《Python程序设计基础教程（微课版）》

http://dblab.xmu.edu.cn/post/python

第12章 图形用户界面编程

林子雨 博士/副教授

厦门大学计算机科学与技术系

E-mail: ziyulin@xmu.edu.cn

主页：http://dblab.xmu.edu.cn/linziyu

主讲教师

2017年度厦门大学奖教金获得者
2020年度厦门大学奖教金获得者

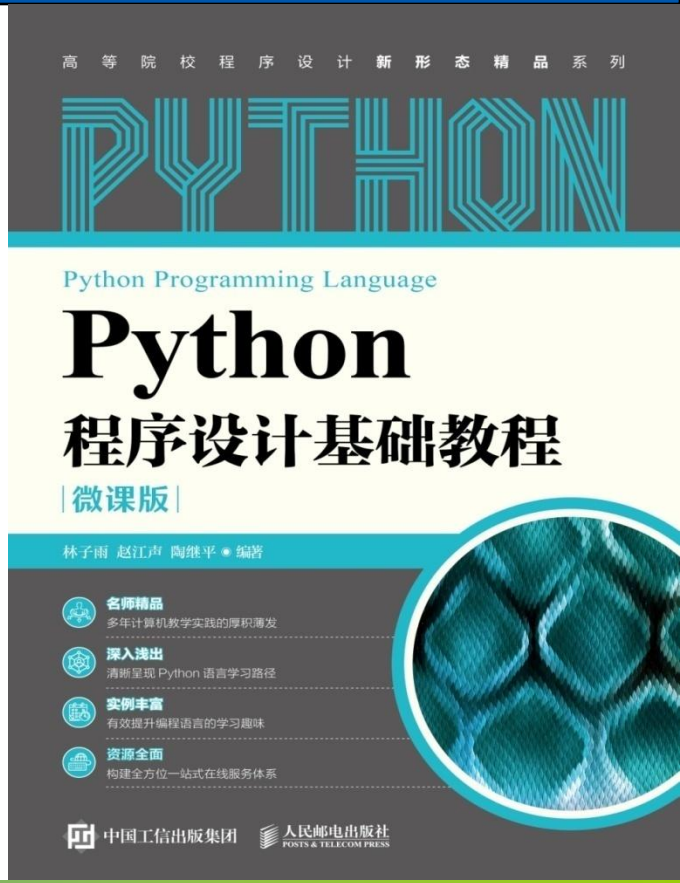
主讲教师：厦门大学 林子雨 博士/副教授
中国高校首个“**数字教师**”提出者和建设者
2009年7月从事教师职业以来
累计**免费**网络发布超过1500万字高价值教学和科研资料
网络浏览量超过1500万次

第12章 图形用户界面编程

本PPT是如下教材的配套讲义：
《Python程序设计基础教程（微课版）》

厦门大学 林子雨,赵江声,陶继平 编著，人民邮电出版社

《 Python程序设计基础教程（微课版） 》教材官方网站：
http://dblab.xmu.edu.cn/post/python



12.1 面向对象编程概述

12.1.1 从命令行界面到图形用户界面
12.1.2 图形用户界面程序的运行与开发
12.1.3 Python中的图形界面编程

12.1.1 从命令行界面到图形用户界面

•与命令行界面完全采用文本进行信息交互的方式不同，图形用户界面通过按钮及文本框等图形化元素实现程序与用户的信息交互。在图形用户界面中，用户通过鼠标点击或拖拉菜单、按钮、窗口等图形元素向程序发出命令，同时，程序通过文本消息框等图形元素向用户显示信息。

•与命令行界面相比，图形用户界面最大的优势在于简单直观，不需要用户记住各种复杂的文本命令，只需要简单地操作鼠标就可以与计算机进行交互，具有更强的“用户友好性”，降低了计算机技术的使用门槛。计算机应用技术的普及在很大程度上正是得益于图形用户界面的出现。

12.1.1 从命令行界面到图形用户界面

•图形用户界面中的基本图形元素称为“控件（control）”或“构件（widget）”。窗口（window）是图形用户界面中最基本的控件。

•一个图形界面应用程序至少包含一个窗口。窗口通常的作用是用于放置其它控件，因此也称为“容器控件”。

•窗口支持的基本操作包括移动和改变大小。当窗口被移动时，其包含的其它控件也跟随一起被移动。除了作为容器的窗口控件外，其它常用控件可以按功能划分为四大类，分别是分组的选择及显示、文本输入、输出显示、导航。各个类别所包含的常用控件及功能如表12-1所示。

12.1.1 从命令行界面到图形用户界面

表12-1 常用控件及其功能

类别	控件	功能简述
分组的选择及显示	命令按钮（Button）	通过鼠标点击来执行相关操作的控件，类似机电仪器设备上的按钮
	单/复选按钮 （Radio button/Check box）	从一组选项中选择一个或多个选项。通常，单选按钮用一个小圆圈表示，复选按钮用一个小方框表示
	列表框 (List box)	允许用户从一个静态的多行文本框列表中选择一个或多个项
	下拉列表 (Drop-down list)	类似列表框。仅在鼠标点击时才显示整个列表项，非活动状态只显示一项已被选中的内容或者留空
	菜单 (Menu)	具有多个可选操作的控件，可单击某一项激活相关操作
	工具条 （Toolbar）	用于放置按钮及菜单等其它控件，用于快捷访问
文本输入	文本框 (Text box)	允许用户进行文本输入的控件
	组合框 （Combo box）	组合了文本框和下拉列表的复合控件，允许用户手动输入文本或者从下拉列表中选择已有内容
输出显示	标签 （Label）	用于描述其它控件的文本
	状态条 （Status bar）	用于简要显示程序相关动态信息的区域，通常位于窗口底部
	进度条 （Progress bar）	用于可视化诸如下载等需要持续较长时间操作的进度
导航	滚动条 （Scrollbar）	用于在一个窗口内按各个方向(上、下、左或右)滚动显示连续的文本或图片等内容

12.1.2 图形用户界面程序的运行与开发

•命令行界面程序一般采用过程驱动的程序设计方法。程序从启动开始按顺序运行，在需要的地方提示用户输入，并将相关计算结果输出，直到执行完所有指令结束退出。在这个过程中，用户的所有输入行为都完全由程序控制，如果没有程序的输入请求，除非强行终止，否则用户不能对程序的运行做任何额外的干涉。

•与命令行界面程序不同的是，图形界面程序的执行路径是由用户控制的，用户可能随时做出干预。例如，操作过程中可能调整窗口的大小或者点击某个按钮等。用户的这类行为是不可预期的。为了适应这种特点，图形用户界面程序采用了事件驱动的程序设计模式。

12.1.2 图形用户界面程序的运行与开发

•事件指的是用户与程序的交互行为。例如，单击某个按钮，改变窗口大小，或者在文本框里输入文本等。一旦发生某个特定的事件，程序就必须做出相应的操作来响应该事件（什么都不做也是一种响应），这些响应称为“事件处理程序”。

•事件处理程序通常对应于一个函数或方法，由于这个函数是在相应事件发生时被自动调用，因此也常称为“回调（callback）函数”。

12.1.2 图形用户界面程序的运行与开发

•图形界面程序启动后，首先创建根窗口，并加载诸如菜单栏、工具栏及状态条等控件。

•在创建完这个初始的图形界面，并进行一些必要的初始化工作后，开始启动一个所谓的事件循环，该循环不停地监测是否有事件发生，一旦发生了事件，将交给事件处理程序进行处理。

•这一循环直到发生了程序退出事件（用户关闭主窗口）才终止运行。

12.1.2 图形用户界面程序的运行与开发

•GUI 程序的开发一般包括两大类工作，即界面外观设计和业务逻辑程序设计。界面外观设计主要包括各种控件的设计以及窗口的整体布局规划；业务逻辑程序设计是GUI程序开发的核心任务，包括应用问题的建模，管理应用问题的数据和行为，同时还要负责用户交互的事件处理程序。这些工作涉及很多与操作系统相关的底层细节，如果完全从零开始写代码，将涉及到很多复杂琐碎而又与实际业务逻辑无关的工作。

•实际上，不同的GUI程序在功能上存在很多通用的地方，因此，很多第三方的厂商或社区就会将这些共性的功能抽象成与具体应用无关的工具包（toolkit），提供给开发者使用。这些工具包通常也称为“GUI库”。一个GUI库包含了各种常用控件以及基本的事件循环框架的实现。这些GUI库将极大简化GUI程序的开发，使得开发人员只需要专注于具体的业务逻辑，提高了开发效率。

12.1.3 Python中的图形界面编程

•Python本身并不提供原生的完全由Python语言写的GUI库，而是通过在其它语言编写的GUI库之上加一个Python的封装接口。也就是说，虽然可以像使用其它Python模块一样使用GUI库，但其实现的功能并不是由Python提供的。

•Python的标准库里包含了Tk图形界面库，它是采用一种名为Tcl的脚本语言和C语言进行编写的，因此有时也写为Tcl/Tk。Tk具有轻量、可定制及跨平台等诸多优点，非常适合原型系统的开发。Python自带的集成开发环境IDLE就是使用Tk 实现用户界面的开发。Tk在Python里面被封装为tkinter包。严格意义上讲，tkinter并不是Python标准库的一部分，只是在有些平台的Python发布版中默认跟随标准库一起安装，也就成了事实上的标准库。

12.1.3 Python中的图形界面编程

•尽管tkinter中基本控件的外观显得相对比较简陋，但作为Python的事实标准库，tkinter的最大优点是轻量和稳定，非常适合对GUI美观要求不高的中小型原型系统的开发。

•对于没有任何GUI编程经验的初学者，在学习完Python的基本语法知识后，建议从tkinter开始入门GUI编程。

12.2tkinker概述

•如上节所述，tkinter是Tk图形库在Python下的封装，它对应Python的一个包。

•这个包在Python的Windows二进制发布版中是默认安装的，在Ubuntu下可以用“sudo apt-get install python3-tk”等方式手动安装，其它系统，请查阅相关资料进行安装。

•安装完毕后，可以在命令行下输入“python –m tkinter”测试是否正常，如果一切正常，该命令将显示一个简单的GUI界面，其中显示了tkinter所对应的Tk库的版本。

12.2tkinker概述

下面分别是该界面在Windows 10和Ubuntu 18.04下的截图，可以看出显示样式稍有差异。后文所有的测试例子都将是针对Windows 10操作系统，在其它平台下的显示样式可能存在差异。

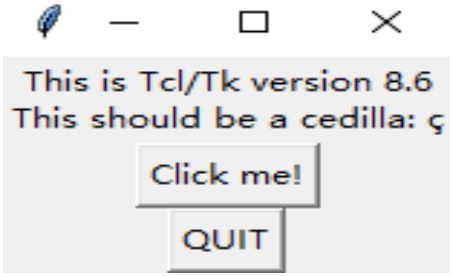
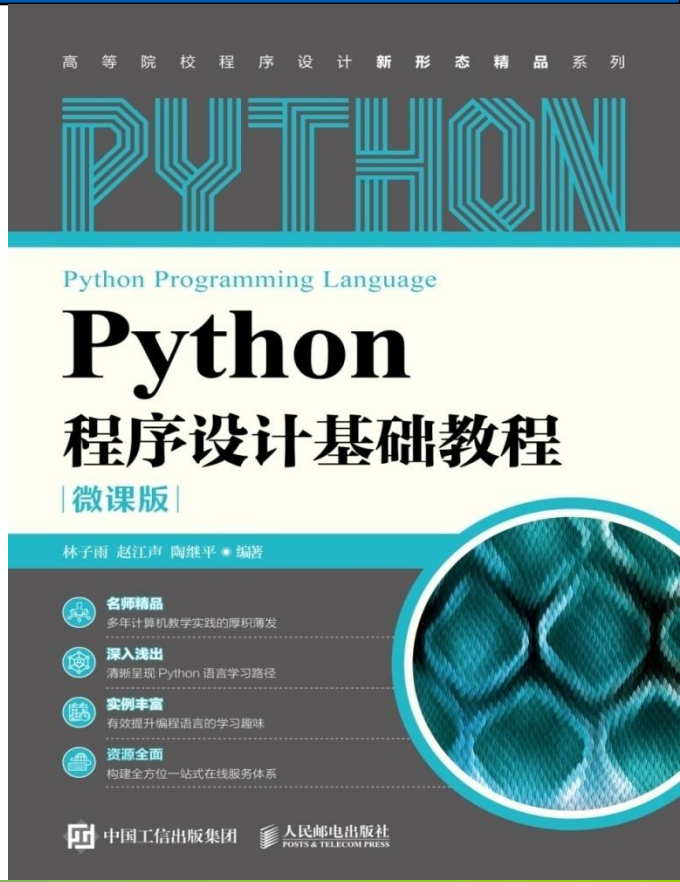


图12-1 tkinter在Windows10下的测试程序

图12-2 tkinter在Ubuntu18.04下的测试程序

12.2tkinker概述

12.2.1类的层次结构
12.2.2基本开发步骤



本PPT是如下教材的配套讲义：
《Python程序设计基础教程（微课版）》

厦门大学 林子雨,赵江声,陶继平 编著，人民邮电出版社

《 Python程序设计基础教程（微课版） 》教材官方网站：
http://dblab.xmu.edu.cn/post/python

12.2.1类的层次结构

•tkinter是完全按照面向对象的方式进行组织的。各种控件的显示与交互控制都是通过相应的Python类来实现。

•tkinter中最基本的一个类是Tk类（注意首字母是大写的）。每个应用程序都需要也只需要一个Tk类的实例，该实例表示应用程序，同时也表示应用程序的根窗口。

•根窗口是可以在用户屏幕上随意移动和改变大小的，这种窗口也称为“顶层窗口”。

12.2.1类的层次结构

•一个简单的应用程序一般只有一个顶层窗口，而一些复杂的程序可能有多个顶层窗口，这时就需要用到TopLevel类，用户一般不直接实例化TopLevel类的实例，而是通过继承的方式创建特定样式的顶层窗口类。

除了Tk和TopLevel类，其它常用类包括Frame、Label、Entry、Text、Button、Radiobutton、Checkbutton、Listbox、Scrollbar、Scale、LabelFrame、Menu、Spinbox及Canvas等，这些都用于构造一个特定功能的GUI控件，大部分从名字就可以看出其功能。

这些控件不同于顶层窗口，在默认情况下，它们不能由用户移动和改变大小，只能跟随父窗口移动。

12.2.1类的层次结构

•除了以上这些控件类，主要还包括用于布局管理的Pack、Grid和Place三个类，以及用于用户事件处理的Event类。这些都属于非控件类，一般都不需要程序直接实例化相应的对象，tkinter会在需要时自动生成对象，用户只需要调用控件的相关方法。

•另外，在构造控件时还经常涉及的一个概念是“控制变量”，主要用于在多个控件之间共享一些共同的属性值。一旦该变量的值改变了，与之共享的各个控件的相应属性也自动改变。

•tkinter提供了三种类型的控制变量，对应StringVar、IntVar和DoubleVar三个类，分别表示字符串型、整型和浮点型的控制变量。为了创建控制变量，只需要使用相应类的构造器进行实例化，并使用get和set方法进行读取和设置控制变量的值。

12.2.1类的层次结构

以上提到的这些类都是直接定义在包的__init__文件中，因此导入tkinter包后可以直接使用。另外，tkinter还包含了多个子模块，主要提供了一些样式更丰富或者具有特定功能的控件。主要的子模块包括：

（1）filedialog子模块：提供了用于文件操作的对话框；
（2）font子模块：封装了用于字体样式控制的相关类；
（3）ttk子模块：对Tk库在8.5版本后引入的所谓主题式控件的封装，其使得控件在外观上更接近平台的原生界面样式。ttk子模块重写了tkinter主模块中同名的基本控件类，但使用方式有所不同，另外，ttk还提供了树状视图等高级控件；
（4）constants子模块：包含了很多预定义的常量值，例如表示布局关系的TOP、BOTTOM、LEFT、RIGHT等，该子模块在导入主模块时默认被导入，可以直接使用。

12.2.2基本开发步骤

简单来说，一个tkinter程序的开发就是通过实例化各种控件类得到一组控件对象，然后再使用这些控件对象进行界面的布局设计，并绑定相应的事件处理程序。tkinter程序的开发主要包括如下几个基本步骤：

12.2.2基本开发步骤

1、Tk类的实例化

调用Tk类的构造器来实例化一个Tk实例，根据需要，可以指定程序名及图标等属性。每个应用程序都需要也只需要一个Tk类的实例，如果不显式构造Tk实例，在创建其它控件时会默认创建，但不建议这么做。对于某些GUI，为了防止用户调整根窗口导致内部布局混乱，通常需要设置根窗口的初始化大小、最大最小宽度，或者限制窗口的缩放功能。

这里会涉及如下个几方法：

- geometry(“width*height”): 设置窗口的初始宽高，注意参数为字符串类型；
- maxsize(width,height)： 设置用户拖曳时窗口的最大宽和高；
- minsize(width,height) ： 设置用户拖曳时窗口的最小宽和高；
- resizable(width_resizable, height_resizable):设置是否允许在宽和高方向进行拖拽。

12.2.2基本开发步骤

2、创建各种控件实例

•每个控件对象的构造流程都类似。只需要调用tkinter提供的控件类的构造器，指明父窗口以及各种外观及行为属性。除self参数外，控件类构造器的第一个参数是可选参数master，表示父窗口对象，默认为None；第二个参数是可选参数cnf，默认是一个空的字典对象；第三个参数是可变的关键字参数。后两个参数都是用于控制控件对象的相关外观和行为属性（在Tk文档中称为configuration options），不同类型的控件所支持的属性不完全一样。

•每一个控件实例都必须有一个父窗口，父窗口可以是Tk对象，也可以是其它容器类控件实例。如果将父窗口设置为None，程序将自动寻找已存在的Tk类实例，如果没有，将自动创建一个Tk类实例作为父窗口，但强烈建议显式设置父窗口。

12.2.2基本开发步骤

3、对各个控件进行布局

•大部分控件在创建后还不能直接显示在父窗口中，必须进一步确定其在父窗口中的具体位置以及与其他控件的摆放关系。

•tkinter提供了三种布局管理器（Geometry Manager）来实现不同的布局需求，对应名为Pack、Grid和Place的三个类。

•对控件进行布局时，不需要显式创建这些布局管理器类的对象，只需要调用控件对应的一个布局方法，分别是pack、grid和place。

12.2.2基本开发步骤

4、事件绑定

•除了界面外观的设计，GUI应用程序开发的另一个主要任务就是事件处理程序的设计。事件处理程序是对用户各种操作事件的响应处理，例如鼠标单击和键盘输入等，不同的事件可能需要不同的处理程序，这种事件与事件处理程序之间的关系是通过所谓的事件绑定来建立的。

•可以通过控件的相关属性或者bind方法进行事件绑定。

12.2.2基本开发步骤

5、启动事件循环程序

•如果是在解释器环境里一句句执行语句，则在Tk对象创建后，图形界面就已经显示在用户屏幕上，其它控件在进行布局后也会先后显示，并且可以接收用户的交互操作。

•但如果是通过脚本文件解释执行，在完成上述步骤后，图形界面还没有真正显示在屏幕上，还需要主动调用Tk对象的mainloop方法，该方法会显示设计好的界面，并启动事件循环程序，接收用户的交互操作。

12.2.2基本开发步骤

```
01 # -*- coding: utf-8 -*-
02 # 例12-1：hello.py
03 import tkinter as tk
04 app = tk.Tk()
05 app.title("hello")
06 app.iconbitmap("python.ico")
07 label = tk.Label(app, text="欢迎开启GUI编程之旅！ ")
08 label.pack(padx=50,pady=5)
09 def change_button_text():
10             btn.configure(text="[%s]" % btn['text'])
11 btn = tk.Button(app,text="点击我",command=change_button_text)
12 btn.pack(padx=50,pady=5)
13 app.mainloop()
```

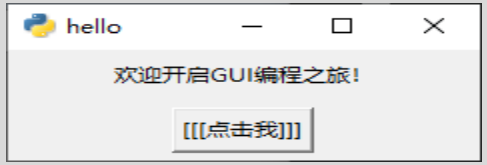


图12-3 hello.py的执行效果

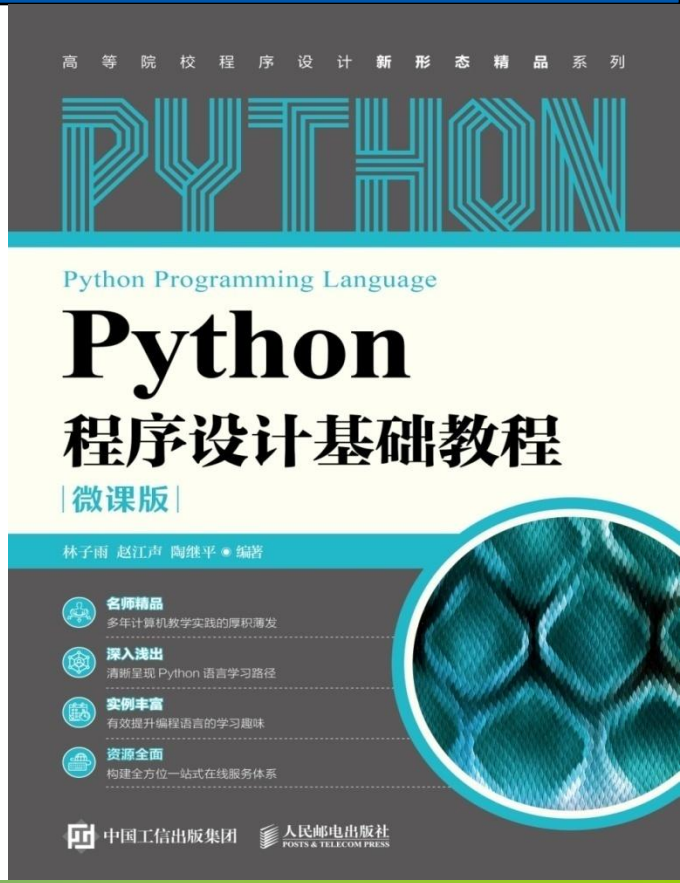
12.3 tkinter常用控件的使用

12.3.1常用控件的基本属性
12.3.2 Label
12.3.3 Button
12.3.4 Entry
12.3.5 Checkbutton
12.3.6 Radiobutton
12.3.7 Listbox
12.3.8 Frame/LabelFrame

本PPT是如下教材的配套讲义：
《Python程序设计基础教程（微课版）》

厦门大学 林子雨,赵江声,陶继平 编著，人民邮电出版社

《 Python程序设计基础教程（微课版） 》教材官方网站：
http://dblab.xmu.edu.cn/post/python



12.3.1常用控件的基本属性

1、尺寸属性

•每个控件在视觉上都显现为屏幕上的一个矩形区域，区域的内部为控件内容，区域的外围有一个边框（根据不同的样式，可能在视觉上是不可见的），在边框的内外可以设置边距。

•该矩形区域在屏幕上显示的实际尺寸只能查看，不能直接设置。程序直接设置的尺寸称为“期望尺寸”，在显示时能否达到这个期望值，还受到布局的影响，具体造成怎样的影响，由布局管理器控制。

12.3.1常用控件的基本属性

•可以调用控件的winfo_reqwidth方法和winfo_reqheight方法查看设计的期望宽高（其中，字符req是单词requested的缩写），单位为像素点，该值由三类属性共同决定。

•第一类属性是width和height，表示为矩形区域内部的控件内容预留的期望宽高；第二类是padx和pady，表示在控件内容与边框之间预留的期望间距；第三类是borderwidth，表示控件的边框宽度。

•根据控件内容不同，padx和pady的作用略有不同，大体上，在单位统一的情况下，控件的整体期望尺寸与上述三类属性的关系为“期望尺寸≈期望宽高 + 2*期望间距 + 2*边框”。

12.3.1常用控件的基本属性

•对于只包含文本的非容器类控件，属性width和height的取值只能是一个整数，表示占用多少个标准字符的宽和高。对于其它非容器类控件和所有容器类控件，属性width和height的默认单位为p，表示像素点，其它单位还包括i（英寸）、c（厘米）及m（毫米）。如果将width或height设置为0，表示自适应所包含内容的大小。padx、pady以及borderwidth的默认单位为像素，也可以用其它单位。

•关于控件的尺寸属性，一个使用建议是，对于容器类控件，不要手动设置width和height属性，应该由其所包含的子控件来自动适应，根据排版需要，可以适当设置padx和pady的值。对于非容器类控件，可以根据需要，将width或height设置为指定的值。

12.3.1常用控件的基本属性

2、边框属性

每一个控件最外围都有一个边框，这个边框涉及两个属性borderwidth和relief，分别表示边框的宽度和边框的3D效果。relief的取值包括5个预定义的常量，分别是FLAT、RAISED、SUNKEN、RIDGE、GROOVE。不同属性值对应的显示效果如图12-4所示，其包含了五个分别取不同relief值的按钮。不同控件对应的relief属性的默认值不同，例如Label控件默认为FLAT，Button控件默认为RAISED。

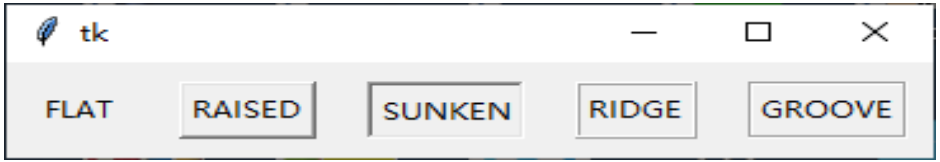


图12-4 relief不同的取值效果对比

12.3.1常用控件的基本属性

3、颜色属性

•每个控件最常用的两个颜色属性是bg和fg。bg表示背景颜色，fg表示前景颜色。颜色属性的取值可以是已经定义的标准颜色名，例如'white', 'black', 'red', 'green', 'blue', 'yellow'等，还可以是一个形如“#rrrgggbbb”的9位16进制串，分别表示红绿蓝三种基准色的比例，例如“#000ffffff”表示纯绿色和蓝色的混合色。

•如果想了解控件所支持的完整属性，可以调用控件的keys方法，该方法将返回控件所支持的全部属性名，关于每个属性名所代表的含义，可以参阅官方文档。可以通过以下四种方式设置控件的属性：

12.3.1常用控件的基本属性

•方式一：用属性名和相应的值作为键值对构造一个字典对象，在实例化控件时将该字典对象作为第二个位置参数传入（第一个参数为父窗口）。在多个控件之间共享一些共同的属性值时，可以用这种方式进行设置。

•方式二：在实例化控件时用“属性名=值”的形式作为关键字参数传入，用这种方式传入的属性会自动覆盖方式一中传入的字典对象的同名属性。

•方式三：tkinter提供了一种快速访问控件属性的语法糖，可以用“控件实例名['属性名']”的形式读写属性。

•方式四：调用控件对象的configure方法(也可以用同名方法config)，并以“属性名=值”的形式作为关键字参数传入，可以通过传入多个参数同时设置多个属性。

12.3.1常用控件的基本属性

下面的例子通过以上几种方式设置一个Label控件的属性，可以在解释器环境下逐条运行以查看界面效果。

```
>>> from tkinter import *
>>> app = Tk()
>>> label=Label(app,{'bg':'red','fg':'yellow'}, text="hello world") # 通过方式一初始化bg和fg属性，通过方式二初始化text属性
>>> label.pack()#对控件进行布局，pack方法将在12.4节具体介绍
>>> label['text']='I love python' # 通过方式三修改text属性
>>> label.configure(padx=10,pady=20) # 通过方式四修改边距属性
>>> label.keys() # 返回所有Label控件支持的所有属性
['activebackground', 'activeforeground', 'anchor', 'background', 'bd', 'bg', 'bitmap', 'borderwidth', 'compound', 'cursor', 'disabledforeground', 'fg', 'font', 'foreground', 'height', 'highlightbackground', 'highlightcolor', 'highlightthickness', 'image', 'justify', 'padx', 'pady', 'relief', 'state', 'takefocus', 'text', 'textvariable', 'underline', 'width', 'wraplength']
```

12.3.2 Label

Label是用于显示文本或图片的标签控件。这些文本或图片在程序中可以随时更新，但对终端用户是不可编辑的，主要用于界面各项功能的提示。除前文介绍的公共属性外，Label的其它常用属性如下。

•text：要显示的文本字符串；
•bitmap: 要显示的位图图像，tkinter提供了一些内置的位图图像，可以直接用相应的字符串引用，包括‘error’, ‘gray75’, ‘gray50’, ‘gray25’, ‘gray12’, ‘hourglass’, ‘info’, ‘questhead’, ‘question’, 及‘warning’，具体样式见图12-5。也可以自己建立相应的位图文件。

图12-5 test_label.py的执行效果

12.3.2 Label

•image：要显示的全色图像，可以使用tkinter中的PhotoImage控件构造一个图像对象，PhotoImage控件支持png、gif、pgm及ppm四种图片格式，如果需要用到其它图片格式，则需要相应图像库支持，例如广泛使用的pillow库。image属性比bitmap优先级高，如果同时设置，将优先显示image属性。

•compound：当标签上同时有文本和图像时，compound属性控制文本和图像的显示关系，默认值为None，表示只显示图像，其它可选值包括“text”（只显示文本）、“image”（只显示图片）、“center”（文本在图片中间）、“top”（图片在文本上方）、“left”（图片在文本左边），“bottom”（图片在文本下方）和“right”（图片在文本右边）。下面的程序通过内建的所有位图建立了相应的标签，并同时显示了文本，另外还使用一个外部的png文件建立了一个标签，执行效果如图12-5所示。

图12-5 test_label.py的执行效果

12.3.2 Label

```
01  # -*- coding: utf-8 -*-
02  # 例12-2：test_label.py
03  from tkinter import *
04  app = Tk()
05  bitmaps = ['error', 'gray75', 'gray50', 'gray25', 'gray12', 'hourglass', 'info',
'questhead', 'question', 'warning']
06  for b in bitmaps: # 遍历bitmaps生成多个标签控件
        #创建标签后，同时调用pack方法对控件进行布局
07      Label(text=b,bitmap=b,compound="left").pack(side=LEFT, padx=3)
08  img=PhotoImage(file="like.png") # 通过一个外部图片生成一个图像对象
09  label = Label(text="Like",image = img,compound="top")#创建标签
10  label.pack(side=LEFT,pady=3)#调用pack方法进行布局
11  app.mainloop()
```

12.3.2 Label

•anchor：如果标签的大小超过了内容的大小，anchor属性控制内容相对标签的放置方位，anchor的取值为预定义的一些常量字符串，包括N、S、W、E、NW、SW、NE、SE和CENTER，例如，NW和W表示西北角和西边（左西右东、上北下南），其它类似，默认取值为CENTER，表示居中放置。该属性只有在标签的大小比内容大时才起作用。

•justify：控制文本的对齐方式。justify的取值也为预定义的一些常量字符串，分别是CENTER（居中对其，默认值）、LEFT（左对齐）和RIGHT（右对齐）。

12.3.2 Label

•wraplength：控制文本占用的屏幕宽度达到多少后自动换行。默认单位为像素点，可以采用其它单位。0表示不自动换行。下例通过一段文字的显示对anchor、justify及wraplength属性进行了演示，运行结果如图12-6所示，可以修改相应属性的值并观察界面变化。

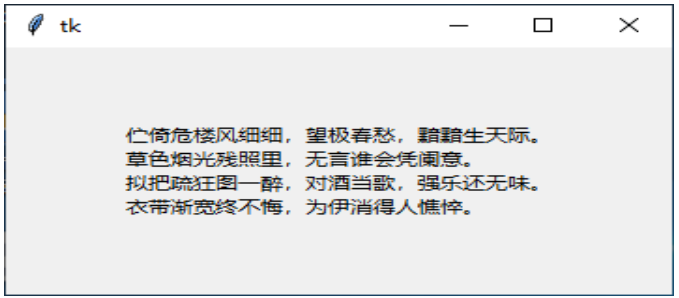


图12-6 test_anchor.py执行效果

12.3.2 Label

```
01  # -*- coding: utf-8 -*-
02  # 例12-3：test_anchor.py
03  from tkinter import *
04  app = Tk()
05  text = """伫倚危楼风细细，望极春愁，黯黯生天际。
06  草色烟光残照里，无言谁会凭阑意。
07  拟把疏狂图一醉，对酒当歌，强乐还无味。
08  衣带渐宽终不悔，为伊消得人憔悴。"""
09  label=Label(text=text,width=50, height=10)
10  label['justify']=LEFT # 尝试改为CENTER等值
11  label['wraplength']=300 # 尝试改为诸如200等更小的数值
12  label['anchor'] = CENTER # 尝试改为N等值
13  label.pack()
14  app.mainloop()
```

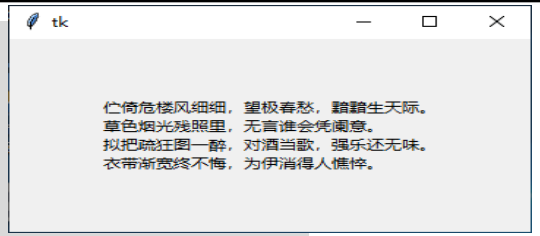


图12-6 test_anchor.py执行效果

12.3.3 Button

Button表示命令式按钮控件。主要用于捕获鼠标单击事件，以启动预定义的处理程序。Button控件的边框relief属性默认值为RAISED，使得其外观看起来像一个物理的按钮。类似上面的Label控件，Button控件上也可以包含文本和图片，因此，Button也具有上文Label中列出的各种属性。与Label不同的是，Button默认响应鼠标点击事件，涉及到的几个常用属性如下：

•command：表示点击时相应的事件处理程序。其取值为某个函数或方法对象，也可以是一个匿名函数。这个函数或方法不能包括位置参数。

•state：按钮状态，表示是否接受用户点击。默认NORMAL表示可以点击，DISABLED表示不响应点击，此时按钮表面显示为灰色；

12.3.3 Button

下面的程序演示了按钮的基本使用。在按钮的事件处理程序中，首先将按钮的状态改为DISABLED状态，并调用Tk对象的update即时刷新界面，延时5秒后再恢复按钮状态。执行效果如图12-7所示。

图12-7 test_button.py执行效果

12.3.3 Button

```
01 # -*- coding: utf-8 -*-
02 # 例12-4：test_button.py
03 from tkinter import *
04 import time
05 app = Tk()
06 texts={'begin':'点击按钮开始计算',
07 'computing':'计算中...','end':'计算完成,点击按钮开始重复计算'}
08 def compute():
09     info['text']=texts['computing'] #设置提示文本信息
10     btn['state']=DISABLED # 修改按钮状态为不可点击
11     app.update() # 即时刷新界面，否则要等到函数返回才刷新。
12     time.sleep(5) # 延时5秒以模拟长时间计算过程
13     info['text']=texts['end'] #设置提示文本信息
14     btn['state']=NORMAL #恢复按钮为可点击状态
15 info = Label(text=texts['begin'],width = 50)
16 info.pack(side=TOP,pady=5) #对控件进行布局
17 btn = Button(text="开始",command=compute)
18 btn.pack(side=TOP,pady=5) #对控件进行布局
19 app.mainloop()
```

12.3.4 Entry

•Entry表示单行文本框控件，用来读取用户输入的单行字符串。由于只能容纳单行文本，Entry控件没有height属性，width属性表示文本框中预留的标准字符数目。

•如果输入的字符串长度大于设定的宽度，输入的文字会自动向左隐藏，此时可以使用键盘上箭头键移动鼠标光标到看不到的区域。

12.3.4 Entry

Entry的其它常用属性和方法包括：

•textvariable：用于绑定用户输入文本的控制变量，一般设为某个StringVar类型的对象，程序的其它位置就可以用控制变量的get方法得到用户的输入；

•show：默认为空，这时用户输入什么，文本框内就显示什么，如果设置为一个字符，则不论用户输入什么，文本框都显示为该设置的字符，通常在密码输入时设为‘*’。该属性只是控制屏幕上的显示，不影响上面所绑定的控制变量的值；

•state：输入状态，默认NORMAL表示可以输入，DISABLED表示无法输入；
•select_range(start, end)：将文本框内相应索引范围内的字符改为选中状态。

12.3.4 Entry

•对于文本输入，为了体现更好的用户友好性，有时会要求对输入的合法性进行实时检查，即用户每输入一个字符，都要检查是否合乎要求，在出现非法输入时及时提醒用户。

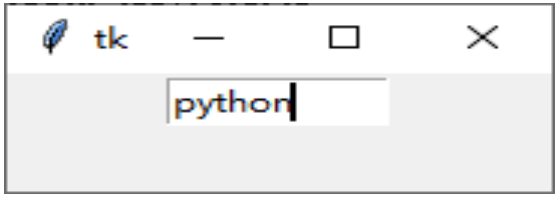
•为了达到这个要求，最常用的做法是调用所绑定控制变量的trace_add方法来为变量绑定一个回调函数。trace_add方法接收两个参数，第一个是表示追踪模式的字符串，第二个是回调函数。

•追踪模式最常用的取值为“write”，即表示仅在控制变量被改写时才执行回调函数，其它取值还有“read”和“unset”，分别表示读取时和变量被删除时执行，一般较少用到。

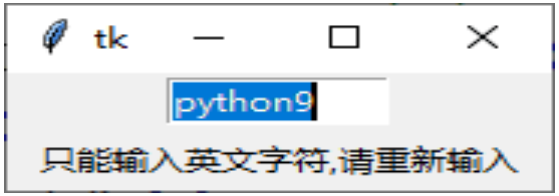
•下例简要演示上述过程，其中的文本框要求只能为英文字母或空格，当用户输入其它字符时就会提示。

12.3.4 Entry

```
01 # -*- coding: utf-8 -*-
02 # 例12-5：input_check.py
03 from tkinter import *
04 import re
05 app = Tk()
06 text= StringVar() # 定义一个控制变量
07 def check(*arg): #控制变量的回调函数
08     newval = text.get()#获取控制变量的值
09     if re.match('^[a-z A-Z]*$', newval) is None:#正则表达式进行匹配
10         entry.select_range(0,END) # 调用select_range框选当前输入
11         output['text']="只能输入英文字符,请重新输入"
12     else:
13         output['text']=""
14 text.trace_add("write",check)#为控制变量添加回调函数进行合法性检查
15 entry = Entry(app,width="10",textvariable=text)# 创建单行文本框控件
16 entry.pack(pady=2) #对控件进行布局
17 output = Label(app,width=25,text="")
18 output.pack(pady=2)
19 app.mainloop()
```



a) 正常输入



b) 非法输入

图12-8 input_check.py执行效果

12.3.5 Checkbutton

•Checkbutton表示复选按钮控件，也称“多选按钮”，用于向用户提供一个可选的选项。复选按钮在外观上由一个小方框和一个相邻的描述性标题组成。

•其中的方框在未选中时里面为空白，选中时会出现勾号，描述性标题类似一个标签控件，可以包含图片或者文本。

•同时，复选按钮也具有类似Button的性质，默认相应鼠标单击事件，每次单击方框或标题时，复选按钮的选择状态发生改变（从选中到未选中，或相反），同时执行command属性对应的回调函数。

12.3.5 Checkbutton

除了具有上文Label和Button中列出的各种属性，Checkbutton其它常用属性和方法包括：

•variable：用于绑定复选按钮选择状态的控制变量，一般设为某个IntVar类型的对象。默认情况下，复选按钮选中时其值为1，未选中为0。如果设置了下面的onvalue/ offvalue为非整型值，则要改变控制变量为相应类型；

•onvalue/ offvalue：复选按钮选中和未选中时控制变量对应的值，默认分别为1和0；

•select()：将复选按钮设为选中状态；

•deselect(): 将复选按钮设为未选中状态；

•toggle()：改变复选按钮的选中状态。

12.3.5 Checkbutton

•提供现成的选项让用户选择是图形用户界面设计中经常使用的做法。用户不需要记住各种复杂的命令，只需要用鼠标在已有的选项中点选即可。

•下例中创建了4个复选框按钮，分别表示4个候选项，并在用户按下“检查”按钮时检查复选框的选择情况，按下“提示”按钮时给出正确答案。

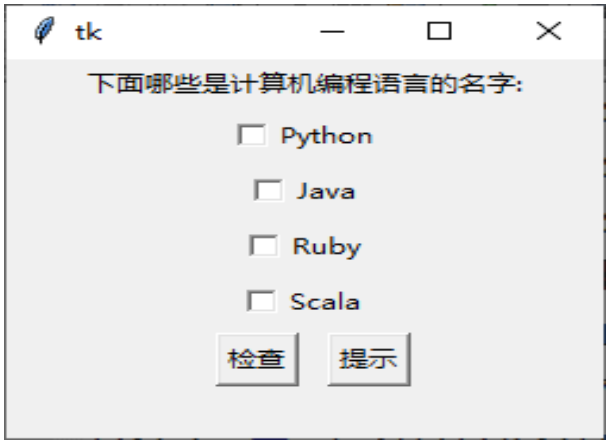


图12-9 test_checkbutton.py执行效果

12.3.5 Checkbutton

```
01 # -*- coding: utf-8 -*-
02 # 例12-6：test_checkbutton.py
03 from tkinter import *
04 app = Tk()
05 options = [IntVar() for _ in range(4)]#每个复选框所绑定的控制变量
06 def check():
07       for opt in options:
08             if (opt.get()!=1): #正确答案是每个候选项都应该被选择
09                   info['text']="请再想想!"
10                   return
11       info['text']="你真棒!"
12 def hint():
13       cb1.select() # 选择相应的复选框
14       cb2.select()
15       cb3.select()
16       cb4.select()
```

12.3.5 Checkbutton

```
17 Label(app,width=35,text="下面哪些是计算机编程语言的名字:").pack()
18 cb1 = Checkbutton(app,variable=options[0],text="Python")
19 cb1.pack()
20 cb2 = Checkbutton(app,variable=options[1],text="Java")
21 cb2.pack()
22 cb3 = Checkbutton(app,variable=options[2],text="Ruby")
23 cb3.pack()
24 cb4 = Checkbutton(app,variable=options[3],text="Scala")
25 cb4.pack()
26 frm = Frame(app) # 创建一个框架用于容纳后面的按钮控件，将在12.3.8介绍
27 frm.pack() # 对框架进行布局
28 Button(frm,text="检查",command=check).pack(side=LEFT,padx=2)
29 Button(frm,text="提示",command=hint).pack(side=LEFT,padx=2)
30 info = Label(app,width=10,text="")
31 info.pack(pady=2)
32 app.mainloop()
```

12.3.5 Checkbutton

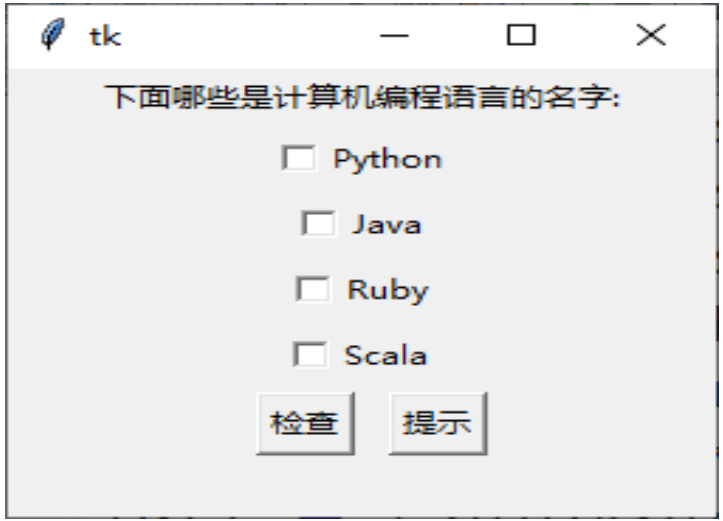


图12-9 test_checkbutton.py执行效果

12.3.6 Radiobutton

•Radiobutton表示单选按钮控件。与复选按钮类似，单选按钮也是向用户提供选项的控件，在外观上是一个小圆框加上相邻的描述性标题，其中的圆框在未选中时里面为空白，选中时会出现一个小圆点。但与复选按钮不同的是，单选按钮被选中后，再次单击时不会改变其选中状态。

•实际上，单选按钮通常是多个一起出现，以表示一组互斥的选项，这组单选按钮共享一个绑定的控制变量，当选中某个选项时，自动取消同一组中原先被选中的选项。Radiobutton的属性与方法与Checkbutton基本一致，只是Radiobutton没有onvalue和offvalue两个属性，而以一个value属性表示选中时控制变量的取值，没选中就是其它值。

12.3.6 Radiobutton

```
01 # -*- coding: utf-8 -*-
02 # 例12-7：test_radiobutton.py
03 from tkinter import *
04 app = Tk()
05 favorite = IntVar()
06 favorite.set(-1) # -1不同于任何选项的value，表示默认没有选项被选中
07 languages = ['Python','Java','Ruby','Scala']
08 def greet():#单选按钮的回调函数，通过共享控制变量的值获取已选择的选项
09     info['text']="你是一个{}控".format(languages[favorite.get()])
```

如果在单选按钮中选中了“java”，则会显示信息“你是一个Java控”。

favorite.get()获得的是一个整数值，代表了列表的下标，如果下标是0，就是列表的第1个元素“Python”

```
10 Label(app,width=35,text="你最喜欢的计算机编程是:").pack()
```

12.3.6 Radiobutton

```
11 frm = Frame(app) # 创建一个框架用于容纳后面的单选按钮控件
12 frm.pack()
13 for i, language in enumerate(languages):#创建多个单选按钮，共享同一个variable,但value的值不同
```

这里用到了enumerate() 函数，enumerate() 函数用于将一个可遍历的数据对象(如列表、元组或字符串)组合为一个索引序列，同时列出数据和数据下标，一般用在 for 循环当中。比如，这里使用enumerate() 函数对languages列表进行遍历，遍历第1次时，会把下标0赋值给变量i，把列表languages的第1个元素'Python'赋值给变量language，遍历第2次时，会把下标1赋值给变量i，把列表languages的第2个元素'Java'赋值给变量language，依此类推。

```
14     Radiobutton(frm, text=language,  variable=favorite, value=I, command=greet).pack(side=LEFT)
15 info = Label(app,text="")
16 info.pack()
17 app.mainloop()
```

12.3.6 Radiobutton

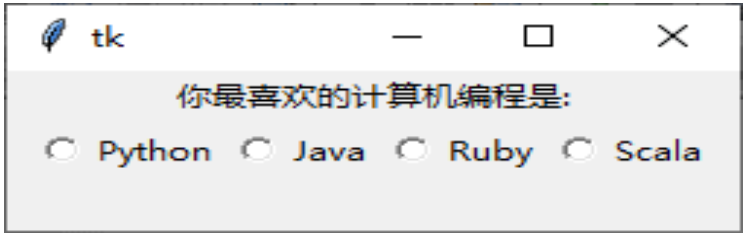


图12-10 test_radiobutton.py执行效果

12.3.7 Listbox

•Listbox表示列表框控件。主要用于较多相关项的选择。Listbox将这些相关的选项包含在一个多行文本框内，每一行代表一个选项。根据需求，可以设置成单选或者多选。

•不同于上面的checkbutton和Radiobutton，Listbox没有command属性，即默认没有绑定单击事件。

12.3.7 Listbox

Listbox的主要属性和方法有：

•get(index): 返回指定索引对应的选项文本；

•selectmode：选择模式。可选取值包括4个预定义的常量，分别是BROWSE（默认值）、SINGLE、MULTIPLE和EXTENDED。BROWSE和SINGLE都表示只能单选，但前者支持按住鼠标左键上下移动选择；类似地，EXTENDED和MULTIPLE都表示多选，前者支持鼠标框选，而后者只能一个个地选择。具体实例可以参见后面的test_listbox.py；

12.3.7 Listbox

•curselection()：返回当前所有被选中的各项索引（行号）构成的元组；

•insert(index,*elements):在指定索引index对应的选项后插入一个或多个新的选项，常量END表示末尾位置；

•selection_set (first, last=None): 选择从索引first到last之间所有的选项，且不改变已有的选择；

•selection_clear(first, last=None)：取消从索引first到last之间所有已选择的选项。

12.3.7 Listbox

```
01 # -*- coding: utf-8 -*-
02 # 例12-8：test_listbox.py
03 from tkinter import *
04 app = Tk()
05 selectmode = IntVar()
06 modes = [BROWSE,SINGLE,MULTIPLE,EXTENDED]#表示不同选择模式
07 days = ["Monday", "Tuesday", "Wednesday", "Thursday","Friday",
"Saturday", "Sunday"]
08 def check(): # 按钮的回调函数
09      selected = options.curselection() # 取得选择的索引号
10      info['text']= "你选择的是:"+str([options.get(i) for i in selected])
11 def change_mode(): # 单选按钮的回调函数
12      options['selectmode']=modes[selectmode.get()]#改变选择模式
13      options.selection_clear(0,END) # 清除已有的所有选择
```

12.3.7 Listbox

```
14 options=Listbox(app,selectmode=modes[0])# 创建一个空的列表框
15 options.pack()
16 options.insert(END,*days)#在列表框中添加选项
17 frm = Frame(app) # 创建一个框架用于容纳后面的单选按钮控件
18 frm.pack()
19 Label(frm,text="selectedmode:").pack(side=LEFT)
20 for i, mode in enumerate(modes):
21     Radiobutton(frm, text=mode, variable=selectmode, value=i, command
=change_mode).pack(side=LEFT)
22 Button(text="check my selections",command=check).pack()
23 info = Label(app,text="")
24 info.pack()
25 app.mainloop()
```

12.3.7 Listbox

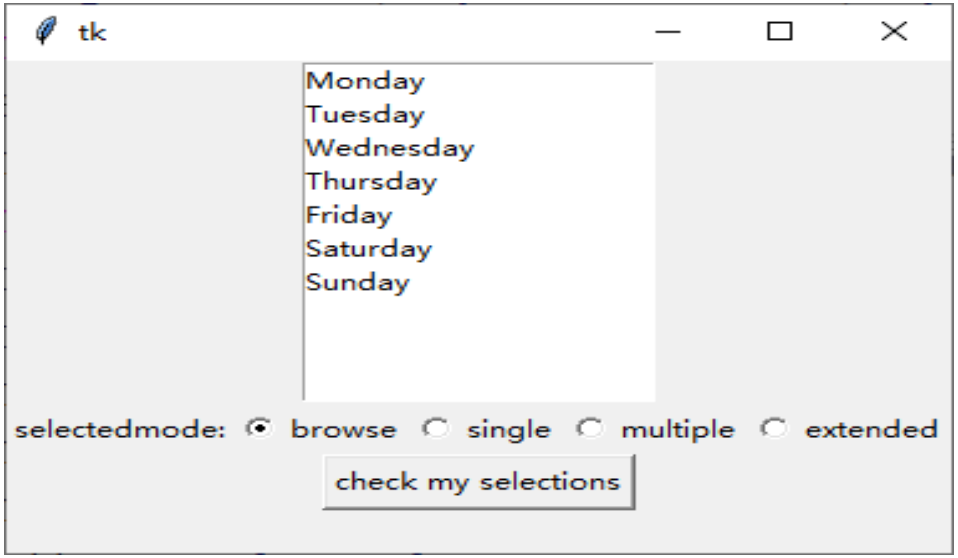


图12-11 test_listbox.py的执行效果

12.3.8 Frame/LabelFrame

•Frame和LabelFrame表示框架和标签框架控件。这两个控件都是容器类的控件，主要用途是作为父窗口对前面讲的各种非容器类控件进行组织，以便于界面的布局。

•两个控件的使用几乎一样，只需要调用相应的构造器，指定父窗口，并进行合适的布局。一旦创建好后，其作用就类似前面所有例程中Tk对象对应的根窗口，只需要将相关子控件的父窗口设置为Frame或者LabelFrame对象，并进行布局即可。

12.3.8 Frame/LabelFrame

•两个控件唯一的区别是，Frame控件的边框relief属性默认值为FLAT，使得其在外观上是不可见的，而LabelFrame控件的relief属性默认值GROOVE，同时在左上方的边框线上还显示了一个由text属性设置的文本。

•利用框架和标签框架对窗口进行层次化的组织，就可以便捷地建立布局更加美观的图形界面了。

•在下例中（为了代码简洁，省略了事件处理程序部分），建立了两个LabelFrame分别容纳用户输入框和复选按钮，并且在第一个LabelFrame内再建立两个Frame，便于用简单的Pack布局进行对齐。不难发现，通过引入框架容器，可以使得界面层次更加清晰。如果结合后文的Grid布局管理器，将可以进行更加复杂界面的布局。

12.3.8 Frame/LabelFrame

```
01 # -*- coding: utf-8 -*-
02 # 例12-9：test_label_frame.py
03 from tkinter import *
04 app = Tk()
05 # 基本信息录入
下面设置一个标签框架，框架左上角会显示4个字——“基本信息”
06 frame_inf= LabelFrame(app,padx=60, pady=5,text="基本信息")
07 frame_inf.pack(padx=10, pady=5)
08 frame_name=Frame(frame_inf) # 生成一个包含姓名信息的子框架
09 frame_name.pack()
下面设置一个标签，文本内容是“姓名”
10 Label(frame_name, text="姓名").pack(side=LEFT,padx=3)
下面设置一个输入文本框
11 Entry(frame_name,width=15).pack(side=LEFT,padx=3)
12 frame_ph = Frame(frame_inf) # 设置一个包含电话信息的子框架
13 frame_ph.pack()
```

12.3.8 Frame/LabelFrame

```
下面设置一个标签，文本内容是“电话”
14 Label(frame_ph, text="电话").pack(side=LEFT,padx=3)
下面设置一个输入文本框
15 Entry(frame_ph,width=15).pack(side=LEFT,padx=3)
16 # 特长选择
下面设置一个标签框架，框架左上角会显示2个字——“特长”
17 frame_spec = LabelFrame(app, padx=5, pady=5,text="特长")
18 frame_spec.pack(padx=10, pady=5)
下面设置4个复选框
19 Checkbutton(frame_spec,text="篮球").pack(side=LEFT,padx=5)
20 Checkbutton(frame_spec,text="足球").pack(side=LEFT,padx=5)
21 Checkbutton(frame_spec,text="乒乓球").pack(side=LEFT,padx=5)
22 Checkbutton(frame_spec,text="排球").pack(side=LEFT,padx=5)
23 # 提交
24 Button(app, text="提交").pack(padx=10, pady=10)
25 app.mainloop()
```

12.3.8 Frame/LabelFrame

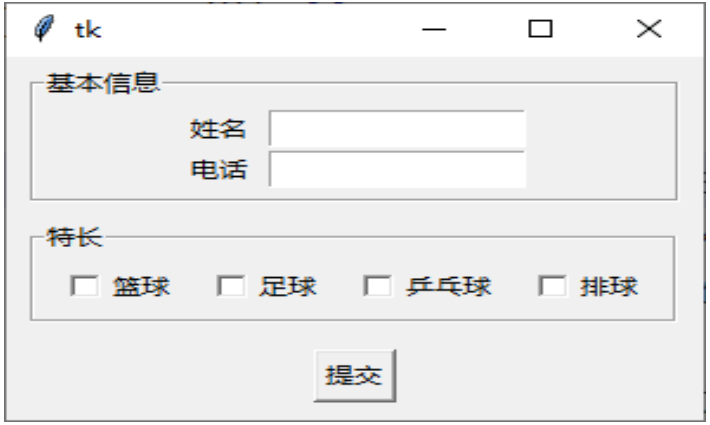
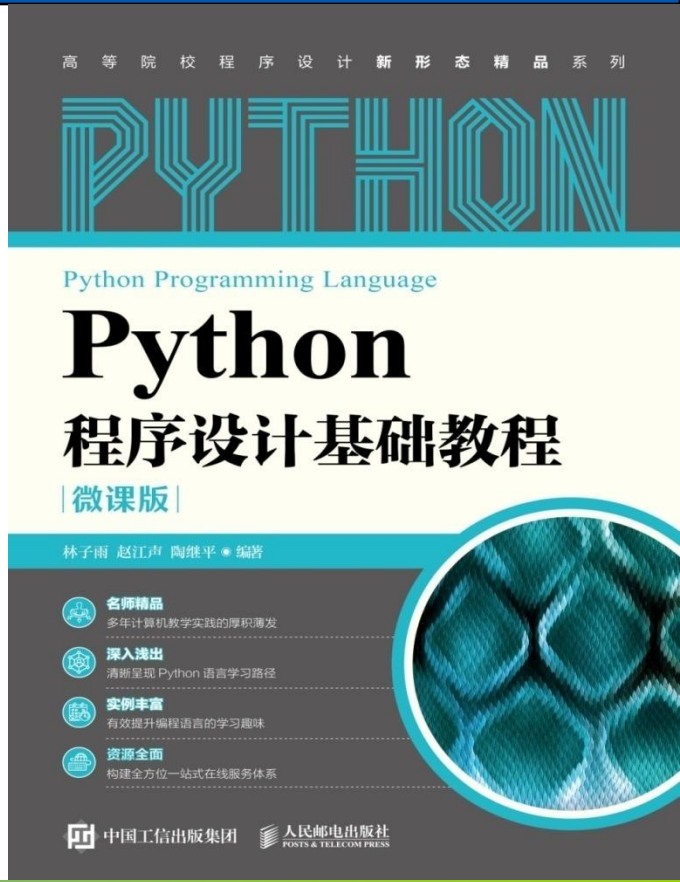


图12-12 test_label_frame.py的执行效果

12.4 tikinter中的布局管理

12.4.1 Pack布局
12.4.2 Grid布局
12.4.3 Place布局



本PPT是如下教材的配套讲义：
《Python程序设计基础教程（微课版）》

厦门大学 林子雨,赵江声,陶继平 编著，人民邮电出版社

《 Python程序设计基础教程（微课版） 》教材官方网站：
http://dblab.xmu.edu.cn/post/python

12.4.1 Pack布局

•Pack是一种基于顺序关系的布局管理。

•对于同一个容器窗口，Pack布局管理器按照容器子控件调用pack方法的顺序维护了一个布局顺序列表，布局时，将按照这个顺序列表依次将控件放入容器的某一侧，同时另一侧的剩余空间作为下一个控件的容器。

•例如，如果前一个控件沿左侧放置，则下一个控件只能放置在剩余的右侧空间；如果前一个控件沿上侧放置，则下一个控件只能放置在剩余的下侧空间。

12.4.1 Pack布局

pack方法的常用参数包括：

•**side**：表示沿着容器窗口剩余空间的哪条边放置控件，取值选项包括预定义常量TOP（默认值）、BOTTOM、LEFT、RIGHT，分别表示上侧、下侧、左侧和右侧。

•**padx, pady**: 在控件外围四周保留的外边距，padx为水平方向边距，pady为垂直方向边距。默认值为0，单位为像素，也可以采用其它单位。

•**ipadx, ipady**:在控件边框和内容之间保留的内边距，其在视觉上的作用将和控件本身的padx和pady属性累加，但值互不影响。默认值及单位同上。

12.4.1 Pack布局

•**anchor**：类似Label中的anchor属性，表示控件在分配的空间中的放置方位，默认取值为CENTER，表示居中放置。其它值包括“NW”、“W”等。

•**fill**: 按照side属性在某侧放置控件后，如果这一侧的高度或宽度大于控件的高度或宽度，fill属性指定是否在相应方向拉伸控件。取值选项包括预定义常量NONE (默认值，不拉伸)、X (沿着水平方向拉伸)、Y (沿着垂直方向拉伸)以及BOTH (沿着水平垂直两个方向拉伸)。

•**expand**：按照side属性在某侧放置控件后，是否允许控件放大以填充容器的另一侧空间，默认为False。当多个控件的expand都为True时，将由同方向放置的各个控件平分剩余空间，即沿LEFT和RIGHT的控件平分X方向的剩余空间，沿TOP和BOTTOM方向的控件平分Y方向上的剩余空间。

12.4.1 Pack布局

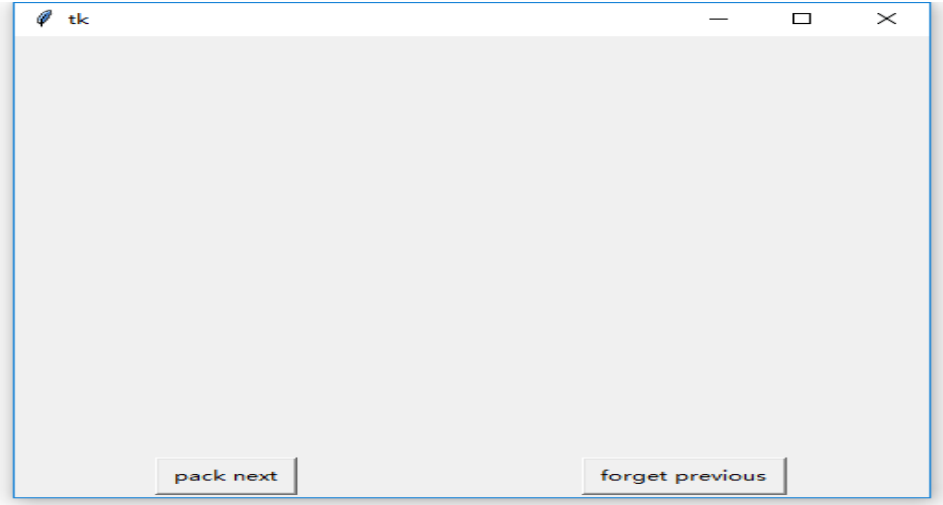
```
01 # -*- coding: utf-8 -*-
02 # 例12-10：test_pack.py
03 from tkinter import *
04 app = Tk()
05 frame = Frame(app)
06 frame.pack(fill=BOTH,expand=True)#允许框架随着主窗体大小自动拉伸
07 pading = {'padx': 2, 'pady': 2,'ipadx':10,'ipady':10}
08 A_label = Label(frame,text="Label A", bg="red" )
09 B_label = Label(frame,text="Label B", bg="green")
10 C_label = Label(frame,text="Label C", bg="blue")
11 D_label = Label(frame,text="Label D", bg="yellow")
12 E_label = Label(frame,text="Label E", bg="purple")
13 F_label = Label(frame,text="Label F", bg="pink")
14 labels = (A_label,B_label,C_label,D_label,E_label,F_label)
15 # 可以尝试调整下面三个参数的不同组合，测试不同的排版结果
16 sides=(TOP,TOP,LEFT,BOTTOM,RIGHT,TOP)#每个标签的放置方向不一样
17 fills=(BOTH,BOTH,BOTH,BOTH,BOTH,BOTH)
18 expands=(False,False,True,True,True,True)
```

12.4.1 Pack布局

```
19 i = 0
20 def pack_next(): # 回调函数
21     global i
22     if i<6: # 依次布局下一个标签
23         labels[i].pack(pading, side=sides[i], fill=fills[i], expand =expands[i])
24         i+=1
25 def forget_pre():# 回调函数
26     global i
27     if i>0: # 依次移除上一个标签
28         labels[i-1].forget()
29         i-=1
30 btn1 = Button(text="pack next",command=pack_next)
31 btn2 = Button(text="forget previous",command = forget_pre)
32 btn1.pack(pading,ipadx=5,ipady=2,side=LEFT,expand=True)
33 btn2.pack(pading,ipadx=5,ipady=2,side=LEFT,expand=True)
34 app.mainloop()
```

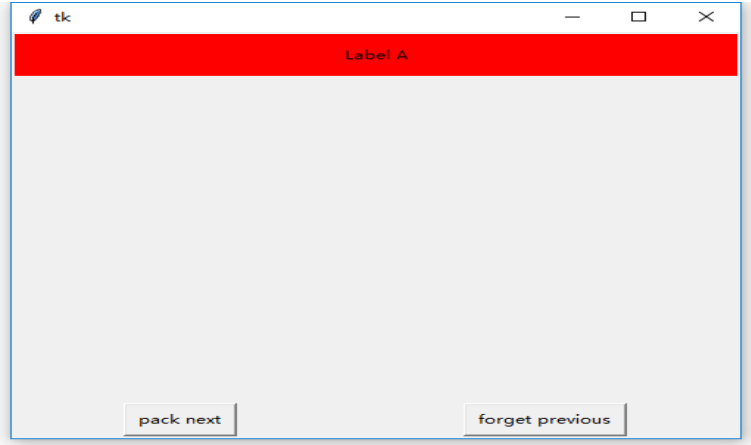
12.4.1 Pack布局

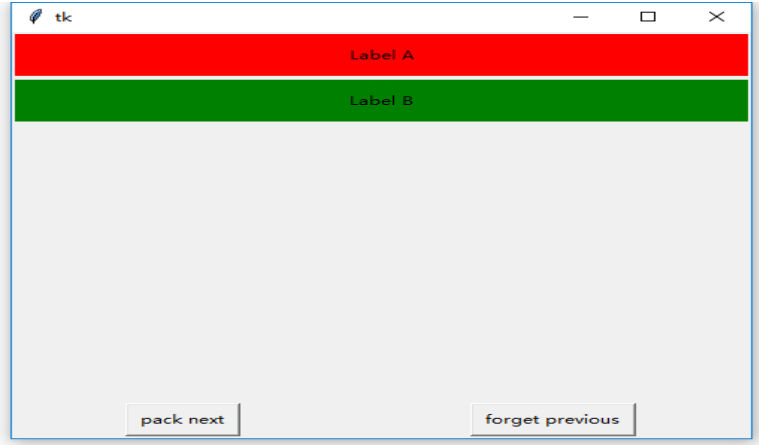
该程序创建了六个标签控件，其对应的side、fill及expand属性取值不同，打开程序后，读者可以先调整窗口到合适大小，再点击两个按钮一步步观察布局情况，其中按钮“pack next”开始布局下一个标签，按钮“forget previous”移除上一个标签。效果如这张图所示。



12.4.1 Pack布局

首先，标签A和B沿TOP放置，且fills为BOTH，因此将在X方向拉伸，但expand为false，因此不会填充下侧的剩余空间；效果如这两张图所示。





12.4.1 Pack布局

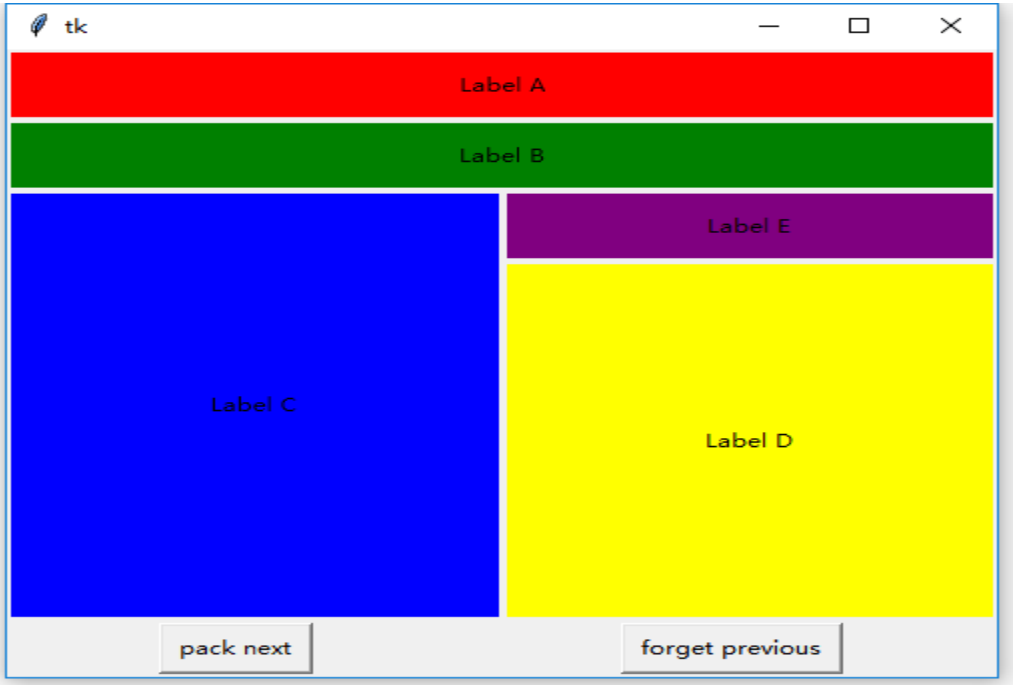
C标签沿LEFT放置，且fills为BOTH，expand为True，因此将占据剩余的所有空间；效果如这张图所示。

12.4.1 Pack布局

D标签沿BOTTOM放置，虽然fills为BOTH，expand为True，但X方向已被C填充，因此D只在Y方向拉伸填充；效果如这张图所示。

12.4.1 Pack布局

E沿RIGHT放置，fills为BOTH，expand为True，由于Y方向已经被D填充，因此只在X方向与C平分剩余空间；效果如这张图所示。



12.4.1 Pack布局

F沿TOP放置，fills为BOTH，expand为True，因此在Y方向将与D平分剩余空间。效果如这张图所示。

12.4.2 Grid布局

•Grid布局管理器将容器窗口按照行和列划分为纵横的二维表格，每一个单元格按行号和列号进行编号。

•布局控件时只需要指定相应的行号和列号。

•单元格的宽度由所在列中所有控件的最大宽度决定，单元格的高度由所在行中所有控件的最大高度决定。

•控件可以占据多个单元格。

12.4.2 Grid布局

grid方法的常用参数如下：

•**row/column**：控件在表格中的行号和列号。左上角的单元格对应第0行和第0列，默认值为0。如果一个单元格被多个控件指定，这些控件在视觉上可能将重叠，应避免这种情况。

•**padx/pady/ipadx/ipady**: 布局控件时的内外边距，同上节pack方法。

12.4.2 Grid布局

•**sticky**：确定控件在单元格中的放置方式。类似pack中的anchor参数，sticky的取值可以是N、NW等除CENTER以外表示方位的常量，如果不设置该参数，表示居中放置。另外，sticky还可以取这些方位的组合，这时候的作用就相当于在某个方向对控件进行拉伸。例如，N+S表示水平居中且沿着垂直方向拉伸控件，E+W表示上下居中且沿着水平方向拉伸，NW+S表示靠左并垂直拉伸。

•**rowspan/columnspan**: 表示在row/column对应的单元格基础上向下或向右合并多个单元格。例如w.grid(row=0, column=2, columnspan=3)表示将控件w放置在第0行和第2,3,4列。

12.4.2 Grid布局

默认情况下，采用Grid布局完成后，二维表格的大小不会随着容器的缩放而改变大小的，如果需要表格跟随容器缩放，需要调用容器的下列两个方法为各行各列设置缩放的权重。

•**rowconfigure(n,weight)**：设置指定行允许缩放，其中n为表示行号，weight为缩放时该行占的权重。

•**columnconfigure(n,weight)**：设置指定列允许缩放，其中n为表示列号，weight为缩放时该列占的权重。

12.4.2 Grid布局

下例是采用Grid布局完成图12-13的布局效果(如下图所示)，其将容器划分为4行3列。不难发现，相比Pack布局，采用Grid布局的程序更加简单清晰。

12.4.2 Grid布局

```
01 # -*- coding: utf-8 -*-
02 # 例12-11:test_grid.py
03 from tkinter import *
04 app = Tk()
05 pading = {'padx': 2, 'pady': 2,'ipadx':10,'ipady':10}
06 A_label = Label(app,text="Label A", bg="red" )
07 B_label = Label(app,text="Label B", bg="green")
08 C_label = Label(app,text="Label C", bg="blue")
09 D_label = Label(app,text="Label D", bg="yellow")
10 E_label = Label(app,text="Label E", bg="purple")
11 F_label = Label(app,text="Label F", bg="pink")
12 A_label.grid(pading, row=0, column=0, columnspan=3, sticky = NW+SE)
#占据第0行，第0-2列。sticky的取值表示拉伸以填充单元格。
```

12.4.2 Grid布局

```
13 B_label.grid(pading, row=1, column=0, columnspan=3, sticky = NW+SE) #占据第1行，第0-2列。
14 C_label.grid(pading, row=2, column=0, rowspan=2, sticky = NW+SE) #占据第2-3行，第0列。
15 D_label.grid(pading,row=3,column=1,columnspan=2,sticky=NW+SE) #占据第3行，第1-2列。
16 E_label.grid(pading,row=2,column=2,sticky= NW+SE) #占据第2行，第1列。
17 F_label.grid(pading,row=2,column=1,sticky= NW+SE) #占据第2行，第2列。
18 app.mainloop()
```

12.4.3 Place布局

Place布局将容器窗口看成一个原点在左上角的二维坐标系，并直接采用绝对坐标或相对坐标对控件进行精确定位。当容器窗口改变大小时，采用绝对坐标布局的控件位置固定不变，而采用相对坐标进行布局的控件将随之调整。place方法的基本使用很简单，只需要指定坐标和锚点即可，相关参数介绍如下：

•anchor：表示放置“锚点”，该点将与设定的坐标点对齐。取值是N、NW等值，默认为NW，表示左上角。
•x/y: 放置位置的绝对坐标，默认单位为像素，也可以采用其它单位，见12.3.1中关于单位的相关介绍。
•relx/rely: 放置位置的相对坐标，取值为0至1之间的数值，表示在容器窗口中的相对位置，如 relx=0.25 表示在容器宽度方向上的四分之一处。

12.4.3 Place布局

•对于大部分布局需求，上面的Grid布局管理器都能很好地胜任。

•如果还需要自由度更高的布局管理，可以采用Place布局。

•下例是Place布局的简单演示，其中Label A采用绝对定位，而Label B采用相对定位在窗口中间，可以调整窗口大小查看效果。

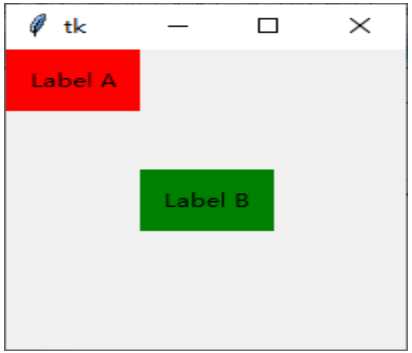


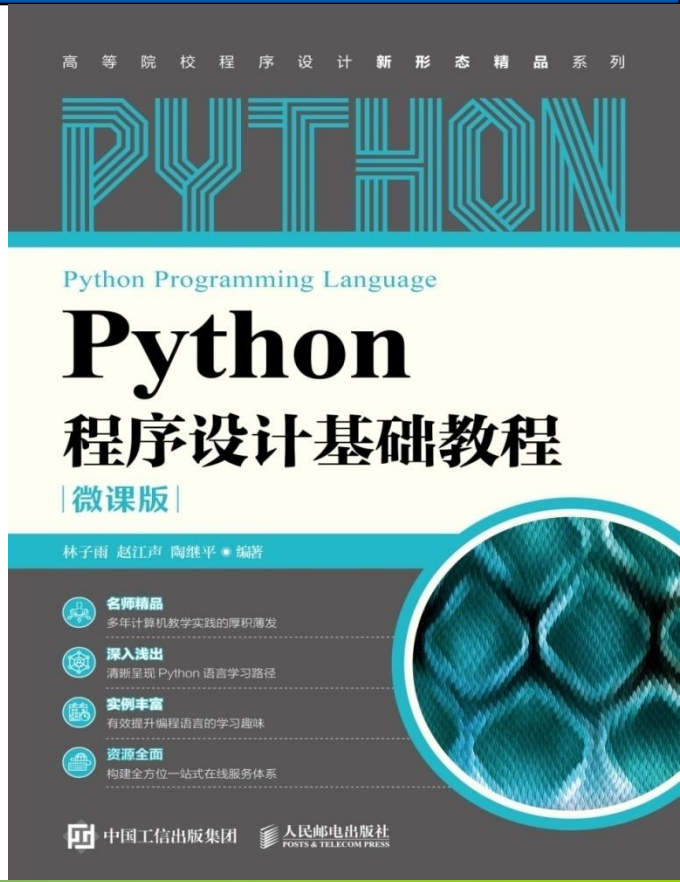
图12-14：test_place.py的执行效果

12.4.3 Place布局

```
01 # -*- coding: utf-8 -*-
02 # 例12-12:test_place.py
03 from tkinter import *
04 app = Tk()
05 pading = {'padx': 10, 'pady': 10}
06 A_label = Label(app,pading,text="Label A", bg="red" )
07 B_label = Label(app,pading,text="Label B", bg="green")
08 A_label.place(x=0,y=0) # 采用绝对坐标固定在窗口左上角
09 B_label.place(relx=0.5,rely=0.5,anchor=CENTER)# 采用相对坐标放置在容器中间
10 app.mainloop()
```

12.5 tkinter中的事件处理

12.5.1事件的表示
12.5.2事件处理程序的绑定



本PPT是如下教材的配套讲义：
《Python程序设计基础教程（微课版）》

厦门大学 林子雨,赵江声,陶继平 编著，人民邮电出版社

《 Python程序设计基础教程（微课版） 》教材官方网站：
http://dblab.xmu.edu.cn/post/python

12.5.1事件的表示

•tkinter采用事件模式标识符来表示不同种类的事件，事件模式描述符是一个形如“<modifier-type-detail>”的字符串。

•其中，type表示事件的一般类型，最常用的取值包括：Button（鼠标键按下）、ButtonRelease（鼠标键释放）、Key（键盘键按下）、KeyRelease（键盘键释放）、Enter（鼠标进入控件的可视化区域）、Motion（移动鼠标）等；
modifier是可选的修饰符，表示是否有一些组合键按下，常用取值为：Control/Alt/Shift（相应键被按下）、Double（事件连续两次快速发生）；
detail是可选的事件具体信息，例如1/2/3分别表示鼠标的左中右三个键，a表示键盘的a键被按下。各项完整的取值列表请参考Tk的官方文档。

12.5.1事件的表示

下面列出了一些最常用的事件描述符：
<Button-1>：按下鼠标左键；
< ButtonRelease-1>：释放鼠标左键；
<Button-3>：按下鼠标右键；
<Double-Button-1>：双击鼠标左键；
<Motion>：移动鼠标；
<Key>: 按下键盘任意键；
<Shift-Key-a>: 同时按下Shift和a键；
<Return>：按下回车键。

12.5.1事件的表示

一旦有事件发生，系统将自动实例化一个 Event类的对象，并将该对象传递给事件处理程序。事件对象包含了一些属性用于描述事件发生时的相关信息，常用属性包括：

- x/y：事件发生时，鼠标相对于控件左上角的位置坐标，单位是像素；
- x_root/y_root：事件发生时，鼠标相对于屏幕左上角的位置坐标，单位是像素；
- num：按下的鼠标键，1、2、3 分别表示左、中、右键；
- char：对于Key或KeyRelease事件类型，如果按下的是 ASCII 字符键，此属性即是该字符；如果按下特殊键，此属性为空。

12.5.2事件处理程序的绑定

包括Button、Checkbutton及Radiobutton在内的一些控件默认与鼠标左键单击事件绑定，对于这类控件只需要设定相应的command属性，即建立起了鼠标左键单击事件与相应事件处理程序之间的关系。如果要绑定非默认事件，就必须手动地在事件与事件处理程序之间建立绑定关系。tkinter提供了三种级别的绑定方法，分别是：

•控件对象级别的绑定：将某个控件对象上发生的特定事件与事件处理程序进行绑定。通过调用控件对象的bind方法实现绑定。基本用法为“控件对象.bind(事件模式描述符, 事件处理程序)”。例如，下述语句为控件对象w绑定了鼠标右键按下事件：

```
w.bind('<Button-3>',callback) #w是某个控件对象，callback为回调函数
```

12.5.2事件处理程序的绑定

•控件类级别的绑定：为某一控件类的所有实例上发生的特定事件与事件处理程序进行绑定。在任意控件对象上调用bind_class方法实现控件类级别绑定。基本语法为“控件对象.bind_class(控件类名,事件模式描述符,事件处理程序)”，例如，下述语句为Entry类绑定了Enter事件：

```
w.bind_class("Entry","<Enter>",callback) #w是某个Entry控件对象
```

•应用程序级别的绑定：为应用程序中的所有控件进行特定的事件绑定，在程序的任意一个控件对象上调用bind_all 方法实现应用程序级别的绑定。基本用法为“控件对象.bind_all(事件模式描述符, 事件处理程序)”。例如，下述语句在应用程序级别将F1功能键与一个回调函数绑定，当应用程序窗口处于活动状态时，按下F1键都将调用相应函数。

```
w.bind_all('<Key-F1>', callback) #w是某个控件对象
```

12.6 tkinter的综合应用案例

•在前面几节的实例代码中，由于涉及控件较少，我们都是在全局空间内操纵各个控件对象。

•在实际项目中，当涉及较多控件以及复杂的业务逻辑时，为了使得代码层次结构更清晰，一般都需要采用自定义类来进行组织。

•本节通过一个简单的计算器程序帮助大家进一步掌握tkinter的基本使用。该程序可以实现不带括号的四则运算，既可以用鼠标点击相应按钮完成计算，也可以直接使用键盘输入。

12.6 tkinter的综合应用案例

•在该例中，我们通过继承Tk类构建了一个自定义类Calculator。在Calculator的构造函数中完成了控件的设计与布局。

•整个界面采用了7*4的Grid布局，其中，第一行和第二行对应两个标签控件，第二个标签显示当前待计算的表达式，按回车键计算结束后，第一行显示计算完毕的表达式，第二行显示计算结果。

•后5行对应18个按钮控件，分别是0-9的数字按键、四折运算按键、小数点、退格键、清空键和回车键。在事件处理程序设计上，数字键、四折运算按键以及小数点键都采用一个匿名函数来调用一个统一的方法；退格键、清空键以及回车键的事件处理程序分别对应一个类方法。为了用户使用方便，该程序还在应用程序级别绑定了各个按钮对应的键盘事件。

•下面的calculator.py给出了程序代码，其执行效果如图12-15所示。

12.6 tkinter的综合应用案例

```
01 # -*- coding: utf-8 -*-
02 # calculator.py
03 import tkinter as tk
04 class Calculator(tk.Tk):
05     def __init__(self):
06         super().__init__() # 调用父类的构造器
07         self.title("Calculator")
08         self.iconbitmap("python.ico")
09         # 7*4布局，第一行和第二行是两个标签控件，后5行是18个按钮控件
10         opts={'padx': 2, 'pady': 2,'ipadx':3,'ipady':2, 'sticky':tk.NSEW}
11         buttonwidth=7
12         self.exp = tk.StringVar() # 输入表达式标签的控制变量
13         self.res = tk.StringVar(self,"0")# 计算结果标签的控制变量
14         exp_label=tk.Label(self,anchor=tk.E,textvariable=self.exp)#输入表达式标
签控件
15         exp_label.grid(opts,row = 0, column = 0, columnspan = 4)
```

12.6 tkinter的综合应用案例

```
16          res_label=tk.Label(self,anchor=tk.E,textvariable=self.res) #计算结果（开始计算前用于显示待计算表达式）标签控件
17          res_label.grid(opts,row = 1, column = 0, columnspan = 4)
18          tk.Button(self, text = "C", width=buttonwidth, command = self.clear).grid(opts, row = 2, column = 0)
19          tk.Button(self,text="/", width=buttonwidth, command = lambda:self.show("/")).grid(opts,row=2,column=1)
20          tk.Button(self,text="*",width=buttonwidth, command= lambda:self.show("*")).grid(opts,row=2,column=2)
21          tk.Button(self,text="BS",width=buttonwidth, command= self.backspace).grid(opts,row=2,column=3)
22          tk.Button(self,text="-",width=buttonwidth, command= lambda:self.show('-')).grid(opts,row=3,column=3)
23          tk.Button(self,text="+",width=buttonwidth, command= lambda:self.show('+')).grid(opts,row=4,column=3)
24          tk.Button(self,text="Enter",anchor=tk.S,width=buttonwidth, command = self.calculate).grid(opts,row=5,column=3,rowspan=2)
```

12.6 tkinter的综合应用案例

```
25              tk.Button(self,text=".",width=buttonwidth,
command=lambda:self.show('.')).grid(opts,row=6,column=2)
26              tk.Button(self,text="0",width=buttonwidth,
command=lambda:self.show('0')).grid(opts,row=6,column=0,columnspan=2)
27              tk.Button(self,text="7",width=buttonwidth,
command=lambda:self.show("7")).grid(opts,row=3,column=0)
28              tk.Button(self,text="8",width=buttonwidth,
command=lambda:self.show("8")).grid(opts,row=3,column=1)
29              tk.Button(self,text="9",width=buttonwidth,
command=lambda:self.show("9")).grid(opts,row=3,column=2)
30              tk.Button(self,text="4",width=buttonwidth,
command=lambda:self.show("4")).grid(opts,row=4,column=0)
31              tk.Button(self,text="5",width=buttonwidth,
command=lambda:self.show("5")).grid(opts,row=4,column=1)
32              tk.Button(self,text="6",width=buttonwidth,
command=lambda:self.show("6")).grid(opts,row=4,column=2)
33              tk.Button(self,text="1",width=buttonwidth,
command=lambda:self.show("1")).grid(opts,row=5,column=0)
```

12.6 tkinter的综合应用案例

```
34              tk.Button(self,text="2",width=buttonwidth,
command=lambda:self.show("2")).grid(opts,row=5,column=1)
35              tk.Button(self,text="3",width=buttonwidth,
command=lambda:self.show("3")).grid(opts,row=5,column=2)
36              # 允许除第一二行以外的各行各列等比例缩放
37              for i in range(2,7):
38                  self.rowconfigure(i,weight=1)
39              for i in range(0,4):
40                  self.columnconfigure(i,weight=1)
41              # 添加应用程序级别键盘输入事件
42              self.bind_all("<Return>",lambda e:self.calculate()) # 回车键
43              self.bind_all("<Key-BackSpace>",lambda e:self.backspace()) # 退格键
44              self.bind_all("<Key-Delete>",lambda e:self.clear()) # 删除键
45              self.bind_all("<Key-plus>",lambda e:self.show('+'))
46              self.bind_all("<Key-minus>",lambda e:self.show('-'))
47              self.bind_all("<Key-asterisk>",lambda e:self.show('*'))
48              self.bind_all("<Key-slash>",lambda e:self.show('/'))
49              self.bind_all(“<Key>”,self.check_key) # 其它数字及操作符
```

12.6 tkinter的综合应用案例

```
50
51  def check_key(self,event): # 检查数字键及操作符键事件
52          if (event.char>='0') and (event.char<="9"):
53                  self.show(event.char)
54
55  def calculate(self): # 调用eval函数计算表达式结果
56          res = eval(self.res.get()) # 计算当前的表达式
57          self.exp.set(self.res.get())
58          self.res.set(str(res))
59
60  def clear(self): # 清除当前的表达式
61          self.exp.set("")
62          self.res.set("0")
63
```

12.6 tkinter的综合应用案例

```
64   def show(self,key): # 在开始计算前将当前的输入添加到待计算表达式
65           content = self.res.get()
66                   if content == "0":
67                           content = ""
68                   self.res.set(content + key)
69
70   def backspace(self): # 输入回撤一位
71           self.res.set(str(self.res.get()[:-1]))
72
73 if __name__ == "__main__":
74           app = Calculator()
75           app.mainloop()
```

12.6 tkinter的综合应用案例

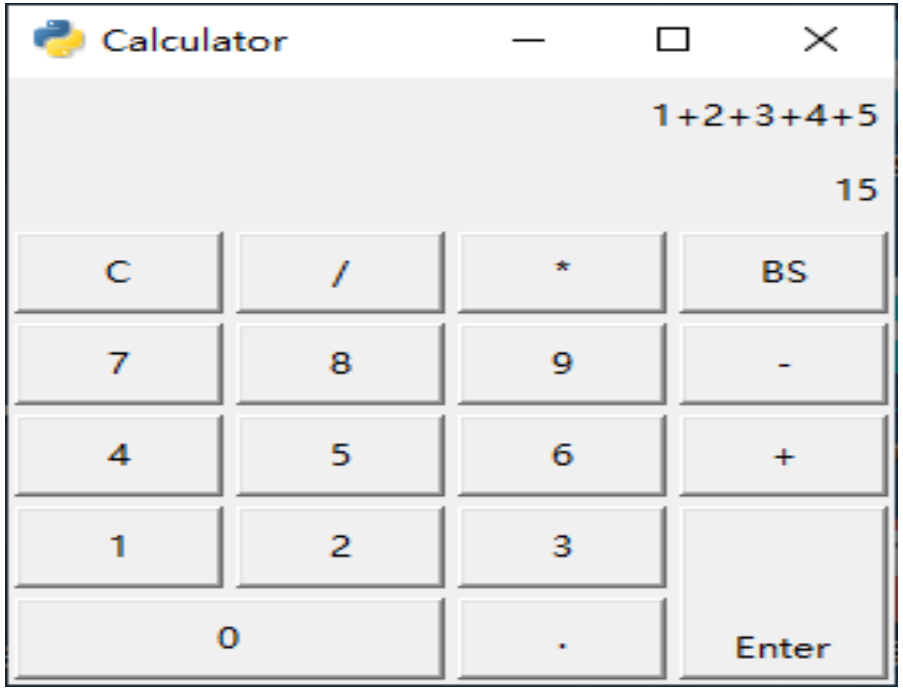


图12-15 calculator.py执行效果

附录A：主讲教师林子雨简介

主讲教师：林子雨

单位：厦门大学计算机科学与技术系
E-mail: ziyulin@xmu.edu.cn
个人网页：http://dblab.xmu.edu.cn/post/linziyu
数据库实验室网站：http://dblab.xmu.edu.cn

扫一扫访问个人主页

林子雨，男，1978年出生，博士（毕业于北京大学），全国高校知名大数据教师，现为厦门大学计算机科学系副教授，厦门大学信息学院实验教学中心主任，曾任厦门大学信息科学与技术学院院长助理、晋江市发展和改革局副局长。中国计算机学会数据库专业委员会委员，中国计算机学会信息系统专业委员会委员。国内高校首个“数字教师”提出者和建设者，厦门大学数据库实验室负责人，厦门大学云计算与大数据研究中心主要建设者和骨干成员，2013年度、2017年度和2020年度厦门大学教学类奖教金获得者，荣获2019年福建省精品在线开放课程、2018年厦门大学高等教育成果特等奖、2018年福建省高等教育教学成果二等奖、2018年国家精品在线开放课程。主要研究方向为数据库、数据仓库、数据挖掘、大数据、云计算和物联网，并以第一作者身份在《软件学报》《计算机学报》和《计算机研究与发展》等国家重点期刊以及国际学术会议上发表多篇学术论文。作为项目负责人主持的科研项目包括1项国家自然科学青年基金项目(No.61303004)、1项福建省自然科学青年基金项目(No.2013J05099)和1项中央高校基本科研业务费项目(No.2011121049)，主持的教改课题包括1项2016年福建省教改课题和1项2016年教育部产学协作育人项目，同时，作为课题负责人完成了国家发改委城市信息化重大课题、国家物联网重大应用示范工程区域试点泉州市工作方案、2015泉州市互联网经济调研等课题。中国高校首个“数字教师”提出者和建设者，2009年至今，“数字教师”大平台累计向网络免费发布超过1000万字高价值的研究和教学资料，累计网络访问量超过1000万次。打造了中国高校大数据教学知名品牌，编著出版了中国高校第一本系统介绍大数据知识的专业教材《大数据技术原理与应用》，并成为京东、当当网等网店畅销书籍；建设了国内高校首个大数据课程公共服务平台，为教师教学和学生学习大数据课程提供全方位、一站式服务，年访问量超过400万次，累计访问量超过1500万次。

附录B：大数据学习路线图

大数据学习路线图访问地址：http://dblab.xmu.edu.cn/post/10164/

附录C：林子雨大数据系列教材

林子雨大数据系列教材
用于导论课、专业课、实训课、公共课

了解全部教材信息：http://dblab.xmu.edu.cn/post/bigdatabook/

附录D：《大数据导论（通识课版）》教材

开设全校公共选修课的优质教材

本课程旨在实现以下几个培养目标：

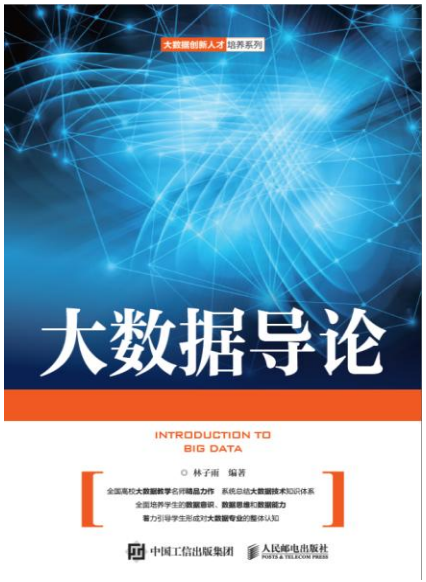
- 引导学生步入大数据时代，积极投身大数据的变革浪潮之中
- 了解大数据概念，培养大数据思维，养成数据安全意识
- 认识大数据伦理，努力使自己的行为符合大数据伦理规范要求
- 熟悉大数据应用，探寻大数据与自己专业的应用结合点
- 激发学生基于大数据的创新创业热情

高等教育出版社 ISBN:978-7-04-053577-8 定价：32元 版次：2020年2月第1版
教材官网：http://dblab.xmu.edu.cn/post/bigdataintroduction/

附录E：《大数据导论》教材

- 林子雨 编著《大数据导论》
- 人民邮电出版社，2020年9月第1版
- ISBN:978-7-115-54446-9 定价：49.80元

教材官网：http://dblab.xmu.edu.cn/post/bigdata-introduction/



开设大数据专业导论课的优质教材

扫一扫访问教材官网

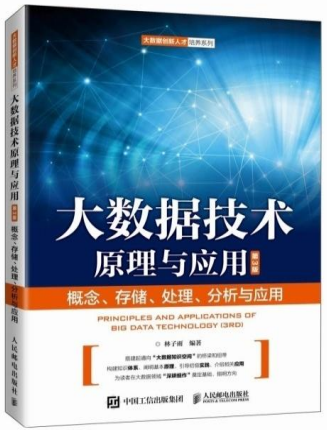
附录F：《大数据技术原理与应用（第3版）》教材

《大数据技术原理与应用——概念、存储、处理、分析与应用（第3版）》，由厦门大学计算机科学系林子雨博士编著，是国内高校第一本系统介绍大数据知识的专业教材。人民邮电出版社 ISBN:978-7-115-54405-6 定价：59.80元

全书共有17章，系统地论述了大数据的基本概念、大数据处理架构Hadoop、分布式文件系统HDFS、分布式数据 库HBase、NoSQL数据库、云数据库、分布式并行编程模型MapReduce、Spark、流计算、Flink、图计算、数据可视化以及大数据在互联网、生物医学和物流等各个领域的应用。在Hadoop、HDFS、HBase、MapReduce、Spark和Flink等重要章节，安排了入门级的实践操作，让读者更好地学习和掌握大数据关键技术。

本书可以作为高等院校计算机专业、信息管理等相关专业的大数据课程教材，也可供相关技术人员参考、学习、培训之用。
欢迎访问《大数据技术原理与应用——概念、存储、处理、分析与应用》教材官方网站：
http://dblab.xmu.edu.cn/post/bigdata3

扫一扫访问教材官网



附录G：《大数据基础编程、实验和案例教程（第2版）》

本书是与《大数据技术原理与应用（第3版）》教材配套的唯一指定实验指导书

大数据教材

1+1黄金组合

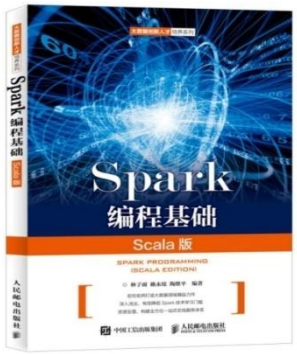
厦门大学林子雨编著

配套实验指导书

- 步步引导，循序渐进，详尽的安装指南为顺利搭建大数据实验环境铺平道路
- 深入浅出，去粗取精，丰富的代码实例帮助快速掌握大数据基础编程方法
- 精心设计，巧妙融合，八套大数据实验题目促进理论与编程知识的消化和吸收
- 结合理论，联系实际，大数据课程综合实验案例精彩呈现大数据分析全流程

林子雨编著《大数据基础编程、实验和案例教程（第2版）》
清华大学出版社 ISBN:978-7-302-55977-1 定价：69元 2020年10月第2版

附录H：《Spark编程基础（Scala版）》



《Spark编程基础（Scala版）》

厦门大学 林子雨，赖永炫，陶继平 编著

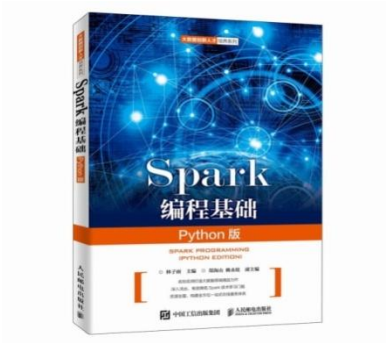
披荆斩棘，在大数据丛林中开辟学习捷径
填沟削坎，为快速学习Spark技术铺平道路
深入浅出，有效降低Spark技术学习门槛
资源全面，构建全方位一站式在线服务体系

人民邮电出版社出版发行，ISBN:978-7-115-48816-9

教材官网：http://dblab.xmu.edu.cn/post/spark/

本书以Scala作为开发Spark应用程序的编程语言，系统介绍了Spark编程的基础知识。全书共8章，内容包括大数据技术概述、Scala语言基础、Spark的设计与运行原理、Spark环境搭建和使用方法、RDD编程、Spark SQL、Spark Streaming、Spark MLlib等。本书每个章节都安排了入门级的编程实践操作，以便读者更好地学习和掌握Spark编程方法。本书官网免费提供了全套的在线教学资源，包括讲义PPT、习题、源代码、软件、数据集、授课视频、上机实验指南等。

附录I：《Spark编程基础（Python版）》



《Spark编程基础（Python版）》

厦门大学 林子雨，郑海山，赖永炫 编著

披荆斩棘，在大数据丛林中开辟学习捷径
填沟削坎，为快速学习Spark技术铺平道路
深入浅出，有效降低Spark技术学习门槛
资源全面，构建全方位一站式在线服务体系

人民邮电出版社出版发行，ISBN:978-7-115-52439-3

教材官网： http://dblab.xmu.edu.cn/post/spark-python/

本书以Python作为开发Spark应用程序的编程语言，系统介绍了Spark编程的基础知识。全书共8章，内容包括大数据技术概述、Spark的设计与运行原理、Spark环境搭建和使用方法、RDD编程、Spark SQL、Spark Streaming、Structured Streaming、Spark MLlib等。本书每个章节都安排了入门级的编程实践操作，以便读者更好地学习和掌握Spark编程方法。本书官网免费提供了全套的在线教学资源，包括讲义PPT、习题、源代码、软件、数据集、上机实验指南等。

附录J：高校大数据课程公共服务平台

高校大数据课程

公 共 服 务 平 台

http://dblab.xmu.edu.cn/post/bigdata-teaching-platform/

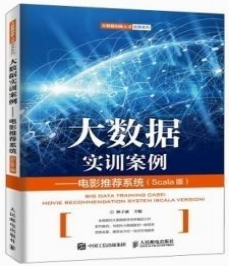
扫一扫访问平台主页

扫一扫观看3分钟FLASH动画宣传片

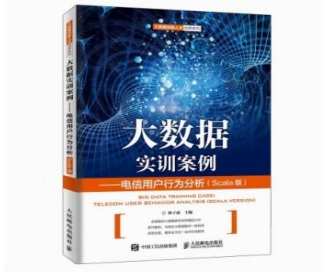
附录K：高校大数据实训课程系列案例教材

为了更好满足高校开设大数据实训课程的教材需求，厦门大学数据库实验室林子雨老师团队联合企业共同开发了《高校大数据实训课程系列案例》，目前已经完成开发的系列案例包括：

《电影推荐系统》（已经于2019年5月出版）
《电信用户行为分析》（已经于2019年5月出版）
《实时日志流处理分析》
《微博用户情感分析》
《互联网广告预测分析》
《网站日志处理分析》



系列案例教材将于2019年陆续出版发行，教材相关信息，敬请关注网页后续更新！
http://dblab.xmu.edu.cn/post/shixunkecheng/



扫一扫访问大数据实训课程系列案例教材主页


Thank You!