《tkinter 教程》

简介

- 作者: 魏明择
- 组织：达内科技

目录

Tkinter

- Tkinter是基于Python 实现的 GUI(Graphical User Interface)图形用户接口
- 是指采用图形方式显示的计算机操作用户界面。

GUI的作用：

- 用图形交互方式操作机算机

GUI简介

GUI的种类:

- Qt
- GTK
- MFC
- ...

python下GUI的种类:

- tkinter(当前教学)
- pyQt
- wxpython

tkinter 官网

- http://effbot.org/tkinterbook/tkinter-index.htm

tkinter 安装:

- sudo apt-get install tk-dev
- sudo apt-get install python3-tk

tkinter包的导入:

```
import tkinter
或
from tkinter import xxx
或
from tkinter import *
```

从tkinter的顶层窗口说起

```
import tkinter  # 导入tkinter包
root = tkinter.Tk(className="这是一个未放置任何控件的白板")
print("正在进入主循环")
root.mainloop()
print("结束主循环程序退出")
```

让 GUI 程序启动和运行起来需要以下 5 个主要步骤。

1. 导入 Tkinter 包
    - import tkinter
    - from tkinter import xxx
    - from tkinter import *
2. 创建一个顶层窗口对象，用于容纳整个GUI应用;
3. 在顶层窗口对象之上(或“其中”)构建所有的 GUI 组件(及其功能);
4. 通过底层的应用代码将这些GUI组件连接起来;
5. 进入主事件.

第一个hello world 程序

```
# file : tkinter_helloworld.py
import tkinter   # 导入tkinter 包
root = tkinter.Tk()  # 创建tkinter 主
label = tkinter.Label(root, text="hello world!")
label.pack()
root.mainloop()
```

图形用户编程

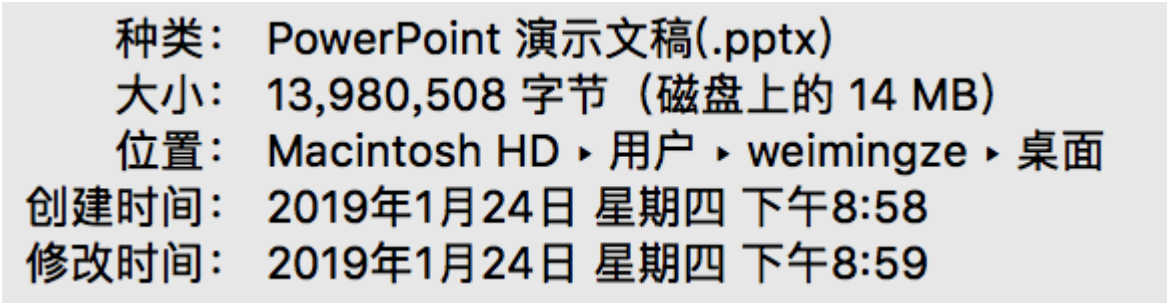
Tkinter 控件

控件	描述
Button	与 Label 类似，但提供额外的功能，如鼠标悬浮、按下、释放以及键盘活
Canvas	提供绘制形状的功能(线段、随圆、多边形、矩形)，可以包含图像或位图
Checkbutton	一组选框，可以 选其中的任意个(与 HTML 的 checkbox 输入类似)
Entry	单行文本框，用于收集键盘输入(与 HTML 的文本输入类似)
Frame	包含其他控件的纯容器
Label	用于包含文本或图像
LabelFrame	标签和框架的组合，拥有额外的标签属性
Listbox	给用户显示一个选项列表来进行选择
Menu	按下 Menubutton 后弹出的选项列表，用户可以从中选择
Menubutton	用于包含 单(下拉、级联等)
Message	消息。与 Label 类似，不过可以显示成多行
PanedWindow	一个可以控制其他控件在其中 放的容器控件
Radiobutton	一组按钮，其中只有一个可以“按下”(与 HTML 的 radio 输入类似)
Scale	线性“块”控件，根据已设定的起始值和终止值，给出当前设定的精确值
Scrollbar	为 Text、Canvas、Listbox、Enter 等支持的控件提供滚动功能
Spinbox	Entry 和 Button 的组合，允许对值进行调整
Text	多行文本框，用于收集(或显示)用户输入的文本(与 HTML 的 textarea 类似)
Toplevel	与 Frame 类似，不过它提供了一个单独的窗口容器

tkinter 常见的窗口部件(Widget)

- Label
- Button
- Entry
- Checkbutton

Label 控件



Label 控件作用：

- 用于显示文字信息
- 用于显示图片信息
  - Label支持的图片类型为:GIF, PGM, PPM图片

Label的属性:

属性	说明	类型
text	文字信息	str
bg	背景色(background)	str
fg	前景色(foreground)	str
width	宽(字符宽为单位)	int
height	高(行为单位)	int
font	字体	tuple
image	图片	PhotoImage

Label 控件示例

```
import  tkinter
root = tkinter.Tk()
label = tkinter.Label(root, text="hello", bg='red')
label.config(bg='green', fg='red')
label.pack(ipadx=10, ipady=20,padx=30, pady=40, anchor=tkinter.CENTER)
root.mainloop()
```

Label的属性:

属性	说明	类型
file	指定图片的路径	str

tkinter颜色表示方式:

1. 几乎全部的表示颜色的英文的字符串:
    - 'red', 'green', 'blue', 'yellow', 'gray'..
2. 十六进制的RGB(Red, Green, Blue)红绿蓝三原色显示

```
    "#FF0000"  红色  (24bit色)
    "#00FF00"  绿色
    "#0000FF"  蓝色
```

图片PhotoImage

```
import tkinter
root = tkinter.Tk()
img = tkinter.PhotoImage(file='myimage.fig')
label = tkinter.Label(root, image=img)
label.pack()
root.mainloop()
```

Tkinter 字体元组

- (字体族， 字体大小, 样式)

```
import  tkinter
root = tkinter.Tk()
label = tkinter.Label(root, text="hello", font=('黑体', 30, 'bold'))
label.pack()
root.mainloop()
```

Button 按钮

- 

Button 控件作用:

- 给计算机提供一个命令输入

Button 控件 属性:

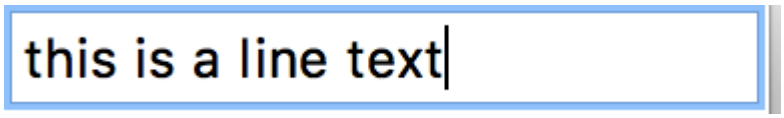
属性	说明	类型
text	文字	str
fg	前景色	str

属性	说明	类型
bg	背景色	str
width	宽(像素)	int
image	图片	PhotoImage
bitmap	位图	str='error', 'info', 'question'
command	执行回调操作	function

Button 控件示例

```
import tkinter
root = tkinter.Tk()
btn = tkinter.Button(root, text="点我退出!", command=root.destroy)
btn.pack()
root.mainloop()
```

Entry 控件

- 

Entry 控件作用

- 用于获取用户的文本输入(input类似)

Entry 控件属性

属性	说明	类型
fg	前景色	str
bg	背景色	str
width	宽(像素)	int
borderwidth	边框宽度	int
以下需要补充		
cursor	光标	???
relief	???	
state	???	
xscrollcommand	???	

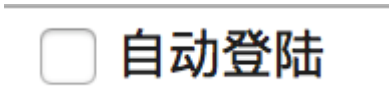
Entry控件方法：

方法	说明	返回类型
get()	获取文本框的内容	str

####### Entry控件示例

```

```

Checkbutton 控件

- 

Checkbutton 控件属性

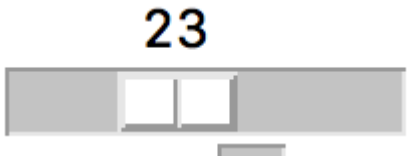
属性	说明	类型
text	文字	str
fg	前景色	str
bg	背景色	str
width	宽(像素)	int
height	商(像素)	int
image	图片	PhotoImage
bitmap	位图	str='error', 'info', 'question'
command	执行回调操作	function
textvariable	绑定文字的字符串变量	StringVar
variable	tkinter的整数变量	IntVar
font	字体(字体，大小，样式)	
anchor	???	
cursor	???	
font	??	

Checkbutton 控件示例

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
```

```
import tkinter

root = tkinter.Tk()

def onCheckButton():
    print("保存密码的值", v.get())

v = tkinter.IntVar(root, value = 2)
checkbtn = tkinter.Checkbutton(root, text='保存密码', variable=v,
command=onCheckButton)
checkbtn.pack()

checkbtn = tkinter.Checkbutton(root, text='自动登陆')
checkbtn.pack()

root.mainloop()
```

scale 控件

- 

scale 控件作用

- 用于获取用户的文本输入(写input类似)

scale 控件属性

属性	说明	类型
fg	前景色	str
bg	背景色	str
label	???	
orient	方向	HORIZONTAL(水平) , VERTICAL(竖直)
width	宽(像素)	int
borderwidth	边框宽度	int
from_	起始整数值	int
to	终止值(包含)	int
以下需要补充		
command	值变化回调函数	def moved(value):

属性	说明	类型
variable	绑定变量	IntVar
relief	???	
state	???	
xscrollcommand	???	

scale 控件方法

方法	说明	返回类型
get()	获取当前值	int
set(value)	设置当前值	int

Scale控件command 回调(callback)函数:

```
def xxxx(v):  # v代表当前的值
    ....
```

scale 控件示例

```
"""用scale 控件控制回调函数"""
def resize(event=None):
    print(scale.get())

import tkinter
root = tkinter.Tk()
scale1 = tkinter.Scale(root, from_=12, to=40, orient=tkinter.HORIZONTAL,
command=resize)
scale1.set(20)
scale1.pack()
scale2 = tkinter.Scale(root, from_=12, to=40, orient=tkinter.VERTICAL,
command=resize)
scale1.pack()
root.mainloop()
```

布局:

- pack 打包布局
- grid 网格布局
- place 放置布局(已被弃用)

窗口坐标:

- X轴水平方向
- Y轴竖直向下
- Z轴由内向外垂直屏幕

pack 打包布局

pack打包布局 说明

- 此种布局类似向行李箱里摆放东西

pack 常用属性:

属性	说明	类型
side	停靠窗口位置	str='top', 'bottom', 'left', 'right'
padx	横向外边距离	int
pady	纵向外边距离	int
ipadx	横向内边距离	int
ipady	纵向内边距离	int
fill	填充方向	tkinter.NONE, tkinter.X, tkinter.Y, tkinter.BOTH
expand	是否扩展	int = 1或0(默认值)

side 属性:

- 用来设置打包放置边
- 值:
    - tkinter.LEFT (左)
    - tkinter.RIGHT (右)
    - tkinter.TOP (上)
    - tkinter.BOTTOM (下)

fill属性:

- 用于在某个方向上填充空白区域
- 值:
    - tkinter.NONE (默认)
    - tkinter.X (水平方向)
    - tkinter.Y (竖直方向)
    - tkinter.BOTH (水平竖直两个方向)

expand 属性

- 填充整个空白区域
- 值:

    - 0 不扩展(默认)
    - 1 扩展
- 说明:
    - 当expand值为1时，side 属性设置无效

ipadx/ipady 属性

- 设置内边距的值,默认单位为像素
- 值:
    - 0~n的整数值

padx/pady 属性

- 设置外边距的值，默认单位为像素
- 值:
    - 0~n的整数(默认为0)

含有两个控件的窗口的布局

```
import tkinter
root = tkinter.Tk()
label = tkinter.Label(root, text="你好，欢迎来玩TKInter!")
label.pack()
btn = tkinter.Button(root, text="点我退出!", command=root.destroy)
btn.pack(fill=tkinter.X, expand=1)
root.mainloop()
```

含有三个控件的窗口的布局

```
#  +--------+----+
#  |        |    |
#  +--------+    |
#  |        |    |
#  +--------+----+
import tkinter
root = tkinter.Tk()
tkinter.Button(root, text="左").pack(side="left")
tkinter.Button(root, text="右上").pack(side="top")
tkinter.Button(root, text="右下").pack(side="bottom")
root.mainloop()
```

```
#  +--------+----+
#  |        |    |
#  +--------+    |
#  |        |    |
```

```
#  +--------+----+
import tkinter  # 导入tkinter包
root = tkinter.Tk()
tkinter.Button(root, text="右").pack(side="right")
tkinter.Button(root, text="左上").pack(side="top")
tkinter.Button(root, text="左下").pack(side="bottom")
root.mainloop()
```

```
#  +-------------+
#  |             |
#  +------+------+
#  |      |      |
#  +------+------+
import tkinter  # 导入tkinter包
root = tkinter.Tk()
tkinter.Button(root, text="上").pack(side="top")
tkinter.Button(root, text="左下").pack(side="left")
tkinter.Button(root, text="右下").pack(side="right")
root.mainloop()
```

```
#  +------+------+
#  |      |      |
#  +------+------+
#  |             |
#  +-------------+
import tkinter  # 导入tkinter包
root = tkinter.Tk()
tkinter.Button(root, text="下").pack(side="bottom")
tkinter.Button(root, text="左上").pack(side="left")
tkinter.Button(root, text="右上").pack(side="right")
root.mainloop()
```

```
#  +-------------+
#  |             |
#  +------+------+
#  |      |      |
#  +------+------+
#  |             |
#  +-------------+
import tkinter  # 导入tkinter包
root = tkinter.Tk()
tkinter.Button(root, text="上").pack(side="top")
tkinter.Button(root, text="下").pack(side="bottom")
tkinter.Button(root, text="左上").pack(side="left")
tkinter.Button(root, text="右上").pack(side="right")
root.mainloop()
```

grid 网格布局

grid 常用属性：

属性	说明	类型
row	行	int
column	列	int
padx	横向外边距离	int
pady	纵向外边距离	int
ipadx	横向内边距离	int
ipady	纵向内边距离	int
rowspan	跨越行数	int
columnspan	跨越列数	int
sticky	粘滞方向 (N+E+S+W)	int

grid 网格布局示例

```
import tkinter
root = tkinter.Tk()

btn1 = tkinter.Button(root, text="按钮1")
btn1.grid(row=0, column=0)

btn2 = tkinter.Button(root, text="按钮2")
btn2.grid(row=0, column=1)

btn3 = tkinter.Button(root, text="按钮3")
btn3.grid(row=1, column=1)

btn4 = tkinter.Button(root, text="按钮4")
btn4.grid(row=2, column=2)

root.mainloop()
```

Frame 控件

Frame 控件常用属性：

属性	说明	类型
bg	背景色	str
height	高	int

属性	说明	类型
width	宽	int
borderwidth	边框宽度	int

Frame 控件属性：

```
background/bg
height
width
borderwidth
class,
bd
colormap
container
cursor
highlightbackground,
highlightcolor
highlightthickness
relief
takefocus
visual
```

Frame 控件示例

```
import tkinter
from tkinter.constants import *
tk = tkinter.Tk()
frame = tkinter.Frame(tk, relief=RIDGE, borderwidth=2)
frame.pack(fill=BOTH,expand=1)
label = tkinter.Label(frame, text="Hello, World")
label.pack(fill=X, expand=1)
button = tkinter.Button(frame,text="Exit",command=tk.destroy)
button.pack(side=BOTTOM)
tk.mainloop()
```

窗口部件 widget 的 config 方法:

- 用于修改widget对象的属性
- 语法:
  - xxx.config(属性1=值1, 属性2=值2, ...)

示例：

```
label = tkinter.Label(root, text="hello")
lable.config(text='world')  # 将label的text属性改为'world' 字符串
```

tkinter 关联控件变量Coupling Widget Variables

已定义的关联变量

- StringVar
- IntVar
- DoubleVar
- BooleanVar

关联的选项有

- variable
- textvariable
- onvalue
- offvalue
- value

关联变量方法:

```
get()        获取相应的值
set(value)   设置相应的值
```

关联变量的属性:

- value 设置初始值

变量的构造函数:

```
class xxxVar(Variable):
    def __init__(self, master=None, value=None, name=None):
        pass

     def get(self):
         pass
```

画布 canvas

作用

- 把一定组件“画”到画布上，显示出来

构造方法:

Canvas(master=None, width=0, height=0)

canvas所支持的组件包括:

```
- bitmap
- image(BitmapImage, PhotoImage)
- line
- oval
- polygon
- rectangle
- text
- arc
```

画布方法:

```
canvas.create_xxx(x, y, other, ...)
- 返回一个创建的组件的ID，同时候也可以用tag属性指定其标签名称
```

####### 通过调用canvas.move(tags, offset_x, offset_y)实现一个移动动作 canvas.move(tagOrId, xAmount, yAmount)

```
canvas. delete(tags) 删除一个图片对象
canvas.coords(tags_or_id) 给一个对象重新定位(默认anchor=center)坐标点位于中心点
.coords(tagOrId, x0, y0, x1, y1, ..., xn, yn)
```

btn 的事件函数:

```
def onClick():
    "此函数用来作用 按扭点击时的回调函数"
    print("OnClick被调用!")

import tkinter
root = tkinter.Tk()
btn = tkinter.Button(root, text="Click me!", command=onClick)
btn.pack(fill=tkinter.X, expand=1)
root.mainloop()
```

事件 event

什么是事件

- 事件是指鼠标，键盘等对窗口元素发生的过程

事件的来源

- 按键，鼠标，或是定时器，窗口关闭等。

绑定事件

```
bind(事件类型，事件处理函数)
```

事件处事函数的定义方法:

```
def xxxx(event):
    pass
```

常用事件

```
- <Button>    鼠标的全部按键按下
- <Button-1>    数值可以为1,2,3分别代表左中右三个鼠标键
- <ButtonRelease> 鼠标的全部按键抬起
- <ButtonRelease-1> 左键抬起
- <Key> 或 <KeyPress> 全部的键盘按键按下
- <KeyPress-a>   a键被按下
- <KeyPress-q>
- <KeyPress-space>
- <KeyPress-Escape>  Esc键按下
- <KeyPress-Left>  左方向键按下
- <KeyPress>      键被按下
- <KeyRelease>    键被抬起
- <KeyRelease-a>  a键抬起
- <Control-v>     Ctrl+v
- <F1>            F1键按下
- <ButtonRelease-1>
```

事件处事函数:

```
def xxxx(event):
    pass
```

```
注: event为事件类型
```

事件event的属性

事件	说明
widget	产生event的实例
type	事件类型
key event	
keycode	键盘的代码(操作系统级别的编码)
keysym	按键的符号名称(字符串)
keysym_num	按键对应的ASCII值
char	按键对应的字符(字符串)
mouse event	
x, y	鼠标相对于窗口控件的位置，单位:像素
x_root, y_root	鼠标相对于屏幕左上角的绝对位置，单位:像素
num	按钮的num编号，(仅鼠标事件)
resize event	
width, height	widget新大小

事件对象的文档

```
|  num - mouse button pressed (ButtonPress, ButtonRelease)
|  focus - whether the window has the focus (Enter, Leave)
|  height - height of the exposed window (Configure, Expose)
|  width - width of the exposed window (Configure, Expose)
|  keycode - keycode of the pressed key (KeyPress, KeyRelease)
|  state - state of the event as a number (ButtonPress, ButtonRelease,
|                          Enter, KeyPress, KeyRelease,
|                          Leave, Motion)
|  state - state as a string (Visibility)
|  time - when the event occurred
|  x - x-position of the mouse
|  y - y-position of the mouse
|  x_root - x-position of the mouse on the screen
|           (ButtonPress, ButtonRelease, KeyPress, KeyRelease, Motion)
|  y_root - y-position of the mouse on the screen
|           (ButtonPress, ButtonRelease, KeyPress, KeyRelease, Motion)
|  char - pressed character (KeyPress, KeyRelease)
|  send_event - see X/Windows documentation
|  keysym - keysym of the event as a string (KeyPress, KeyRelease)
```

```
  |  keysym_num - keysym of the event as a number (KeyPress, KeyRelease)
  |  type - type of the event as a number
  |  widget - widget in which the event occurred
  |  delta - delta of wheel movement (MouseWheel)```
```

按键事件类型

```
KeyPress, Key  按键按下
KeyRelease     按键松开
```

鼠标事件类型

```
ButtonPress, Button   鼠标按键点击
ButtonRelease         鼠标按键抬起
Motion                鼠标按键移动
MouseWheel            鼠标滚轮转动(暂不能用)
```

Widget事件

```
Enter                 进入Widget
Leave                 离开Widget(不起作用)
```

进入Widget和离开Widget

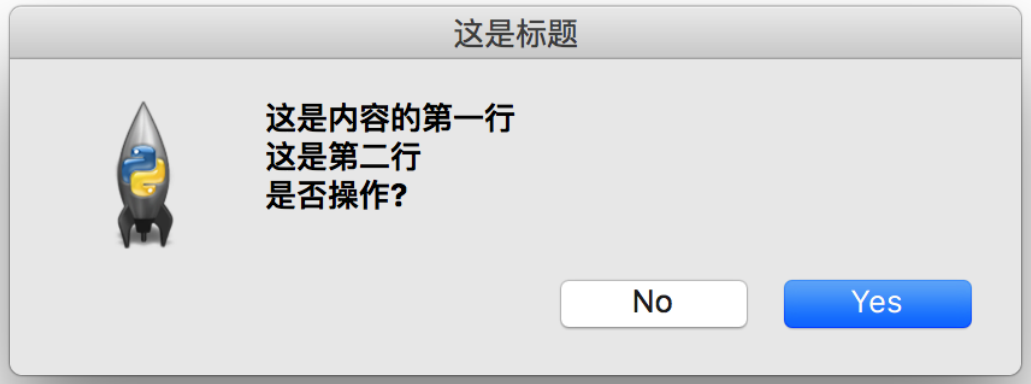
```
FocusIn     获取焦点?
FocusOut    失去焦点?
```

示例示例

```
import tkinter
root = tkinter.Tk()

def mouseDownEvent(e):
    print("鼠标左键按下, 在:",
        e.x, e.y, e.x_root, e.y_root)

def mouseUPEvent(e):
    print("鼠标左键抬起")
```

```
def keyDown(e):
    print("有按键按下")

def keyUp(e):
    print("有按键抬起")

root.bind('<Button-1>', mouseDownEvent)
root.bind('<ButtonRelease-1>', mouseUPEvent)
root.bind('<KeyPress-a>', keyDown)
root.bind('<KeyRelease-a>', keyUp)

root.mainloop()
```

消息对话框



-

作用

- 弹出消息，用于短时简单数据交互

种类

```
1. messagebox
2. filedialog
3. colorchooser
```

messagebox的构造方法:

```
1. askokcancel
2. askquesiton
3. askretrycancel
4. askyesno
5. showerror
6. showwinfo
7. showwarning
```

askyesno 返回布尔值

```
import tkinter
from tkinter import messagebox
root = tkinter.Tk()
r = messagebox.askyesno(title="这是标题", message="这是内容的第一行\n这是第二行
\n是否操作?", icon=messagebox.INFO)
print(r)
if r:
    print("OK被按下")
else:
    print("取消被按下")

root.mainloop()
```

练习项目

- 邮箱界面:

```
+------+------+
| send | save |
+------+------+-----------------+
| recive | [weimz@tedu.cn;    ] |
| title  | [welcome to beijing] |
+-------------------------------+
| Mr. Wei                       |
|     thank!                    |
| weimingze                     |
| 2018-3-25                     |
+-------------------------------+
```

项层窗口的方法:

```
import tkinter
root = tkinter.Tk()
root.title("这是窗口标题")
```

```
root.resizable(width=False, height=False)  # 禁止变化大小
root.update()  强制刷新主窗口
```

其它控件

LabelFrame 控件

- 和Frame一样，查多了一个text属性
- 

LabelFrame 控件示例：

```
# 01_labelframe.py

import tkinter

root = tkinter.Tk()

# 创建一个LabelFrame
labelframe = tkinter.LabelFrame(root, text="远程协助")
labelframe.pack()

ckbtn = tkinter.Checkbutton(labelframe, text="ABCDAAAAAAAAAAAAAA")
ckbtn.pack()

root.mainloop()
```

RadioButton 控件

- 

RadioButton 控件作用：

从多个选项中选择其中的一个

RadioButton 控件属性:

```
text        显示文字信息
value       此控件代表的值
variable    此控件绑定的 IntVar对象
command     回调函数
```

RadioButton 控件示例:

```python
# 02_radiobutton.py

import tkinter

root = tkinter.Tk()
# 设置窗口标题
root.title("请投票")

# 选创建一个LabelFrame
lf = tkinter.LabelFrame(root, text="您的性别")
lf.pack()

# 创建一个IntVar类型的变量,
# 让如下Radiobutton关联在同一个变量
v = tkinter.IntVar(root, value=0)

rb1 = tkinter.Radiobutton(lf, text='男',
              value=1,  # 设置此控件代表的值
              variable=v)  # 设置此控件关联的变量
rb1.pack()

rb2 = tkinter.Radiobutton(lf, text='女', value=0,
            variable=v)
rb2.pack()

rb3 = tkinter.Radiobutton(lf, text='保密', value=2,
            variable=v)
rb3.pack()

# 再创建另一个LabelFrame,里面放三个Radiobutton
# 分别是"支持", "反对", "弃权"
lf2 = tkinter.LabelFrame(root, text='请投票')
lf2.pack()

# 创建一个IntVar对象，默认值为1,(选中支持)
# 将其关联在以下三个按钮上
v2 = tkinter.IntVar(root, value=1)

rb4 = tkinter.Radiobutton(lf2, text="支持", value=1)
rb4.config(variable=v2)
rb4.pack(side=tkinter.LEFT)
```

```
rb5 = tkinter.Radiobutton(lf2, text="反对", value=-1)
rb5.config(variable=v2)
rb5.pack(side=tkinter.LEFT)

rb6 = tkinter.Radiobutton(lf2, text="弃权", value=0)
rb6.config(variable=v2)
rb6.pack(side=tkinter.LEFT)

def onBtn():
    print("按钮被按下")
    print("性别选择的值是:", v.get())
    print("您的投票值:", v2.get())


btn = tkinter.Button(root, text="打印所选的值:",
    command=onBtn
    )
btn.pack(side=tkinter.BOTTOM)

root.mainloop()
```

Spinbox 控件

Spinbox 控件属性:

```
from_    整数起始值
to       整数终止值(包含)
increment  增加/减少的值
```

Spinbox 控件方法:

```
get()        ...略
```

Spinbox 控件示例

```
# 05_spinbox.py
import tkinter

root = tkinter.Tk()
spin1 = tkinter.Spinbox(root, from_=0, to=100, increment=3)
spin1.pack()
root.mainloop()
```

text 控件:

- 

text 控件作用:

- 用于显示和书写多行文本

text 控件方法:

```
insert(index, 文字内容的字符串)
```

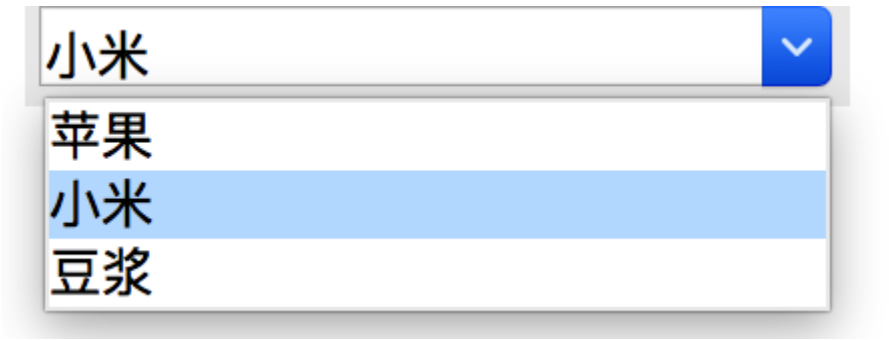
text 控件属性:

```
width  宽(以字符宽度为单位)
height 高(以文字的行为单位)
```

text 控件示例

```
# 06_text.py
import tkinter

root = tkinter.Tk()
txt = tkinter.Text(root, width=50, height=2)
txt.insert(tkinter.END, '老王日记')
txt.pack()

root.mainloop()
```

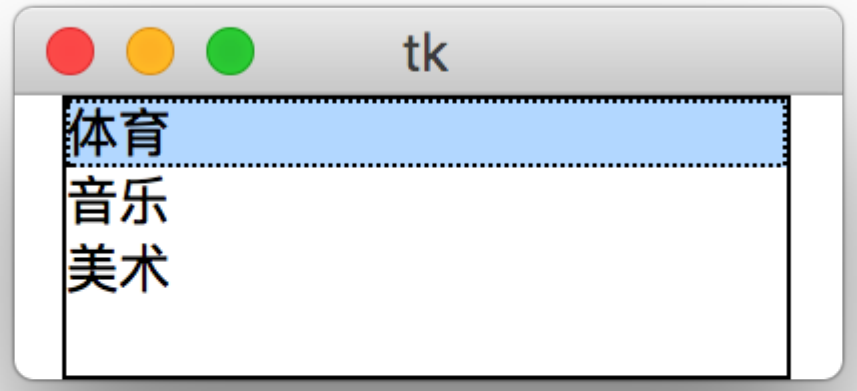
Combobox 控件

- 

示例

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-

# import tkinter
import tkinter.ttk

root = tkinter.Tk()

def onComboSelect(*args):
    print(args,'++++', combo.get())

combo = tkinter.ttk.Combobox(root, values=["苹果", '小米', '豆浆'],
state='readonly')
combo.current(1)
combo.bind('<<ComboboxSelected>>', onComboSelect)
combo.pack()

root.mainloop()
```

Listbox 控件



-

示例

```
import tkinter

root = tkinter.Tk()

lst = tkinter.Listbox(root)
lst.insert(tkinter.END, "体育")
lst.insert(tkinter.END, "音乐")
lst.insert(tkinter.END, "美术")
lst.pack()

root.mainloop()
```

menu 菜单

- 用于向APP发送指定命令

示例

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-

import tkinter
root = tkinter.Tk()

def hello():
    print("菜单被点击")

menubar = tkinter.Menu(root)

# create a pulldown menu, and add it to the menu bar
filemenu = tkinter.Menu(menubar)
filemenu.add_command(label="Open", command=hello)

recent_menu = tkinter.Menu(filemenu)
recent_menu.add_command(label="1.txt")
recent_menu.add_command(label="hello.py")
# 向filmenu 菜单加入 menu_recent_menu 菜单
filemenu.add_cascade(label="open recent", menu=recent_menu)

filemenu.add_command(label="Save", command=hello)
# filemenu.add_separator()
filemenu.add_command(label="Exit", command=root.quit)
menubar.add_cascade(label="File", menu=filemenu)

# create more pulldown menus
editmenu = tkinter.Menu(menubar, tearoff=1)
editmenu.add_command(label="Cut", command=hello)
editmenu.add_command(label="Copy", command=hello)
editmenu.add_command(label="Paste", command=hello)
menubar.add_cascade(label="Edit", menu=editmenu)
```

```
root.config(menu=menubar)

root.mainloop()
```

定时器

定时器作用

- 按给定时间触发一个回调函数（或方法)

定时器说明

- 每一个窗口控件都可以设置一个或多定时器

widget定时器设置方法:

- widget.after(延时毫秒数, 定时器回调函数)
- 说明:
  - 启动一个定时器,返回一个代表此定时器的ID(整数)

widget定时器取消方法:

- widget.after_cancel(定时器id)
- 作用:
  - 取消一个已经启动的定时器
- 注:
  - 定时器id为after返回的值

示例见:

```
# 09_timer.py
import tkinter
root = tkinter.Tk()

label = tkinter.Label(root, text="--")
label.pack()

def onTimer():
    print("定时器函数已经调用!")
    global timer_id  # 声明timer_id 为全局变量
    timer_id = label.after(1000, onTimer)  # 让定时器重复启动


def onStartTimer():
    print("已启动定时器")
    global timer_id
```

```
    timer_id = label.after(5000, onTimer)


def onStopTimer():
    label.after_cancel(timer_id)  # 取消定时器
    print("定时器已取消!")


btn = tkinter.Button(root, text="启动定时器",
                     command=onStartTimer)
btn.pack()

btn2 = tkinter.Button(root, text="取消定时器",
                      command=onStopTimer)
btn2.pack()

root.mainloop()
```