Tkinter 教程

零基础入门系列教程

作者: 魏明择

2025 年版

https://weimingze.com

目录

第一章、Tkinter图形用户界面

为什么要学习 Tkinter?

当你需要学会python 想做一个应用来处理日常事务的时候，尤其是给别人使用你写的程序的时候。使用控制台输入输出数据非常的难以操作，出错的概率高。

这时候为你的应用添加一个友好的人机界面让他人便于使用就显得尤为重要了。当你的应用只有一个交互窗口，功能又比较简单时，可以使用 Tkinter来构建你的程序界面。

Tkinter 简单高效，非常合适开发单页面程序。

以下我们将学习 Tkinter 这个 GUI 的用法。

此课程需要有 《Python 编程语言（基础篇）》 的编程能力。再学习本课程内容会便于理解。

1. GUI 简介

GUI 简介

GUI（Graphical User Interface，图形用户界面）是一种通过图形方式与计算机交互的界面，他与我们之前学习Python编程语言（基础篇）时使用命令行界面（CLI）的输入和输出不同，他是使用视觉元素如图标、按钮、窗口和菜单对程序进行输入和输出。这样大大提高了与用户交互的效率，并降低出错的概率。从而得到更好的用户体验。

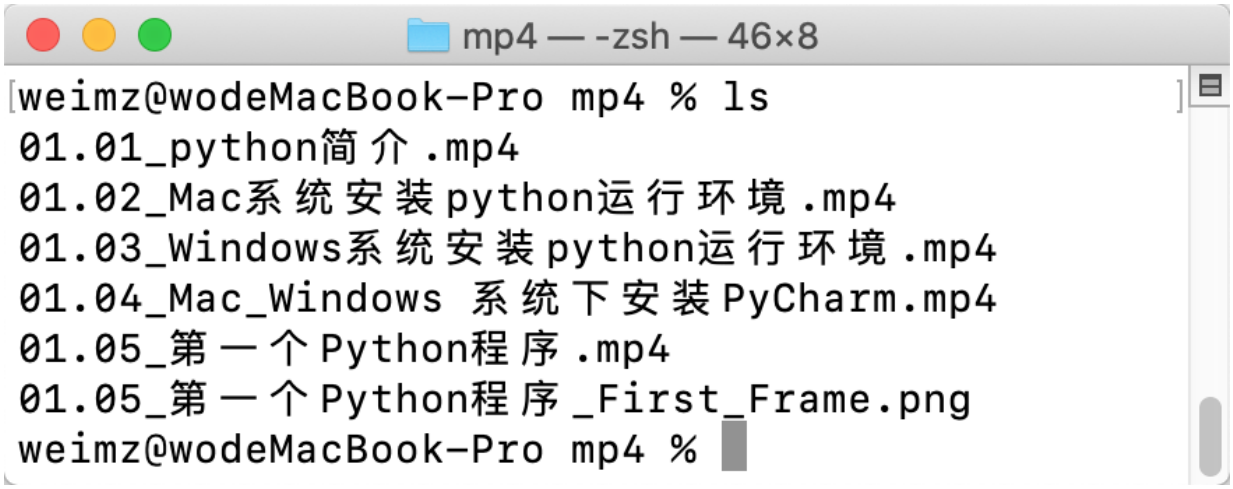
苹果公司（Macintosh）和微软（Windows）是最先在操作系统上使普及图形用户界面，逐渐取代了传统的命令行界面。

GUI的作用

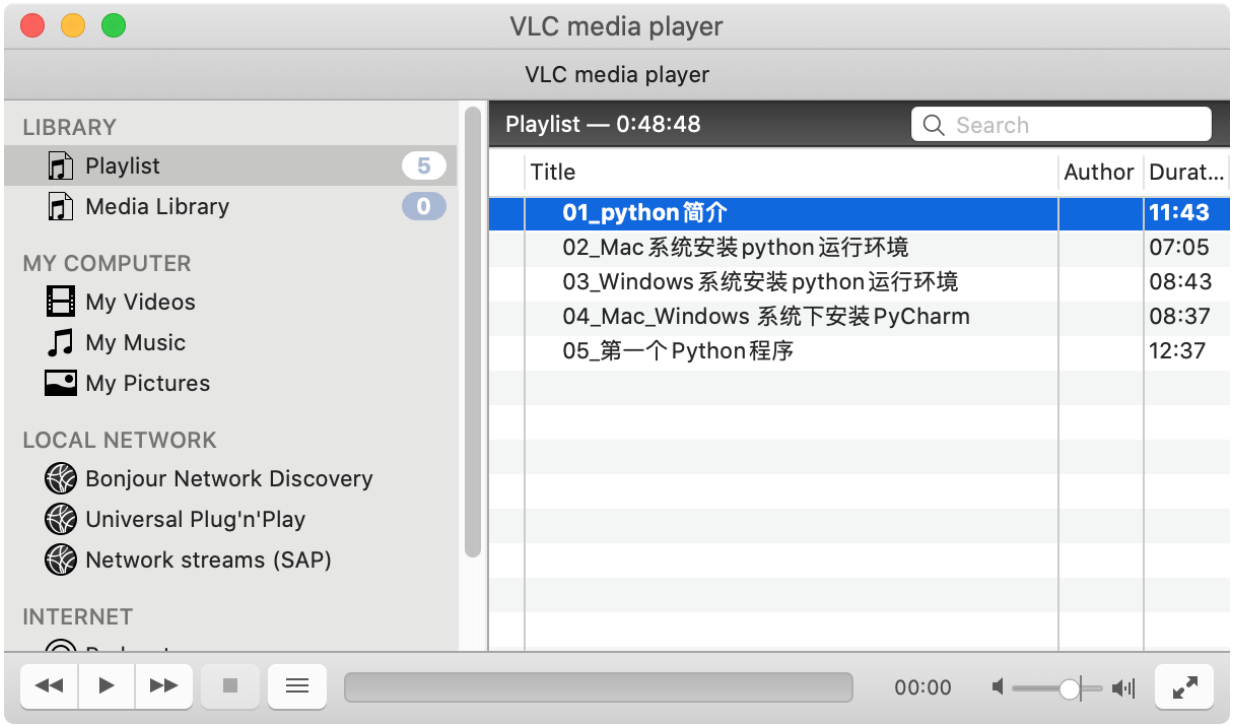
用图形界面与机算机进行交互。

如下图所示，同样是显示文件的列表，使用图形用户界面的VLC 操作起来更加简单方便友好！

控制台终端显示文件列表



使用图形用户界面的VLC 显示文件的列表



显然大家更喜欢图形用户界面。

> 老程序员一定记得 Dos 6.22 和 Win3.1/Win3.2 的区别。

桌面应用GUI的种类

- GTK：主要用于 Linux 环境。
- Qt：功能强大，支持复杂 UI 设计，适用于商业软件。
- MFC：微软的早期应用开发框架，只能用在Window上。
- wxWidgets：提供原生 UI 体验，适合需要跨平台一致性的应用。

Python下GUI的种类:

- Tkinter(当前教学)：Python内建的GUI。
- PyQt：基于C++写的Qt库, 内核依旧是Qt。
- wxPython是Python语言的一套优秀的GUI图形库。

GUI的核心组件

- 窗口：应用程序的主要显示区域。
- 菜单：提供命令和选项列表。
- 图标：代表文件、程序或功能的图形符号。
- 按钮：可点击执行特定操作的控件。
- 对话框：用于用户输入或系统提示的临时窗口。

本课程主要讲解 Tkinter 的使用。

2. Tkinter 简介

在Python中，Tkinter是一个非常流行的GUI（图形用户界面）工具包，它提供了创建窗口、按钮、文本框等GUI元素的能力。Tkinter是基于Tcl/Tk库的，但是它是纯Python编写的，因此在大多数Python安装中都可以使用。

因为 Python 有良好的跨平台特性，因此使用 Tkinter 写的界面可以运行在 Mac、Linux、Windows操作系统上。

对于硬件平台，只要能安装 Python 解释器，则可以使用 Tkinter，比如台式电脑、树莓派和香橙派等设备都能跑 Tkinter。

在大多数 Python 安装解释执行器在安装中就默认安装 Tkinter 包。你可以通过运行以下命令来检查是否安装了Tkinter：

```
>>> import tkinter
>>> print(tkinter.TkVersion)
8.6
```

以前有部分 Python 没有将 Tkinter 纳入 标准库模块。此时需要手动安装。比如在 Ubuntu 系统上，你可以使用如下命令来进行安装：

```
sudo apt-get install python3-tk
```

tkinter 的官方文档

- 网址：https://docs.python.org/zh-cn/3/library/tkinter.html

下面我们来开始写 图形用户界面的程序！

3. 第一个 Tkinter 应用

先尝试运行如下的代码吧！

```
# 1. 导入tkinter包
import tkinter

# 2. 创建一个 Tkinter 主窗口
root = tkinter.Tk(className="白板窗口")

# 3. 进入主事件循环
root.mainloop()

print("结束主事件循环，程序退出")
```

运行效果

恭喜你完成可第一个图形用户界面的程序。

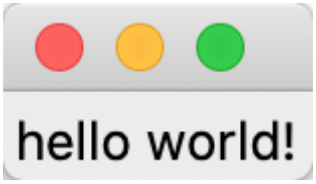
上述程序中 第二步 `root = tkinter.Tk(className="白板窗口")` 是创建一个窗口对象。但此窗口并不能正常显示和接收任何信息。

第三步：`root.mainloop()`，在此`mainloop()`方法的内部其实就是一个死循环，在循环的内部不停地检查是否有键盘事件和鼠标事件，如果有就做出相应的操作。直至此应用中所有的窗口都关闭，此时这个 `mainloop()` 函数才返回。

再来一个 "hello world" 程序!


```
import tkinter

# 1. 创建一个 Tkinter 主窗口，连窗口的标题也不给出!
root = tkinter.Tk(className="Label示例")

# 2. 创建一个 Label 窗口小部件（Widget），将他放在刚才创建的主窗口上。
label = tkinter.Label(root, text="http://weimingze.com")

# 3. 使用Pack布局方式，label在主窗口的布局方式是pack布局（后面会将）
label.pack()

# 4. 进入主事件循环
root.mainloop()
```

执行效果



看到效果后，你会发现，hello world 实在一个Label中，而Lable 又在 主窗口中，主窗口变成了Label一样的大小，你可以用鼠标调整主窗口的大小，这样就能看到其实"hello world" 只是主窗口上的一个部件(Widget)。

关于第三步 `label.pack()` 是设置摆放方式，我们后面再讲。

让 GUI 程序启动和运行起来大概需要以下 5 个主要步骤：

1. 导入 Tkinter 包：
   - `import tkinter`
   - `from tkinter import Tk, Label, Entry`等。
   - `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。

第二章、控件

1. 控件的概念

什么是控件?

控件（Widget） 是 GUI（图形用户界面）的基本组成部分，用于与用户交互或显示信息。每个控件都是 tkinter 模块中的一个类，可以创建窗口、按钮、文本框等界面元素。

如下图所示：

Tkinter 的控件(全部)

控件	控件名称	描述
Button	按钮	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	画布	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	复选框	一组选框，可以 选其中的任意个(与 HTML 的 checkbox 输入类似)
Entry	输入框	单行文本框，用于收集键盘输入(与 HTML 的文本输入类似)
Frame	框架	包含其他控件的纯容器
Label	标签	用于包含文本或图像
LabelFrame	标签框架	标签和框架的组合，拥有额外的标签属性
Listbox	列表框	给用户显示一个选项列表来进行选择
Menu	菜单	按下 Menubutton 后弹出的选项列表，用户可以从中选择
Menubutton	菜单选项	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	面板窗口	一个可以控制其他控件在其中 放的容器控件
Radiobutton	单选框	一组按钮，其中只有一个可以“按下”(与 HTML 的 radio 输入类似)
Scale	滑块	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	滚动条	为 Text、Canvas、Listbox、Enter 等支持的控件提供滚动功能
Spinbox	微调输入框	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	顶级窗口	与 Frame 类似，不过它提供了一个单独的窗口容器

> 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

后续小节中我们会依次介绍各个控件的使用。

2. 标签（Label） 控件

作用

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

类名

```
Label
```

标签控件的类名是 Label，使用前你可以使用 import 语句进行导入，一般我们常用的导入方法是

```
# 直接导入模块
import tkinter
# 使用时
label = tkinter.Label(...)

或者导入类 Label
from tkinter import Label
# 使用时
label = Label(...)
```


以下是 标签 控件的属性。通过属性可以设置控件的各种状态。

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

在Tkinter中，属性可以在创建时通过该类的构造函数的关键字传参的方式来设置，属性名就是关键字传参的形参名。如：

```
import  tkinter
label = tkinter.Label(root, text="hello", bg='red', width=200)
```

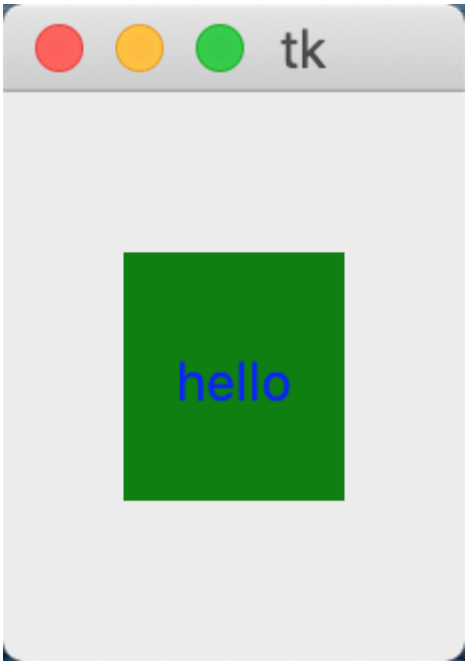
各个控件的属性也可以在运行过程中动态的修改。修改各个控件的属性需要使用控件的`config()`方法。如：

```
label.config(gb='blue', fg='green')
```

Label 控件示例

```
# 导入Tkinter 模块
import  tkinter
# 创建主窗口
root = tkinter.Tk()
# 创建一个标签控件，父窗口是主窗口 root，文字为'hello'，背景色是红色
label = tkinter.Label(root, text="hello", bg='red')
# 修改背景色为绿色，文字颜色为蓝色。
label.config(bg='green', fg='blue')
# 使用打包布局放置，后面再讲。
label.pack(ipadx=10, ipady=20,padx=30, pady=40, anchor=tkinter.CENTER)
# 进入主事件循环。
root.mainloop()
```

运行效果



上面用到了 Tkinter 的颜色表示方法。以下介绍颜色表示方法：

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
```

窗口部件 widget 的 config 方法:

用于修改widget对象的属性

语法:

```
widget.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
```

其他的属性你也来试试吧!

3. 按钮控件

作用:

给计算机提供一个命令输入。

类名

```
Button
```

Button 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
image	图片	PhotoImage
bitmap	位图	str='error', 'info', 'question'
command	执行回调操作	function

> Tkinter 控件的大部分属性基本相同。
>
> command 属性用于给一个无参数的函数或者对象的方法（也为无参数方法），当按钮被按下时会调用这个函数或方法来完成按下的操作。
>
> 需要注意的一点是此方法要尽快返回，因为Tkinter 的单线程的，如果command指定的函数不返回，程序会卡死。

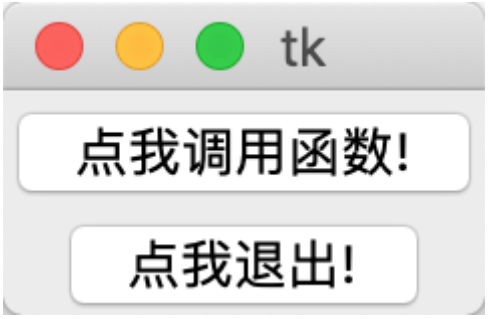
Button 控件示例

```
import tkinter

def on_button():
```

```
    print('欢迎来到 http://weimingze.com!')

root = tkinter.Tk()
btn1 = tkinter.Button(root, text="点我调用函数!", command=on_button)
btn2 = tkinter.Button(root, text="点我退出!", command=root.destroy)
btn1.pack()
btn2.pack()
root.mainloop()
```

> `root.destroy()` 方法被调用，则主窗口就会被销毁，就相当于点击 了关闭的 `x` 按钮!

效果如图所示:



4. 输入框控件

Entry 控件作用

用于获取用户的文本输入(和内建函数 input 作用相同)

类名

```
Entry
```

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

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

```
import tkinter

def on_button():
    print('按钮被按下! 文本框的内容: ', text1.get())

root = tkinter.Tk(className="魏明择的Entry")
# 创建一个文本框对象
text1 = tkinter.Entry(root)
text1.pack()
btn1 = tkinter.Button(root, text="打印文字", command=on_button)
btn1.pack()
root.mainloop()
```

效果如图所示:

5. 复选框控件

复选框（Checkbutton）是 Tkinter 中用于创建二进制选择（选中/未选中）的控件，允许用户从多个选项中选择一个或多个选项。

作用

- 用于布尔类型的输入
- 可以从多个选项中进行多选
- 用于设置选项和问答等场景。

类名

```
Entry
```

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

IntVar 变量

在使用Checkbutton 时，我们可以绑定一个 IntVar 类型的变量，让这个变量关联 某个 Checkbutton。这时变量和Checkbutton的状态就关联在了一起。一个变化，另一个也会跟着变。

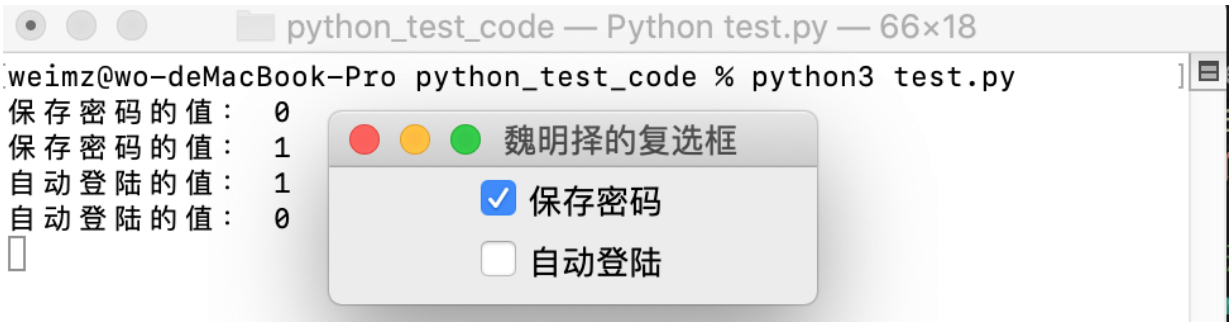
- 当 IntVar 的变量的值为1是，Checkbutton 是选中状态。
- 当 IntVar 的变量的值为0是，Checkbutton 是非选中状态。

IntVar 变量的方法

属性	说明	类型
get()	获取值	int
set(x)	设置值(x)	int

Checkbutton 控件示例

```
import tkinter

root = tkinter.Tk(className='魏明择的复选框')

def onCheckButton():
    print("保存密码的值: ", v1.get())

# 创建一个 整形变量IntVar 对象，让这个对象和 checkbtn1 关联。
v1 = tkinter.IntVar(root, value = 1)
checkbtn1 = tkinter.Checkbutton(root, text='保存密码', variable=v1, command=onChe
ckButton)
checkbtn1.pack()

def onCheckButton2():
    print("自动登陆的值: ", v2.get())

v2 = tkinter.IntVar(root, value = 0)
checkbtn2 = tkinter.Checkbutton(root, text='自动登陆', variable=v2, command=onChe
ckButton2)
checkbtn2.pack()

root.mainloop()
```



6. 控件与变量的关联

控件与变量的关联?

是指通过一些特定参数，某些控件（如文本输入组件）的当前设置可直接与应用程序的变量关联。

控件的参数包括 variable 、 textvariable 、 onvalue 、 offvalue 、 value。这种关联是双向的：只要这些变量因任何原因发生变化，其关联的部件就会更新以反映新的参数值。

Tkinter 中已定义的关联变量

- StringVar
- IntVar

- DoubleVar
- BooleanVar

关联的参数有:

- variable
- textvariable
- onvalue
- offvalue
- value

关联变量方法:

方法	说明
variable.get()	获取相应的值
variable.set(value)	设置相应的值

关联变量构造函数的参数value:

- value 设置初始值

7. 滑块控件

作用

用于获取用户的数字输入(与input类似，但只能输入固定范围内的数值)

类名

```
Scale
```

Scale 控件属性

属性	说明	类型
fg	前景色	str
bg	背景色	str
orient	方向	HORIZONTAL(水平) , VERTICAL(竖直)
width	控件宽度(像素)	int
length	控件长度(像素)	int
borderwidth	边框宽度	int
from_	起始整数值	int
to	终止值(包含)	int
resolution	每次增/减的步长	int
command	步长变化的回调函数	function
variable	关联变量	IntVar或 DoubleVar
digits	显示的数字位数	int

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:

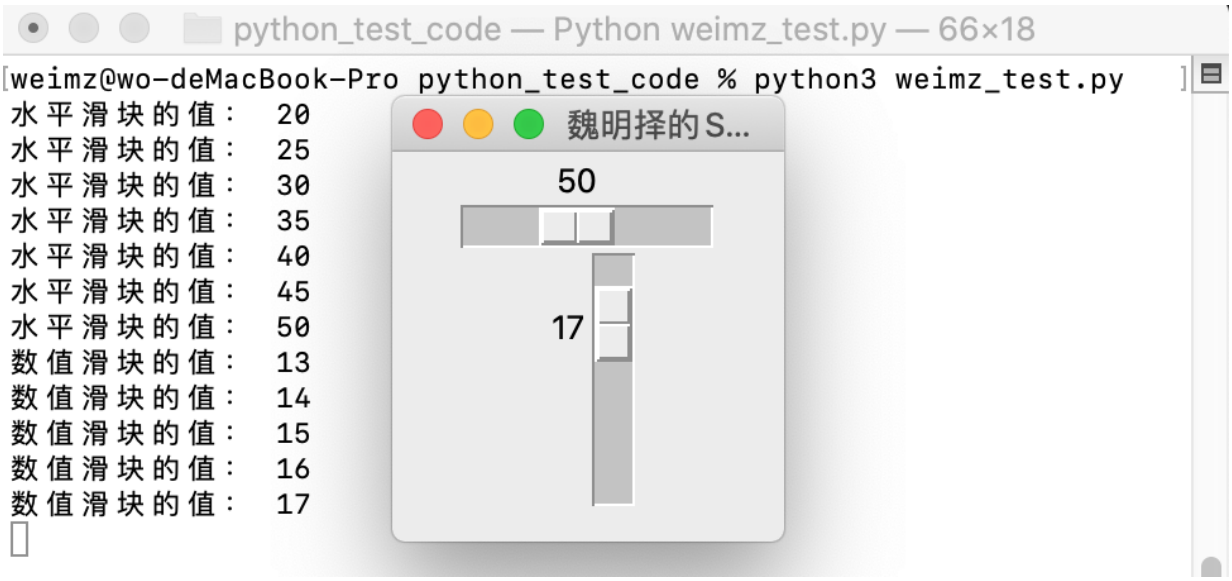
```
def 回调函数名(value):  # value绑定当前的值。
    pass
```

scale 控件示例

```
"""用 滑块 控件控制回调函数"""
def on_scale1_changed(value=None):
    print('水平滑块的值: ', value)

import tkinter
root = tkinter.Tk(className="魏明择的Scale示例")
```

```
# 创建一个Scale对象。
scale1 = tkinter.Scale(root, from_=10, to=100, resolution=5, orient=tkinter.HORI
ZONTAL, command=on_scale1_changed)
scale1.set(20)
scale1.pack()

def on_scale2_changed(value=None):
    print('数值滑块的值: ', value)

scale2 = tkinter.Scale(root, from_=12, to=40, orient=tkinter.VERTICAL, command=o
n_scale2_changed)
scale2.pack()
root.mainloop()
```

如图所示：



第三章、布局

1. 布局简介

什么是布局?

布局（Layout） 是指如何管理和排列控件（Widget）在窗口或容器（如 Frame）中的位置和大小。

Tkinter 提供了三种主要的布局管理器（Geometry Managers），每种方式适用于不同的场景。

三种布局：

- pack 打包布局 - 使用于简单界面，快速布局的场景
- grid 网格布局 - 适合复杂界面，精准控制行列的场景
- place 放置布局 - 需要固定位置或重叠控件的特殊场景。(已被弃用)

窗口坐标介绍

默认每个窗口内都是 三维的，对象坐标系中的x,y,z轴。

- X 轴水平向右。
- Y 轴坚直向下。
- Z 轴由内向外垂直屏幕。

如图：

> Z 轴垂直于屏幕向外。Z 轴值越大，离用户越近，Z 值大的控件会遮挡 Z 值小的控件。

2. 打包布局

什么是打包布局?

打包布局类似向行李箱里摆放积木块，有空间就放进去。默认上而下依次摆放。当然也可以设置参数来自下向上或自左向右摆放。

打包布局的方式比较简单。适合简单的布局场景。

布局方法

布局方法	说明
widget.pack()	按顺序自动排列控件（垂直或水平）

pack 常用属性：

属性	说明	类型
side	停靠窗口位置	str=tkinter.TOP、tkinter.BOTTOM、tkinter.LEFT、tkinter.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)

pack() 参数详解

以下用含有两个控件的窗口的布局来说明上述参数的含义，请注意观察 `widget.pack()` 内部参数的变化!

1. 上下两个控件，控件占用的空间就是内部文字占用的空间。默认自上而下排列，依次挨着。主窗口下方空余，两个控件都不会占用。

```
# 上下两个控件，widget.pack 无参数，都使用默认值!
import tkinter
root = tkinter.Tk(className="pack布局示例")

# 设置 label 的背景色为灰色，好区分边界。
label = tkinter.Label(root, text="欢迎来到weimingze.com!", bg="grey")
label.pack()
btn = tkinter.Button(root, text="点我退出!", command=root.destroy)
btn.pack()
root.mainloop()
```

效果如图

1. 上下两个控件，让Label控件的文字左右两侧内边距空余 20 个像素，内侧上下空余10个像素。使用ipadx和ipady来设置预留widget占用的空间。

```
import tkinter
root = tkinter.Tk(className="pack布局示例")

# 设置 label 的背景色为灰色，好区分边界。
label = tkinter.Label(root, text="欢迎来到weimingze.com!", bg="grey")
label.pack(ipadx=20, ipady=10)
btn = tkinter.Button(root, text="点我退出!", command=root.destroy, bg='red')
btn.pack()
root.mainloop()
```

效果如图

1. 上下两个控件，让Label控件的文字左右两侧外边距空余 50 个像素。上下预留 20个像素的空间。使用padx和pady来设置。

```
import tkinter
root = tkinter.Tk(className="pack布局示例")

# 设置 label 的背景色为灰色，好区分边界。
label = tkinter.Label(root, text="欢迎来到weimingze.com!", bg="grey")
label.pack(padx=50, pady=20)
```



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

效果如图

1. 上下两个控件，让Label控件填充主窗口x轴方向所有的空间。使用fill来设置。

```
import tkinter
root = tkinter.Tk(className="pack布局示例")

# 设置 label 的背景色为灰色，好区分边界。
label = tkinter.Label(root, text="欢迎来到weimingze.com!", bg="grey")
label.pack(fill=tkinter.X)
btn = tkinter.Button(root, text="点我退出!", command=root.destroy, bg='red')
btn.pack()
root.mainloop()
```

效果如图

1. 上下两个控件，让Label控件扩展到主窗口所有的剩余空间。将Button控件挤到下面，使用expand=1来设置。

```
import tkinter
root = tkinter.Tk(className="pack布局示例")
```



```
# 设置 label 的背景色为灰色，好区分边界。
label = tkinter.Label(root, text="欢迎来到weimingze.com!", bg="grey")
label.pack(expand=1)
btn = tkinter.Button(root, text="点我退出!", command=root.destroy, bg='red')
btn.pack()
root.mainloop()
```

> Label控件占据红框区域。 **效果如图**

1. 上下两个控件，让Label和Button控件同时扩展到主窗口所有的剩余空间。使用expand=1来设置。

```
import tkinter
root = tkinter.Tk(className="pack布局示例")

# 设置 label 的背景色为灰色，好区分边界。
label = tkinter.Label(root, text="欢迎来到weimingze.com!", bg="grey")
label.pack(expand=1)
btn = tkinter.Button(root, text="点我退出!", command=root.destroy, bg='red')
btn.pack(expand=1)
root.mainloop()
```

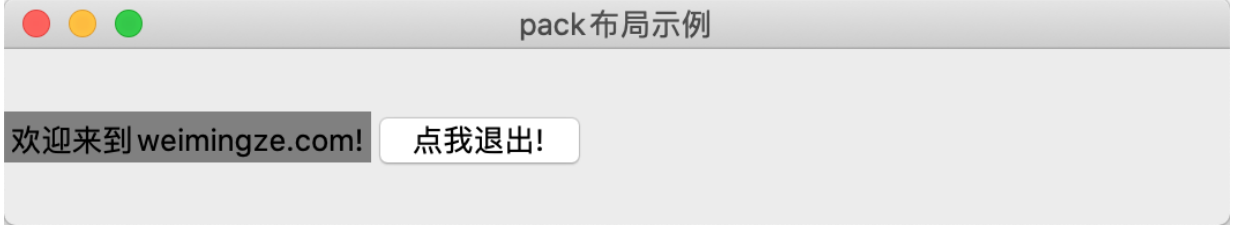
> 两个控件平分主窗口空间。

效果如图

1. 让Label和Button两个控件左右排布。使用side属性来设置。

```
import tkinter
root = tkinter.Tk(className="pack布局示例")

# 设置 label 的背景色为灰色，好区分边界。
label = tkinter.Label(root, text="欢迎来到weimingze.com!", bg="grey")
label.pack(side=tkinter.LEFT)
btn = tkinter.Button(root, text="点我退出!", command=root.destroy, bg='red')
btn.pack(side=tkinter.LEFT)
root.mainloop()
```

> 两个控件都向左依次排列。



1. 让Label和Button两个控件左右排布。使用side属性来设置。


```
import tkinter
root = tkinter.Tk(className="pack布局示例")

# 设置 label 的背景色为灰色，好区分边界。
label = tkinter.Label(root, text="欢迎来到weimingze.com!", bg="grey")
label.pack(side=tkinter.LEFT)
btn = tkinter.Button(root, text="点我退出!", command=root.destroy, bg='red')
btn.pack(side=tkinter.RIGHT)
root.mainloop()
```

> 两个控件Label 靠左，Button靠右 排列。

效果如图

动手来试试吧!

3. Pack布局之复杂排列

使用打包布局方式可以让控件实现非常复杂的排列方式，让界面更加美观。

下面列举了几个示例，共界面大师设计界面时使用。

含有三个控件的窗口的布局

左上下布局

效果和示例代码

```
#  +----+--------+
#  |    |        |
#  |    +--------+
#  |    |        |
#  +----+--------+
import tkinter
root = tkinter.Tk(className="魏明择的示例")
tkinter.Button(root, text="左").pack(side=tkinter.LEFT)
```

```
tkinter.Button(root, text="右上").pack(side=tkinter.TOP)
tkinter.Button(root, text="右下").pack(side=tkinter.BOTTOM)
root.mainloop()
```

右上下布局

效果和示例代码

```
#  +--------+----+
#  |        |    |
#  +--------+    |
#  |        |    |
#  +--------+----+
import tkinter
root = tkinter.Tk(className="魏明择的示例")
tkinter.Button(root, text="右").pack(side=tkinter.RIGHT)
tkinter.Button(root, text="左上").pack(side=tkinter.TOP)
tkinter.Button(root, text="左下").pack(side=tkinter.BOTTOM)
root.mainloop()
```

上左右布局

效果和示例代码

```
#  +-------------+
#  |             |
#  +------+------+
#  |      |      |
#  +------+------+
import tkinter
root = tkinter.Tk(className="魏明择的示例")
tkinter.Button(root, text="上").pack(side=tkinter.TOP)
tkinter.Button(root, text="左下").pack(side=tkinter.LEFT)
tkinter.Button(root, text="右下").pack(side=tkinter.RIGHT)
root.mainloop()
```

下左右布局

效果和示例代码

```
#  +------+------+
#  |      |      |
#  +------+------+
#  |             |
#  +-------------+
import tkinter
root = tkinter.Tk(className="魏明择的示例")
tkinter.Button(root, text="下").pack(side=tkinter.BOTTOM)
```

```
tkinter.Button(root, text="左上").pack(side=tkinter.LEFT)
tkinter.Button(root, text="右上").pack(side=tkinter.RIGHT)
root.mainloop()
```

上、中左中右、下布局

效果和示例代码

```
#  +-------------+
#  |             |
#  +------+------+
#  |      |      |
#  +------+------+
#  |             |
#  +-------------+
import tkinter  # 导入tkinter包
root = tkinter.Tk(className="魏明择的示例")
tkinter.Button(root, text="上").pack(side=tkinter.TOP)
tkinter.Button(root, text="下").pack(side=tkinter.BOTTOM)
tkinter.Button(root, text="左上").pack(side=tkinter.LEFT)
tkinter.Button(root, text="右上").pack(side=tkinter.RIGHT)
root.mainloop()
```

4. 网格布局

什么是网格布局?

网格布局是最常用的布局管理器之一，它允许你以行（row）和列（column）的方式排列控件，类似于 HTML 表格或 Excel 单元格。grid 提供了灵活的控制方式，适用于复杂的 GUI 设计。

网格布局的概念

1. 网格（Grid）：将窗口或容器（如 Frame）划分为若干行和列，形成单元格。
2. 控件放置：每个控件可以占据一个或多个单元格。
3. 对齐方式：可以控制控件在单元格内的对齐方式（如靠左、靠右、居中、拉伸等）。

布局方法

布局方法	说明
widget.grid()	指定控件的起始位置可占用的宽度和高度）

grid 常用属性：

属性	说明	示例和说明
row	控件所在的行（从 0 开始）	row=0（第一行）
column	控件所在的列（从 0 开始）	column=2（第三列）
padx	横向外边距离（同pack布局）	int类型
pady	纵向外边距离（同pack布局）	int类型
ipadx	横向内边距离（同pack布局）	int类型
ipady	纵向内边距离（同pack布局）	int类型
rowspan	跨越行数控件跨越的行数（合并多行）	rowspan=2（占据 2 行）
columnspan	控件跨越的列数（合并多列）	columnspan=3（占据 3 列）
sticky	控件在单元格内的对齐方式（N, S, E, W 或组合，如 N+W）	sticky="nsew"（上下左右拉伸）

grid 网格布局示例1

```
import tkinter
root = tkinter.Tk(className="魏明择的示例")

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()
```

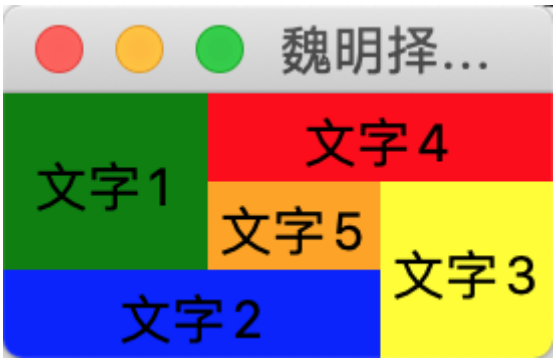
> 以上是3行3列的网格布局，按钮可以放在不同的**单元格**中。

效果如图

grid 网格布局示例2

```
import tkinter
root = tkinter.Tk(className="魏明择的Entry示例")

label1 = tkinter.Label(root, text="文字1", bg="green")
label1.grid(row=0, column=0, rowspan=2, ipadx=5, ipady=5, sticky='news')

label2 = tkinter.Label(root, text="文字2", bg='blue')
label2.grid(row=2, column=0, columnspan=2, sticky='news')

label3 = tkinter.Label(root, text="文字3", bg='yellow')
label3.grid(row=1, column=2, rowspan=2, sticky='news')

label4 = tkinter.Label(root, text="文字4", bg='red')
label4.grid(row=0, column=1, columnspan=2, sticky='news')

label4 = tkinter.Label(root, text="文字5", bg='orange')
label4.grid(row=1, column=1, sticky='news')
```

效果如图



> 以上是3行3列的网格布局，Label可以跨越2个或以上的单元格，最终实现了上述复杂的效果。

5. 框架控件 Frame

什么是框架控件?

框架控件（Frame） 是 Tkinter 中的一个容器控件，用于组织和分组其他控件（如按钮、标签、输入框等）。它类似于 主窗口，内部可以有自己的布局和控件放置方式，一个主窗口可以有多个 Frame，这样可以实现更复杂的布局方式。

作用

1. 结构化布局：将相关控件放在同一个 Frame 中，便于管理。
2. 复杂界面设计：嵌套多个 Frame 实现更灵活的布局。
3. 样式控制：统一设置背景色、边框等。

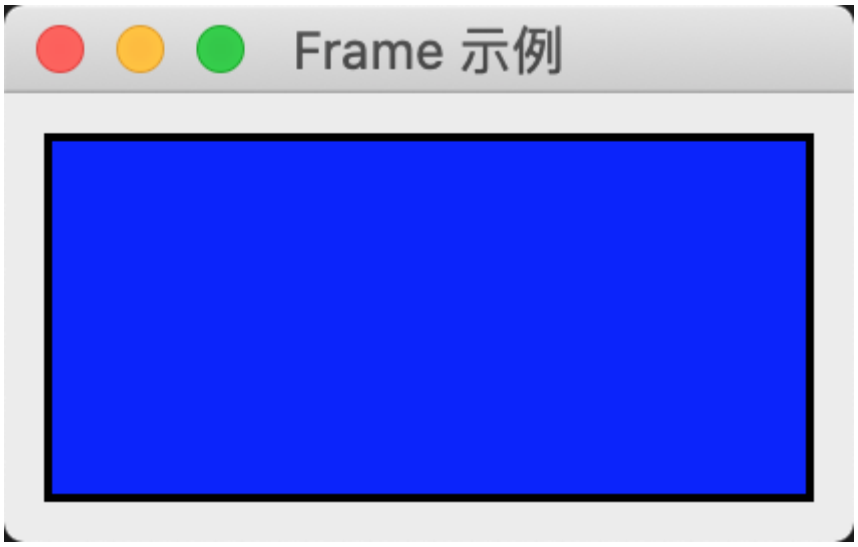
Frame 控件常用属性：

属性	说明	类型
bg	背景色	str
height	高	int
width	宽	int
bd或 borderwidth	边框宽度（默认 0，无边框）	int
relief	边框样式 （"flat"、"raised"、"sunken"、"solid"、"ridge"、"groove"）	str

示例

```
import tkinter

root = tkinter.Tk()
root.title("Frame 示例")

# 创建一个 Frame，并放置到 root 窗口
frame = tkinter.Frame(root, bg="blue", bd=2, relief="solid")
frame.pack(padx=10, pady=10, fill=tkinter.BOTH, expand=1)  # 使用 pack 布局

root.mainloop()
```

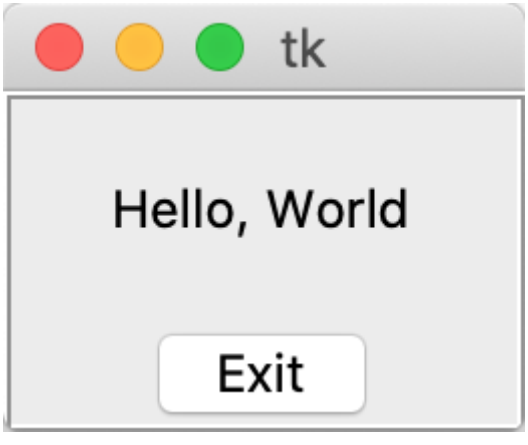
效果如图



示例2

```
import tkinter
from tkinter.constants import *
root = tkinter.Tk()
frame = tkinter.Frame(root, 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=root.destroy)
button.pack(side=BOTTOM)
root.mainloop()
```

效果如图

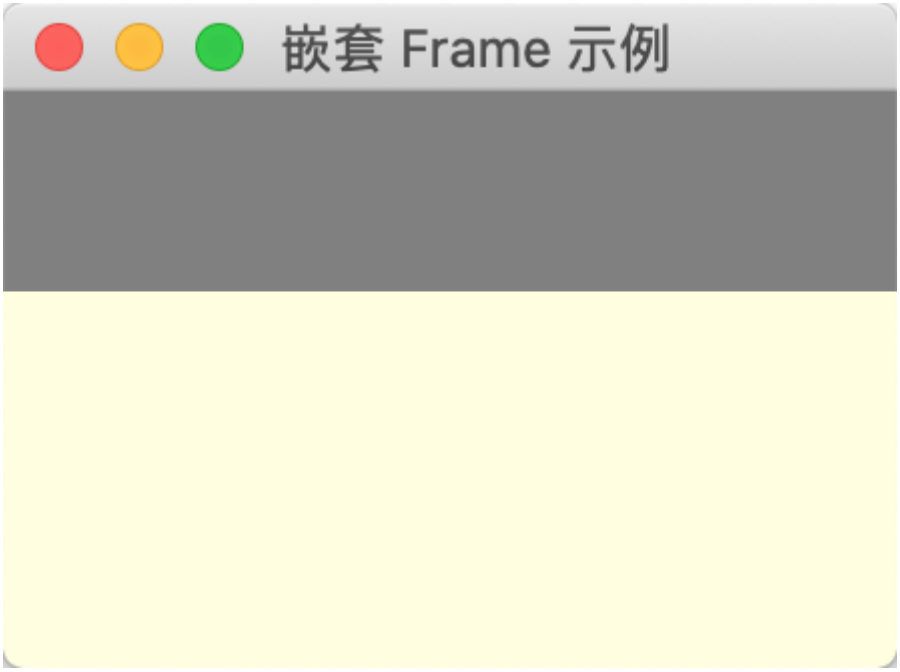


示例3

```
import tkinter

root = tkinter.Tk()
```

```
root.title("嵌套 Frame 示例")

# 外层 Frame（顶部导航栏）
top_frame = tkinter.Frame(root, bg="gray", height=50)
top_frame.pack(fill=tkinter.X)  # 水平填充

# 内层 Frame（内容区域）
content_frame = tkinter.Frame(root, bg="lightyellow")
content_frame.pack(fill=tkinter.BOTH, expand=True)

# 在 content_frame 中添加控件
label = tkinter.Label(content_frame, text="主要内容", bg="lightyellow")
label.pack(pady=50)

root.mainloop()
```

效果如图



第四章、画布

1. 画布 canvas

画布（Canvas）是 Tkinter 中一个功能强大的绘图控件，它提供了一个矩形区域，允许你在上面绘制图形、文字、图像，甚至放置其他控件。

画布支持以下功能：

- 绘制基本图形（线条、矩形、圆形、多边形等）
- 显示文字和图像
- 创建交互式绘图工具（如画板、图表）
- 构建游戏界面（如贪吃蛇、俄罗斯方块）

创建画布的构造方法:

```
tkinter.Canvas(master=None, width=0, height=0)
```

常用参数

- width / height：画布的宽度和高度（像素）。
- bg：背景颜色（如 "white"、"lightblue"）。
- bd：边框宽度（默认 0）。
- relief：边框样式（如 "solid"、"raised"）。

示例：

```
import tkinter

root = tkinter.Tk()
canvas = tkinter.Canvas(root, width=400, height=300, bg="white")
canvas.pack()

root.mainloop()
```

画布支持的图形如下:

- bitmap
- image(BitmapImage, PhotoImage)
- line

- oval
- polygon
- rectangle
- text
- arc

画布创建图形的方法:

```
canvas.create_xxx(x, y, other, ...)
```

如：create_line()、create_rectangle()、create_oval()、create_polygon()等。

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

例如

绘制线条

```
# 从 (x1, y1) 到 (x2, y2) 画一条线
line = canvas.create_line(50, 50, 200, 50, fill="red", width=2)
```

参数

- fill：线条颜色。
- width：线条宽度（像素）。

绘制矩形

```
# 绘制矩形（左上角坐标 x1,y1，右下角坐标 x2,y2）
rect = canvas.create_rectangle(100, 100, 250, 200, fill="blue", outline="black")
```

参数:

- fill：填充颜色。
- outline：边框颜色。

绘制圆形/椭圆

```
# 绘制椭圆（边界矩形的左上角和右下角坐标）
oval = canvas.create_oval(50, 50, 150, 100, fill="green")
```

绘制多边形

```
# 按顺序连接多个点 (x1,y1, x2,y2, ...)
polygon = canvas.create_polygon(200, 50, 250, 100, 150, 100, fill="yellow")
```

显示文字

```
# 在 (x,y) 处显示文字
text = canvas.create_text(200, 150, text="Hello Canvas!", font=("Arial", 14), fi
ll="black")
```

显示图片

```
from tkinter import PhotoImage

# 加载图片（仅支持 GIF 或 PGM/PPM）
img = PhotoImage(file="example.gif")
image_item = canvas.create_image(100, 100, image=img, anchor="nw")  # anchor 指定
对齐方式
```

图形对象的操作

移动图形:实现一个图形的移动动作

```
canvas.move(item_id, dx, dy)  # 沿 x 和 y 轴移动
```

修改属性

```
canvas.itemconfig(item_id, fill="new_color")  # 更改颜色
```

删除图形

```
canvas.delete(item_id)  # 删除单个图形
canvas.delete("all")    # 清空整个画布
```

获取图形坐标

```
coords = canvas.coords(item_id)  # 返回 [x1, y1, x2, y2, ...]
```

事件绑定（交互式绘图）

Canvas 支持鼠标和键盘事件，可用于构建交互式应用：

示例

```
def on_click(event):
    x, y = event.x, event.y
    canvas.create_oval(x-5, y-5, x+5, y+5, fill="red")  # 在点击处画圆

canvas.bind("<Button-1>", on_click)  # 绑定鼠标左键点击事件
```

常见事件：

- `<Button-1>`：鼠标左键点击。
- `<B1-Motion>`：按住左键拖动。
- `<ButtonRelease-1>`：左键释放。
- `<KeyPress>`：键盘按键。

示例

简单画板程序

```
import tkinter as tk

def start_draw(event):
    global last_x, last_y
    last_x, last_y = event.x, event.y

def draw(event):
    global last_x, last_y
    canvas.create_line(last_x, last_y, event.x, event.y, width=2, fill="black")
    last_x, last_y = event.x, event.y

root = tk.Tk()
canvas = tk.Canvas(root, width=500, height=400, bg="white")
canvas.pack()

canvas.bind("<Button-1>", start_draw)
canvas.bind("<B1-Motion>", draw)

root.mainloop()
```

示例

显示动态数据（如折线图）

```
import tkinter as tk

root = tk.Tk()
canvas = tk.Canvas(root, width=400, height=300, bg="white")
canvas.pack()

# 模拟数据点
data = [50, 80, 120, 90, 180]
```

```
for i in range(len(data)-1):
    canvas.create_line(i*80, 300-data[i], (i+1)*80, 300-data[i+1], fill="blue",
width=2)

root.mainloop()
```

高级功能：

- 显示图片（需配合 PIL 显示 PNG/JPG）。
- 事件绑定实现交互（如画板、游戏）。

第五章、事件

1. 事件 event

什么是事件?

事件是用户在操作图形用户界面时发生的动作。如鼠标，键盘等对窗口元素操作，输入框焦点进入、离开等操作。

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格式:

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

> 注: event为事件类型

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

```
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()
```

第六章、消息对话框

1. 消息对话框

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

示例

```
import tkinter
from tkinter import messagebox
root = tkinter.Tk(className="魏明择的Entry示例")
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

实现如下类似于 Foxmail的邮箱编辑邮件的用户界面:

```
+------+------+
| send | save |
+------+------+--------------------+
| recive | [studio@weimingze.com;] |
| title  | [welcome to xi'an!]     |
+---------------------------------+
| Mr. Wei                         |
|     thank!                      |
| weimingze                       |
| 2018-3-25                       |
+---------------------------------+
```

第六章、项目

1. 2048游戏项目

2048 游戏是一个经典的游戏项目。本游戏是根据官方游戏 2048 改编。

2048 的官方网址:

- 网址：https://2048game.com/

建议同学们先去玩一下再学习本节课内容。

此项目就是使用 python 语言，结合 tkinter 图形用户界面重写这个游戏。为了方便理解。这个游戏要比实际游戏简化的很多，代码也有注释，易于理解。

游戏效果如下图所示：

游戏的下载地址:

- 码云地址：https://gitee.com/weimz/py2048

选择"下载ZIP"就可以下载源码的压缩包。

或者使用 git 命令克隆镜像，下载代码请复制以下命令到终端执行：

```
git clone https://gitee.com/weimz/py2048.git
```

运行方法

```
% python3 2048game.py
# 或
% ./2048game.py
```

如有不懂的问题请在公众号：**明择编程** 进行反馈！

祝学习愉快！

2. 飞机大战项目

飞机大战游戏 是根据经典的街机游戏改编。

此游戏用户可以通过上下左右方向键或者鼠标来控制飞机的飞行。飞机定时不间断的发送子弹。

敌机有大有小，碰到飞机则丢失一条命，共计三条命，全部丢失游戏结束。

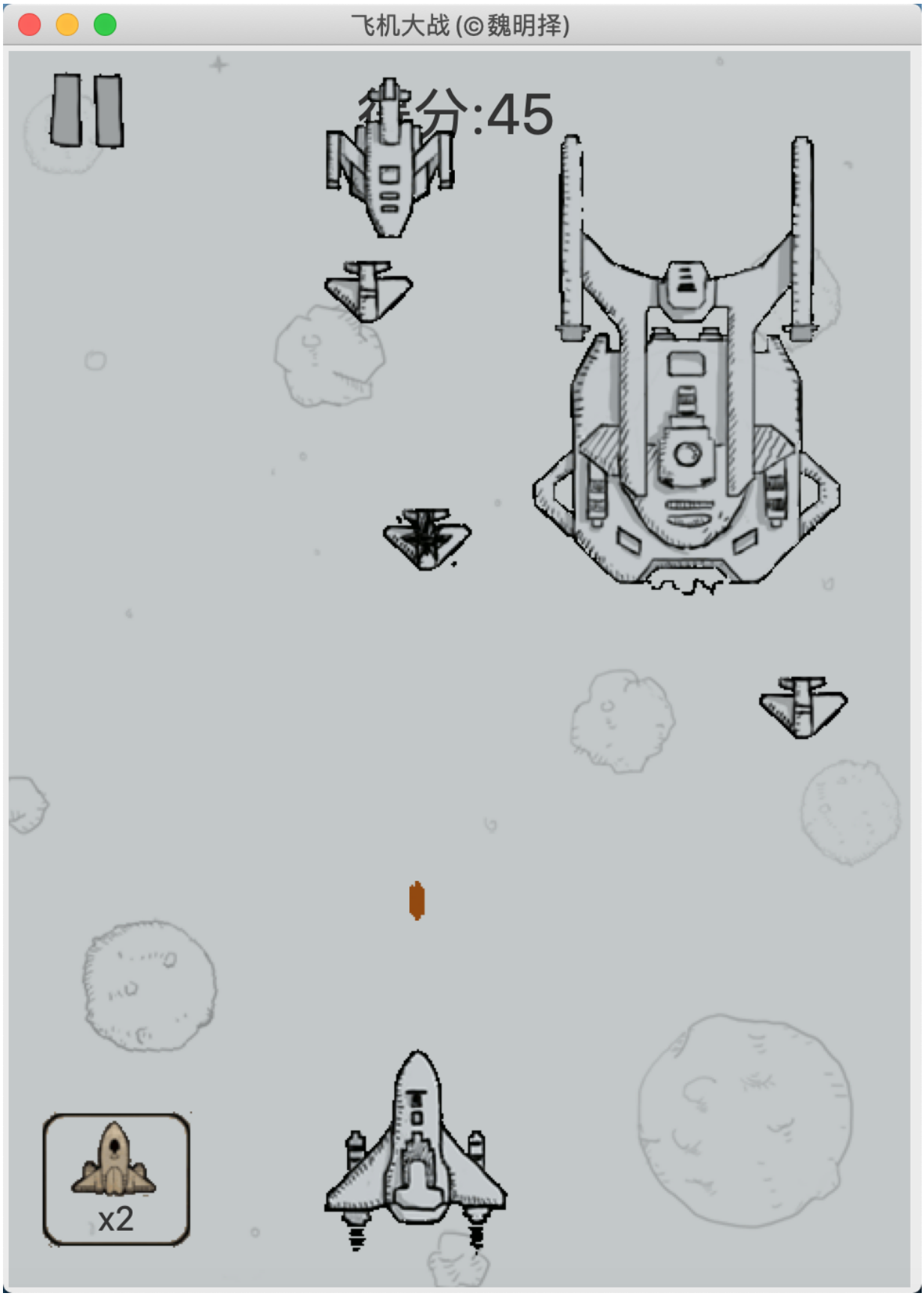
敌机会随时间增加越来越多。一般10分钟左右游戏就会结束。（魏老师就这样设计的，防止沉迷，呵呵！）。

建议同学们先去玩一下再查看源代码学习。

此项目就是使用 python 语言，结合 tkinter 的 画布（Canvas）实现。图片和源码都在其中。供爱好者使用。

此项目使用了面向对象的编程思想，用到了封装，继承和多态。希望 Python 爱好者能够看懂代码并能够独自写出类似游戏。

游戏效果如下图所示：



游戏的下载地址:

- 码云地址：https://gitee.com/weimz/plane_war

选择"下载ZIP"就可以下载源码的压缩包。

或者使用 git 命令克隆镜像，下载代码请复制以下命令到终端执行：

```
git clone https://gitee.com/weimz/plane_war.git
```

运行方法

```
$ python3 main.py
# 或
$ ./main.py
```

如有不懂的问题请在公众号：**明择编程** 进行反馈！

祝学习愉快！