd.place(x=130, y=50)

    new_pwd_confirm = tk.StringVar()
    tk.Label(window_sign_up, text='Confirm password: ').place(x=10,
y=90)
    entry_usr_pwd_confirm = tk.Entry(window_sign_up,
textvariable=new_pwd_confirm, show='*')
    entry_usr_pwd_confirm.place(x=130, y=90)

    # 下面的 sign_to_Hongwei_Website
    btn_comfirm_sign_up = tk.Button(window_sign_up, text='Sign up',
command=sign_to_Hongwei_Website)
    btn_comfirm_sign_up.place(x=180, y=120)

# 第7步，login and sign up 按钮
btn_login = tk.Button(window, text='Login', command=usr_login)
btn_login.place(x=120, y=240)
btn_sign_up = tk.Button(window, text='Sign up', command=usr_sign_up)
btn_sign_up.place(x=200, y=240)

# 第10步，主窗口循环显示
window.mainloop()
```

.

测试效果：



15. 其他部件后续再补充...

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。

本文摘自：https://www.cnblogs.com/shwee/p/9427975.html