第 1 章

GUI 编程

1.1 GUI 简介

在学习《Python 全栈开发——基础入门》一书中的相关示例代码时，其输入和输出都是在 Python 编辑器或 IDE 工具中实现的，但是在现实的项目中，程序却经常以 Web 系统（该内容将在《Python 全栈开发——Web 编程》一书中为读者做详细介绍）的形式，或 GUI 的形式展现在用户面前。

GUI(Graphical User Interface，图形用户界面，又称图形用户接口）是指采用图形方式显示的计算机操作用户界面。

图形用户界面与通过键盘输入文本或字符命令来完成例行任务的字符界面相比，有着不可比拟的优势。由于图形用户界面是一种人与计算机通信的界面显示格式，所以其允许用户使用鼠标、键盘等输入设备操纵屏幕上的图标或菜单选项，以执行选择命令、调用文件、启动程序或其他一些日常任务。

此外，图形用户界面由窗口、下拉菜单、对话框及其相应的控制机制构成，在各种新式应用程序中都是标准化的，即相同的操作总是以同样的方式来完成，并且在图形用户界面中，用户所看到的和所操作的都是图形对象，应用的是计算机图形学的技术。

1.1.1 GUI 的特点

1. 人机交互性

图形用户界面的主要功能是实现人与计算机等电子设备的人机交互。它是用户与操作系统之间进行数据传递和互动操控的工具，用户可以通过一定的操作实现对电子设备的控制，同时电子设备将用户操作的结果通过屏幕进行反馈。作为使用电子信息产品的必备环节，图形用户界面实现了人与软件之间的信息交互。这种人机交互性使用户的操作更加便捷。

2. 美观性

对于日新月异的电子产品来讲，图形用户界面发挥着越来越重要的作用。美观、友好的图形用户界面设计往往更能吸引用户，成为企业获得竞争优势的关键。图形用户界面综合了人机工程学、认知心理学、设计艺术学、语言学、社会学、传播学等众多学科领域的知识，现在已经发展成为一门独立的学科。在电子技术飞速发展的今天，电子产品的性能和功能的区别已经不是很大，致使开发者更加注重产品的美观性。图形用户界面是多种元素的组合，

包含众多艺术性和美观性的设计元素，大气的外观、简约的设计风格、良好的视觉效果日益成为影响用户体验的关键因素，而正是这种美好的视觉感受促使用户购买相应的产品，进而可以提高企业的经济效益。

3. 实用性

图形用户界面的目的是实现人机交互。开发人员研究并设计出具体的用户界面，将晦涩难懂的计算机语言包装成简单易懂的图形，用户通过对图形的识别，进而理解复杂的计算机语言其背后所表达的内容。图形化的操作方式具有很强的实用性，方便了用户的使用，提高了使用效率。这种创造性的转化使冷冰冰的电子产品变得更加亲切，进而从实验室走进千家万户的生活之中。现如今，开发人员通过对图形用户界面的进一步优化，使信息、数据的传输更高效，使运行与反馈结果更准确，这势必会带来更好的用户体验，极大地增加了图形用户界面的实用性。

4. 技术性

早期电子产品的图形用户界面采用字符界面，需要操作人员具有较高的专业性，文字转换为图形后，相应的信息、数据也被转化为图形。用户操作、接收的信息都是图形对象，不再需要记忆大量的命令符号，并且无须具备专业知识和操作技能即可实现对电子产品的操作，但简化了的操作过程并不意味着图形用户界面不具有技术性，与之相反，其隐藏在图形对象背后的是更加专业的代码编写等相关技术性操作。技术人员通过编写和设计相关的代码，将字符界面转换为图形界面，以便用户可以利用图形界面实现他们想要操作的内容。这样的转换方式往往需要较高的技术性，所以图形用户界面只是将技术性命令符号隐藏起来，而并非是不具有技术性。

1.1.2 GUI 的设计原则

1. 风格的一致性原则

图形用户界面的一致性主要是指呈现给用户的通用操作序列、术语和信息的措辞，以及界面元素的布局、颜色搭配方案和排版样式等都要保持一致。具有高度一致的图形用户界面可以让各部分的信息安排得井然有序，给用户以清晰感和整体感，有利于用户对图形用户界面运作建立起精确的心理模型，从而降低培训和支持成本。

除特殊情况外，图形用户界面的设计风格都应保持高度的一致性，一致性是界面设计是否成功的重要因素之一，而保证一致性的一个有效方法就是撰写正式的“设计风格标准”文件。这一文件规定在一个产品或系列产品的图形用户界面设计中都必须遵守的设计准则。“设计风格标准”文件规定的设计准则应当非常具体，包括所使用的图标、尺寸、字体等内容和格式的例子。该文件可以有效地用于图形用户界面的管理和调整，是设计大型、复杂图形用户界面或多人多部门共同协作的设计工作所必不可少的。

2. 布局的逻辑性原则

图形用户界面布局的逻辑性一般情况包含以下几点：第一，图形用户界面的布局应当体现用户操作时的一般顺序和被使用到的频繁程度，例如应当符合人们通常阅读和填写纸质表单的顺序，即人们的阅读顺序是从左至右、由上而下；第二，用户经常使用的图形用户界面元素应当放在突出的位置，让用户可以轻松地注意到它们。相反，一些不常用的元素可以放在不显眼的位置，甚至允许用户可以将它们隐藏起来，以便扩大屏幕的可用区域；第

三，对于那些需要具备一定条件才可以使用的元素，应当把它们显示成灰色状态，当具备了使用条件时才改变成正常状态；第四，特定的元素应放置在它所要控制数据的邻近位置，帮助用户确立元素和数据之间的关系。此外，影响整个对话框的元素应当与那些控制特定数据的元素区分开来，关系紧密的元素应有组织地放置在同一个区域。

3. 图形元素的启示性原则

启示性是唐纳德·诺曼在研究日常物品的设计时提出的术语，定义为事物被感觉到的特性和实际特性，主要确定事物可能的使用方式的基本特征，也就是说启示性指的是物品的某个属性，而这个属性可以让使用者知道如何使用这个物品。例如不同形状的门把手分别暗示着“推”“拉”或“旋转”。图形用户界面中的图形元素（如按钮、图标、滚动条、窗口和链接等）同样可以暗示它们所代表的功能，或启发用户如何使用它们。

而在众多元素中，图标则是图形用户界面中最重要的元素之一，例如把窗口缩小成一个图标，可以用来表示暂时不需要执行的一个对话过程，用户可以随时单击它重新执行该对话；图标也可以用来表示用户可以访问的程序和功能，例如回收站、磁盘等图标；图标还可以用于数据存储形式和组织形式，例如各种类型的文件图标和文件夹图标。

此外，由于技术的限制，最初出现在图形用户界面中的图标，大多数是单色的几何型符号，并且尺寸都比较小，但随着显示器分辨率的增大，越来越多的图标采用写实的设计风格，不再局限于简单的几何型符号，而设计代表系统功能或对象操作方式的图标会给设计师带来一些有趣的挑战，其中最重要的一个挑战就是用图标的视觉语言代表抽象的概念，这就要求图标设计要保持统一的视觉风格，同时也要注意使每个图标具有鲜明的个性。

综上所述，图形元素不仅是让图形用户界面具有视觉艺术性，更重要的是图形元素要具有一定的启示性，以帮助用户理解界面。

4. 界面的习惯性用法原则

习惯性用法是基于用户学习和使用习惯的方式。遵循习惯性用法的界面应当不关注技术知识或人的直觉功能，也不会引发人的联想。图形用户界面容易使用的主要原因是限定了一系列用户和系统进行交互的词汇，即由指向、单击和拖动等不可分割的动作和反馈机制形成基本的使用词汇，再使用基本的使用词汇构成一系列的组合词汇，进而形成更为复杂的组合用法，例如双击、单击并拖动等操作方法，以及按钮、复选框等操作对象。

1.2 GUI的开发工具包

Python中有许多优秀的图形用户界面开发工具包，包括PyQt、PyGTK、Kivy、Flexx、pyui4win、Tkinter和wxPython等。

1. PyQt

PyQt是Qt与Python的成功融合，或者也可以认为PyQt是Qt库的Python版本。PyQt结合了二者的优点，可以用于快速地创建应用程序，并且PyQt还可以进行跨平台开发。PyQt包括PyQt 3、PyQt 4、PyQt 5和PyQt 6等，PyQt 5之前的版本均不再支持更新，所以对于新开发的应用程序，强烈推荐使用PyQt 5或PyQt 6。

2. PyGTK

PyGTK是使用Python封装的GTK图形库，通过PyGTK可以轻松创建具有图形用

户界面的程序。PyGTK 真正具有跨平台性，它能不加任何修改，稳定地运行在各种操作系统之上，如 Linux、Windows 和 macOS 等。除了简单易用和快速的原型开发能力之外，PyGTK 还拥有一流的处理本地化语言的独特功能。

3. Kivy

Kivy 是一个开源工具包，能够让使用相同源代码创建的程序跨平台运行，如图 1-1 所示。Kivy 主要关注创新型的图形用户界面开发，例如多点触摸应用程序等。Kivy 还支持 GPU 加速，拥有 Flash 般的动画效果，开发者只需简单的几行代码便可以写出炫丽的界面。除此之外，Kivy 还具有良好的 API 文档，便于初学者快速入门学习。

图 1-1 Kivy

4. Flexx

Flexx 是一个纯 Python 工具包，用来创建图形化界面应用程序，其使用 Web 技术进行界面的渲染，并且由于 Flexx 是使用 Python 开发的，所以 Flexx 同样具有跨平台性。

5. pyui4win

pyui4win 是一个采用自绘技术的开源界面库，支持 C++ 和 Python。由于 pyui4win 拥有所见即所得的界面设计器，所以在 pyui4win 中，界面设计甚至可以完全交由美工人员去处理，而开发人员只需负责处理业务逻辑，彻底地将开发人员从繁杂的界面设计工作中解放出来。

6. Tkinter

Tkinter 是 Python 官方提供的图形用户界面开发工具包，基于 Tk GUI 工具包封装而来。Tkinter 是一个轻量级的跨平台图形用户界面开发工具包，可以在 UNIX、Linux、Windows 和 macOS 中运行，并且在 Tkinter 8.0 之后可以实现本地窗口风格。Tkinter 用起来非常简单，并且开发速度也较快，Python 自带的 IDLE 就是使用 Tkinter 编写的，但是 Tkinter 所包含的控件较少，在开发复杂的图形用户界面时，会显得力不从心。

7. wxPython

wxPython 是一款开源的 GUI 图形库，其基于 wxWidgets 工具包封装而来，允许 Python 程序员很方便地创建完整的、功能健全的 GUI 用户界面，并且 wxPython 同样具有非常优秀的跨平台能力，如图 1-2 所示。除此之外，wxPython 提供了丰富的控件，可以开发复杂的图形用户界面，而且 wxPython 的帮助文档非常完善，易于初学者快速入门学习。

通过对上述图形用户界面开发工具包的初步介绍，可以得知每个图形用户界面开发工具包都具有其鲜明的优缺点，所以在项目开发前，读者需要根据项目的具体应用场景来选择使用更为合适的图形用户界面开发工具包进行开发。

本书将为读者重点讲解 Tkinter 和 wxPython 的使用方式。

图 1-2　wxPython

第 2 章

Tkinter

2.1 Tkinter 的安装

由于 Tkinter 是 Python 的标准 GUI 库，所以不需要进行额外的安装，而仅需要在使用前引入 tkinter 包即可进行编程，示例代码如下：

```
#资源包\Code\chapter2\2.1\0201.py
import tkinter
```

2.2 Misc 类和 Wm 类

Misc 类和 Wm 类是 Tkinter 中的两大基类，其中，Misc 类是所有控件的根父类，而 Wm 类则提供了一些与窗口管理器相关的功能函数。

图 2-1 中列出了 Tkinter 中类的继承关系。

此外，对于 Misc 类和 Wm 类这两大基类而言，在 GUI 编程的过程中并不会直接使用它们，而是使用它们的子类，并且由于它们是所有 GUI 控件的父类，因此 GUI 中的控件都可以直接使用这两大基类的方法，其常用的方法如下。

1）after()方法

该方法用于按照指定的时间间隔重复执行指定的函数，其语法格式如下：

```
after(ms, func)
```

其中，参数 ms 表示时间间隔，单位为毫秒；参数 func 表示待执行的函数。

2）winfo_x()方法

该方法用于获取当前窗口左上角相对于主屏幕左上角的 x 轴坐标，其语法格式如下：

```
winfo_x()
```

3）winfo_y()方法

该方法用于获取当前窗口左上角相对于主屏幕左上角的 y 轴坐标，其语法格式如下：

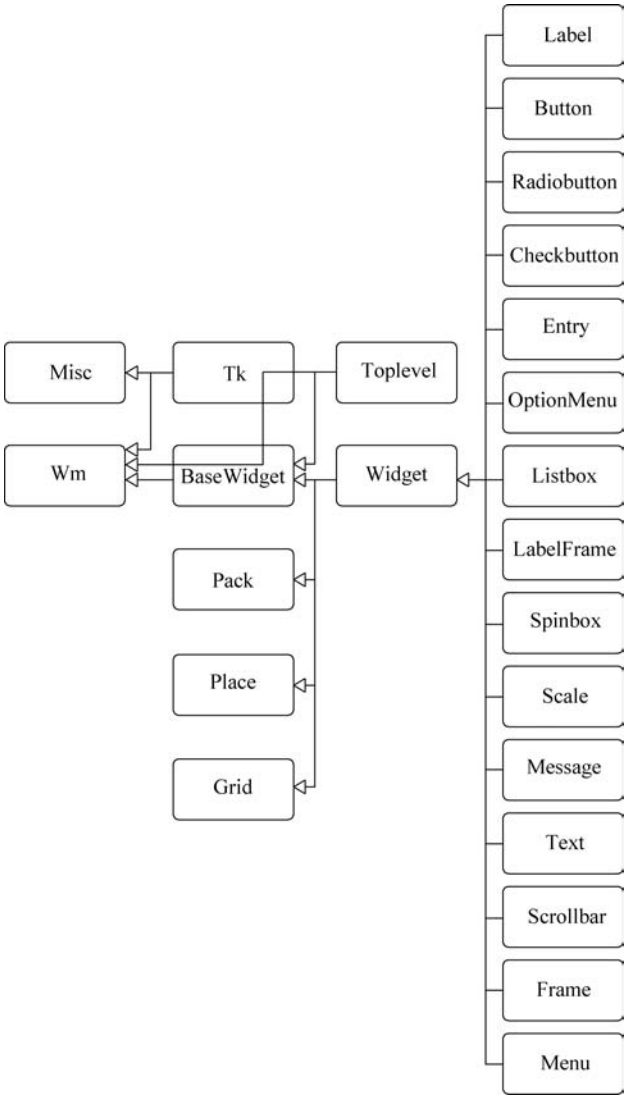

图 2-1 Tkinter 中类的继承关系

```
winfo_y()
```

4) config()方法

该方法用于配置控件中的参数,其语法格式如下:

```
config(options)
```

其中,参数 options 表示控件中的参数。

2.3 主窗口

主窗口是一个容器元素,可以在其上添加控件,并呈现给用户。

1. 创建主窗口对象

可以通过 Tkinter 模块中的 Tk 类创建主窗口对象，用于完成主窗口的创建，其语法格式如下：

```
Tk()
```

2. 主窗口对象的相关方法

1）title()方法

该方法用于设置主窗口的标题，其语法格式如下：

```
title(string)
```

其中，参数 string 表示主窗口的标题。

2）iconbitmap()方法

该方法用于设置和获取主窗口的图标，其语法格式如下：

```
iconbitmap(bitmap)
```

其中，参数 bitmap 表示主窗口的图标。

3）geometry()方法

该方法用于调节主窗口的尺寸和位置，其语法格式如下：

```
geometry(newGeometry)
```

其中，参数 newGeometry 表示主窗口尺寸和位置的特定格式，该格式为 widthxheight $\pm x \pm y$，width 和 height 表示主窗口的宽和高，x 和 y 表示主窗口左上角的 x 轴坐标和 y 轴坐标。

4）resizable()方法

该方法用于设置主窗口能否最大化，其语法格式如下：

```
resizable(width, height)
```

其中，参数 width 表示主窗口横向能否最大化；参数 height 表示主窗口纵向能否最大化。

5）maxsize()方法

该方法用于设置和获取主窗口的最大尺寸，其语法格式如下：

```
maxsize(width, height)
```

其中，参数 width 表示主窗口的宽度；参数 height 表示主窗口的高度。

6）protocol()方法

该方法用于将回调函数与相应的规则进行绑定，其语法格式如下：

```
protocol(name, func)
```

其中，参数 name 表示规则，包括 WM_DELETE_WINDOW（窗口被关闭时）、WM_SAVE_YOURSELF（窗口被保存时）和 WM_TAKE_FOCUS（窗口获得焦点时）；参数 func 表示回调函数。

7）mainloop()方法

该方法用于主事件循环，其语法格式如下：

```
mainloop()
```

3. 创建主窗口

创建主窗口有两种方式，分别为使用 Tk 类和 Tk 类的子类。

1）使用 Tk 类创建主窗口

示例代码如下：

```
#资源包\Code\chapter2\2.3\0202.py
import tkinter as tk
#创建窗口
root = tk.Tk()
#设置窗口标题
root.title('第 1 个 Tkinter 程序')
#设置窗口的尺寸和位置
root.geometry('500x400+20+20')
#设置窗口横向不能最大化
root.resizable(width=False, height=True)
#主事件循环
root.mainloop()
```

上面代码的运行结果如图 2-2 所示。

图 2-2 通过 Tk 类创建主窗口

2）使用 Tk 类的子类创建主窗口

示例代码如下：

```
#资源包\Code\chapter2\2.3\0203.py
import tkinter as tk
class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.set_window()
    def set_window(self):
        self.title("第 1 个 Tkinter 程序")
        self.geometry('500x400+20+20')
        self.resizable(False, True)
if __name__ == "__main__":
    app = App()
    app.mainloop()
```

上面代码的运行结果如图 2-3 所示。

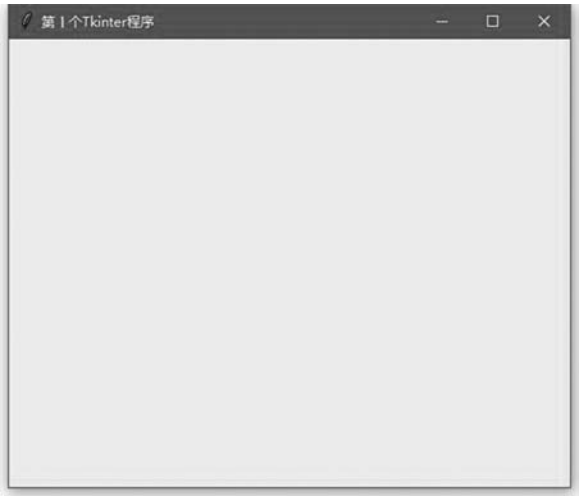

图 2-3　使用 Tk 类的子类创建主窗口

2.4　控件

在 Tkinter 中，通过 Widget 类的子类来创建各种控件，其包括标签（Label 类）、按钮（Button 类）、单选按钮（Radiobutton 类）、多选按钮（Checkbutton 类）、文本输入框（Entry 类）、下拉菜单（OptionMenu 类）、列表框（Listbox 类）、静态框（LabelFrame 类）、微调节器（Spinbox 类）、滑块（Scale 类）、消息（Message 类）、文本（Text 类）、滚动条（Scrollbar 类）、框架（Frame 类）、顶级窗口（Toplevel 类）和菜单栏（Menu 类）等。

2.4.1 跟踪控件的值

Tkinter 支持部分控件与变量进行双向绑定，即当程序更改变量值时，其对应的控件所显示的文本内容或控件的其他参数的值也会随之改变；反之，当控件所显示的文本内容或控件的其他参数的值发生改变时，其对应的变量值同样会随之改变。

而实现这种双向绑定非常简单，只需将变量传递给控件的参数 textvariable、参数 listvariable 或参数 variable，其中，参数 textvariabl 和参数 listvariable 主要与控件的文本内容相关，而参数 variable 主要与控件的其他参数的值相关。

除此之外，双向绑定对变量的类型有着严格的要求，即该变量不能是普通类型的变量，其类型仅可以是 Tkinter 模块中 Variable 类的子类，包括 IntVar 类（整数类型的变量）、DoubleVar 类（浮点数类型的变量）、StringVar 类（字符串类型的变量）和 BooleanVar 类（布尔值型的变量）。

此外，Variable 类具有两个常用的方法，即 get()方法和 set()方法，分别用于获取变量的值和设置变量的值。

2.4.2 标签（Label 类）

在 Tkinter 中，标签主要用于显示文本内容和图像。

1. 创建标签对象

可以通过 Tkinter 模块中的 Label 类创建标签对象，用于完成标签的创建，其语法格式如下：

```
Label(master, text, background, width, height, cursor, image, bitmap, anchor, relief,
textvariable)
```

其中，参数 master 表示标签的父容器；参数 text 表示标签的文本内容；参数 background 表示标签的背景颜色；参数 width 表示标签的宽度；参数 height 表示标签的高度；参数 cursor 表示当鼠标移动到标签上时光标的形状，其值包括 arrow、circle、cross 和 plus，默认值为 arrow；参数 image 表示标签的图片，该图片的类型需为 PhotoImage 类型、BitmapImage 类型，或者其他能兼容的类型；参数 bitmap 表示标签的位图，并且如果指定了标签的图片，则该选项忽略，常用的位图类型包括 gray75、gray50、gray25、gray12、error、hourglass、info、questhead、question 和 warning 等；参数 anchor 表示标签中文本内容或图像的位置，其值包括 n、s、w、e、ne、nw、sw、se 和 center，默认值为 center；参数 relief 表示标签的边框样式，其值包括 flat、sunken、raised、groove 和 ridge，默认值为 flat；参数 textvariable 用于修改标签的文本内容，并且必须与 Variable 类型的变量进行绑定。

2. 创建标签

示例代码如下：

```
#资源包\Code\chapter2\2.4\0204.py
import tkinter as tk
root = tk.Tk()
root.title('标签(Label 类)')
```

```
root.geometry('500x400+20+20')
root.resizable(width=False, height=True)
#在创建完控件之后,必须调用 Tkinter 中的布局管理器才可以正常显示控件.pack 就是 Tkinter
#中的布局管理器之一,关于布局管理器的相关知识,将在后续章节为读者进行详细讲解,本章节读
#者只需知道使用 pack 布局管理可以使控件正常显示
tk.Label(root, text="标签", background='yellow', height='5', width='50', cursor=
"plus").pack()
tk.Label(root, text="标签", background='pink', height='5', width='50', cursor=
"cross", anchor='e', relief='groove').pack()
#通过 PhotoImage 创建图片对象
photo = tk.PhotoImage(file='pic/oldxia.png')
tk.Label(root, image=photo).pack()
root.mainloop()
```

上面代码的运行结果如图 2-4 所示。

图 2-4 标签(Label 类)

这里需要注意的是,如果将上面创建标签的代码放在函数中,在运行代码之后,则会发现标签中的图片无法正常显示,示例代码如下:

```
#资源包\Code\chapter2\2.4\0205.py
import tkinter as tk
root = tk.Tk()
root.title('标签(Label 类)')
root.geometry('500x400+20+20')
root.resizable(width=False, height=True)
def createLabel():
    tk.Label(root, text="标签", background='yellow', height='5', width='50', cursor=
"plus").pack()
    tk.Label(root, text="标签", background='pink', height='5', width='50', cursor=
"cross", anchor='e', relief='groove').pack()
    #通过 PhotoImage 创建图片对象
```

```
    photo = tk.PhotoImage(file = 'pic/oldxia.png')
    tk.Label(root, image = photo).pack()
createLabel()
root.mainloop()
```

上面代码的运行结果如图 2-5 所示。

图 2-5 标签中的图片无法正常显示

这是因为虽然控件可以保存其内部对象的引用，但是 Tkinter 却无法正确处理图片对象的引用，即当函数调用完成之后，Python 的垃圾回收机制会通知控件释放其中的图片，但是由于该图片是由控件所使用的，所以控件并不会完全销毁该图片，而是将该图片清空，使其完全透明。

如果当函数内的控件使用到图片时，则必须保留对该图片对象的引用，否则图片就无法正常显示，示例代码如下：

```
#资源包\Code\chapter2\2.4\0206.py
import tkinter as tk
root = tk.Tk()
root.title('标签(Label 类)')
root.geometry('500x400 + 20 + 20')
root.resizable(width = False, height = True)
#定义图片列表,以达到保存图片对象引用的目的
lt_pic = [ ]
def createLabel():
    tk.Label(root, text = "标签", background = 'yellow', height = '5', width = '50', cursor =
"plus").pack()
    tk.Label(root, text = "标签", background = 'pink', height = '5', width = '50', cursor =
"cross", anchor = 'e', relief = 'groove').pack()
    photo = tk.PhotoImage(file = 'pic/oldxia.png')
    tk.Label(root, image = photo).pack()
    #将图片对象添加到图片列表中
```

```
    lt_pic.append(photo)
createLabel()
root.mainloop()
```

上面代码的运行结果如图 2-6 所示。

图 2-6　标签中的图片正常显示

2.4.3　按钮(Button 类)

在 Tkinter 中,按钮主要用于捕获用户的单击事件。

1. 创建按钮对象

可以通过 Tkinter 模块中的 Button 类创建按钮对象,用于完成按钮的创建,其语法格式如下:

```
Button(master, text, background, width, height, image, anchor, relief, command, textvariable,
state)
```

其中,参数 master 表示按钮的父容器;参数 text 表示按钮的文本内容;参数 background 表示按钮的背景颜色;参数 width 表示按钮的宽度;参数 height 表示按钮的高度;参数 image 表示按钮的图片,该图片的类型需为 PhotoImage 类型、BitmapImage 类型,或者其他能兼容的类型;参数 anchor 表示按钮中文本内容或图像的位置,其值包括 n、s、w、e、ne、nw、sw、se 和 center,默认值为 center;参数 relief 表示按钮的边框样式,其值包括 flat、sunken、raised、groove 和 ridge,默认值为 flat;参数 command 表示按钮关联的函数,即当按钮被单击时,所执行的函数;参数 textvariable 用于修改按钮的文本内容,并且必须与 Variable 类型的变量进行绑定;参数 state 用于设置按钮的状态,其值包括 normal、active 和 disabled,默认值为 normal。

2. 创建按钮

示例代码如下:

```
#资源包\Code\chapter2\2.4\0207.py
import tkinter as tk
root = tk.Tk()
root.title('按钮(Button类)')
root.geometry('500x400+20+20')
root.resizable(width=False, height=True)
def onclick():
    print('按钮被单击')
tk.Button(root, text='按钮', width='10', background='yellow',
relief='sunken', anchor='e', command=onclick).pack()
photo = tk.PhotoImage(file='pic/oldxia.png')
tk.Button(root, image=photo, command=onclick).pack()
root.mainloop()
```

上面代码的运行结果如图 2-7 所示。

图 2-7 按钮(Button 类)

2.4.4 单选按钮(Radiobutton 类)

在 Tkinter 中,单选按钮主要用于选中指定组内的一个选项。

1. 创建单选按钮对象

可以通过 Tkinter 模块中的 Radiobutton 类创建单选按钮对象,用于完成单选按钮的创建,其语法格式如下:

```
Radiobutton(master, text, value, background, width, height, image, anchor, relief, command,
variable, textvariable, state)
```

其中,参数 master 表示单选按钮的父容器;参数 text 表示单选按钮的文本内容;参数 value 表示单选按钮的值,并且在同一组中的所有单选按钮应该拥有各不相同的值;参数 background 表示单选按钮的背景颜色;参数 width 表示单选按钮的宽度;参数 height 表示

单选按钮的高度；参数 image 表示单选按钮的图片，该图片的类型需为 PhotoImage 类型、BitmapImage 类型，或者其他能兼容的类型；参数 anchor 表示单选按钮中文本内容或图像的位置，其值包括 n、s、w、e、ne、nw、sw、se 和 center，默认值为 center；参数 relief 表示单选按钮的边框样式，其值包括 flat、sunken、raised、groove 和 ridge，默认值为 flat；参数 command 表示单选按钮关联的函数，即当单选按钮被单击时，所执行的函数；参数 variable 表示与单选按钮相关联的 Variable 类型的变量，同一组中所有单选按钮的参数 variable 都应该指向同一个变量，并且通过将该变量与参数 value 的值对比，即可判断出用户所选中的单选按钮；参数 textvariable 用于修改单选按钮的文本内容，并且必须与 Variable 类型的变量进行绑定；参数 state 用于设置单选按钮的状态，其值包括 normal、active 和 disabled，默认值为 normal。

2. 创建单选按钮

示例代码如下：

```
#资源包\Code\chapter2\2.4\0208.py
import tkinter as tk
root = tk.Tk()
root.title('单选按钮(Radiobutton类)')
root.geometry('500x400+20+20')
root.resizable(width=False, height=True)
v = tk.IntVar()
#默认选项为"男"
v.set(100)
def onclick():
    print('单选按钮被单击')
tk.Radiobutton(root, text='男', variable=v, value=100, background='yellow',
width='10', anchor='e', relief='raised', command=onclick).pack()
tk.Radiobutton(root, text='女', variable=v, value=200).pack()
root.mainloop()
```

上面代码的运行结果如图 2-8 所示。

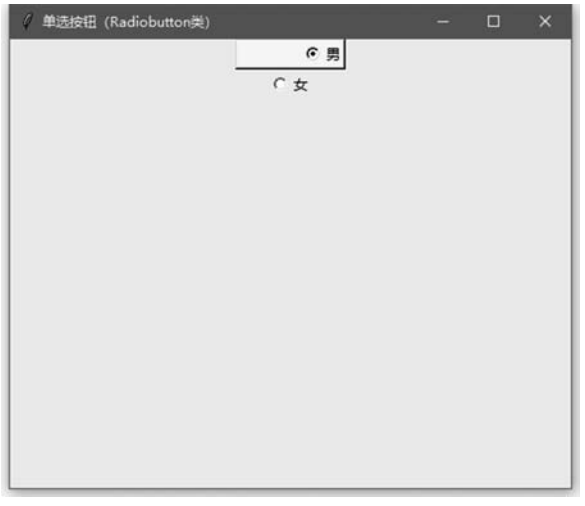

图 2-8　单选按钮(Radiobutton 类)

2.4.5 多选按钮(Checkbutton 类)

在 Tkinter 中,多选按钮主要用于同时选中指定组内的多个选项。

1. 创建多选按钮对象

可以通过 Tkinter 模块中的 Checkbutton 类创建多选按钮对象,用于完成多选按钮的创建,其语法格式如下:

```
Checkbutton(master, text,background, width, height, image, anchor, relief, command, onvalue,
offvalue, variable, textvariable, state)
```

其中,参数 master 表示多选按钮的父容器;参数 text 表示多选按钮的文本内容;参数 background 表示多选按钮的背景颜色;参数 width 表示多选按钮的宽度;参数 height 表示多选按钮的高度;参数 image 表示多选按钮的图片,该图片的类型需为 PhotoImage 类型、BitmapImage 类型,或者其他能兼容的类型;参数 anchor 表示多选按钮中文本内容或图像的位置,其值包括 n、s、w、e、ne、nw、sw、se 和 center,默认值为 center;参数 relief 表示多选按钮的边框样式,其值包括 flat、sunken、raised、groove 和 ridge,默认值为 flat;参数 command 表示多选按钮关联的函数,即当多选按钮被单击时所执行的函数;参数 onvalue 用于设置多选按钮选中状态的值;参数 offvalue 用于设置多选按钮未选中状态的值;参数 variable 表示与多选按钮相关联的 Variable 类型的变量,当多选按钮按下时,该变量在参数 onvalue 的值和参数 offvalue 的值之间自动切换;参数 textvariable 用于修改多选按钮的文本内容,并且必须与 Variable 类型的变量进行绑定;参数 state 用于设置多选按钮的状态,其值包括 normal、active 和 disabled,默认值为 normal。

2. 创建多选按钮

示例代码如下:

```
#资源包\Code\chapter2\2.4\0209.py
import tkinter as tk
root = tk.Tk()
root.title('多选按钮(Checkbutton 类)')
root.geometry('500x400+20+20')
root.resizable(width=False, height=True)
def onclick():
    print('多选按钮被单击')
var = tk.StringVar()
#Python 默认选中
var.set("on")
tk.Checkbutton(root, text='Python', width='10', background='yellow',
anchor='w', relief='raised', variable=var, onvalue='on', offvalue='off',
command=onclick).pack()
tk.Checkbutton(root, text='PHP', width='10', anchor='w',
command=onclick).pack()
tk.Checkbutton(root, text='C++', width='10', anchor='w',
command=onclick).pack()
tk.Checkbutton(root, text='Java', width='10', anchor='w',
command=onclick).pack()
root.mainloop()
```

上面代码的运行结果如图 2-9 所示。

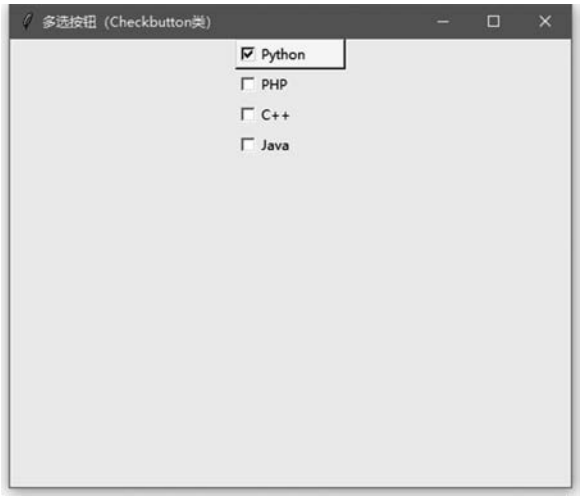

图 2-9　多选按钮(Checkbutton 类)

2.4.6　文本输入框(Entry 类)

在 Tkinter 中,文本输入框主要用于输入文本,并与用户进行交互。

1. 创建文本输入框对象

可以通过 Tkinter 模块中的 Entry 类创建文本输入框对象,用于完成文本输入框的创建,其语法格式如下:

```
Entry(master, show,background, width, cursor, relief, state, textvariable, xscrollcommand)
```

其中,参数 master 表示文本输入框的父容器;参数 show 用于设置文本输入框如何显示文本内容;参数 background 表示文本输入框的背景颜色;参数 width 表示文本输入框的宽度;参数 cursor 表示当鼠标移动到文本输入框上时光标的形状,其值包括 arrow、circle、cross 和 plus,默认值为 arrow;参数 relief 表示文本输入框的边框样式,其值包括 flat、sunken、raised、groove 和 ridge,默认值为 flat;参数 state 用于设置文本输入框的状态,其值包括 normal 和 disabled,默认值为 normal;参数 textvariable 用于修改文本输入框的文本内容,并且必须与 Variable 类型的变量进行绑定;参数 xscrollcommand 用于绑定水平方向上的滚动条。

2. 创建文本输入框

示例代码如下:

```
#资源包\Code\chapter2\2.4\0210.py
import tkinter as tk
root = tk.Tk()
root.title('文本输入框(Entry 类)')
root.geometry('500x400+20+20')
```

```
root.resizable(width = False, height = True)
entry1 = tk.Entry(root)
entry2 = tk.Entry(root, show = '*')
entry3 = tk.Entry(root, relief = 'ridge', state = 'disable')
entry1.pack()
entry2.pack()
entry3.pack()
root.mainloop()
```

上面代码的运行结果如图 2-10 所示。

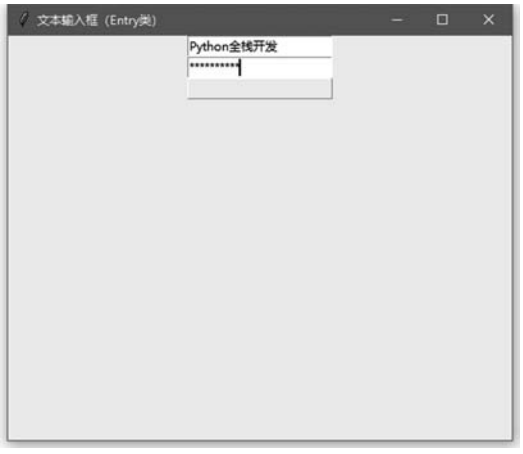

图 2-10 文本输入框(Entry 类)

2.4.7 下拉菜单(OptionMenu 类)

在 Tkinter 中,下拉菜单用于以下拉列表框的形式展现多个选项。

1. 创建下拉菜单对象

可以通过 Tkinter 模块中的 OptionMenu 类创建下拉菜单对象,用于完成下拉菜单的创建,其语法格式如下:

```
OptionMenu(master, variable, *value)
```

其中,参数 master 表示下拉菜单的父容器;参数 variable 表示与下拉菜单相关联的 Variable 类型的变量,用于指定下拉菜单的显示值;参数 value 表示下拉菜单的选项。

此外,还有一点需要重点注意,即 OptionMenu 类的参数不可以使用关键字参数的形式进行传参,而是必须使用位置参数的形式进行传参。

2. 创建下拉菜单

示例代码如下:

```
#资源包\Code\chapter2\2.4\0211.py
import tkinter as tk
root = tk.Tk()
```

```
root.title('下拉菜单(OptionMenu 类)')
root.geometry('500x400 + 20 + 20')
root.resizable(width = False, height = True)
op_list = ['未选择', 'Python', 'PHP', 'Java', 'C++']
val = tk.StringVar()
#设置下拉菜单的初始值
val.set(op_list[0])
tk.OptionMenu(root, val, * op_list).pack()
def onclick():
    label_val.set(val.get())
tk.Button(root, text = "获取下拉菜单的值", command = onclick).pack()
label_val = tk.StringVar()
#设置标签的初始文本内容
label_val.set('未选择')
tk.Label(root, textvariable = label_val, background = 'yellow', height = '5',
width = '50', cursor = "plus").pack()
root.mainloop()
```

上面代码的运行结果如图 2-11 所示。

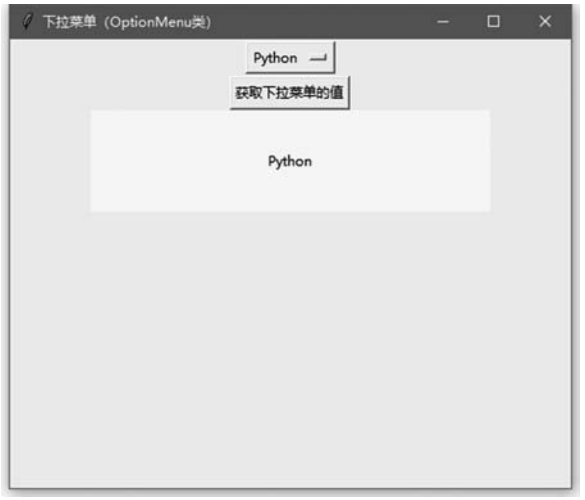

图 2-11　下拉菜单(OptionMenu 类)

2.4.8　列表框(Listbox 类)

在 Tkinter 中,列表框用于从列表中选中一个或者多个选项。

1. 创建列表框对象

可以通过 Tkinter 模块中的 Listbox 类创建列表框对象,用于完成列表框的创建,其语法格式如下:

```
Listbox ( master,  background,  width,  height,  cursor,  relief,  selectmode,  listvariable,
xscrollcommand, yscrollcommand)
```