



北京航空航天大学  
BEIHANG UNIVERSITY

# 大学计算机基础 (理科类)

## 第13讲 图形用户界面设计

北京航空航天大学





# 第13讲 图形用户界面设计

13.1 图形用户界面与Tkinter

13.2 Tkinter常用组件的使用方法

13.3 组件的几何布局



# 本讲重点和难点

## 重点

- 了解使用**Tkinter**工具包设计**GUI**的一般方法
- 掌握Tkinter的**Label**、**Button**、**Check Button**、**Menu**、**Text**组件的使用
- 掌握组件的常用**几何布局**方法

## 难点

- Menu组件和Text组件的应用





北京航空航天大学  
BEIHANG UNIVERSITY

# 13.1 图形用户界面与Tkinter

北京航空航天大学



# 图形用户界面（GUI）

- 现在软件一般使用**图形用户界面**（Graphical User Interface, GUI）（始于1980年）实现与用户的交互和信息交换
- **GUI** 又称**图形用户接口**，是指采用**图形方式**显示的计算机操作环境用户接口

## ◆ 优点

- ✓ 简便易用
- ✓ 无需死记硬背大量的命令，只需通过**窗口**、**菜单**、**按钮**、**文本框**等方式进行操作，与软件进行交互

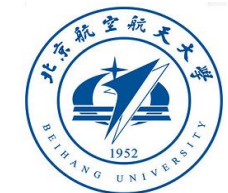
与**命令行界面**不同，GUI主要通过**鼠标**实现人与计算机的交互



# 1、GUI的组成

## ■ GUI由各种图形用户界面元素组成

- ◆ **桌面 (Desktop)** : 界面中最底层
- ◆ **视窗 (窗口)** : 启动一个应用程序对应一个窗口
- ◆ **标签 (标签页, 选项卡) (Tab)** : 同一类相关操作 (功能) 组织在同一个**标签页**中
- ◆ **菜单 (Menu)** : 以**层级** (菜单、子菜单) 的形式显示应用程序的**所有**命令
- ◆ **按钮 (Button)** : 以图形表示的菜单中**常用**命令
- ◆ **图标 (Icon)** : 表示应用**程序**或者某个**文档**或某个**功能的图形**





# 什么是控件（组件）？

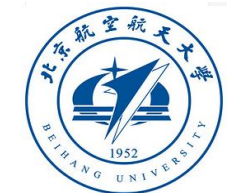
- **控件**（control，或部件、构件：widget，组件）是一种图形用户界面元素（如视窗、按钮、文本框等），控制着应用程序处理的所有数据以及关于这些数据的交互
- **控件是对数据和方法的封装**。当用户点击某个控件（如按钮）时，则访问该控件的**数据**，或者执行相应的**操作**（功能）
- 可以使用**现成的控件**来开发应用程序；也可以**创建控件**



## 2、常用控件及其用途

### ■ 常用控件及其用途

- ◆ **标签-Label**，用于显示提示性的文字内容或图像，**不支持用户操作**
- ◆ **按钮-Button**，支持鼠标按下操作，**触发**某特定行为。例如“确定”，“取消”
- ◆ **单选按钮-Radio Button**，表示与用户交互的**互斥**选项。例如“性别”中“男”、“女”
- ◆ **复选按钮-Check Button**，表示与用户交互的**多选项**，用户通过点击按钮可以选择一个或多个选项。例如“选课”中的各门课程





## 常用控件及其用途（续）

- ◆ **文本框-Text (Editor)**，与用户交互简单输入输出内容，用户可以在其中**输入**一行或多行内容
- ◆ **菜单-Menu**，程序操作核心区，提供应用程序的**所有**操作命令
- ◆ **列表-List**，以**列表**形式显示多条数据，支持选中、单击、双击等操作
- ◆ **工具栏-Tool Bar**，将菜单中的常用功能以**图标**形式显示，多位于菜单下方，方便用户操作



## ■ Tkinter简介

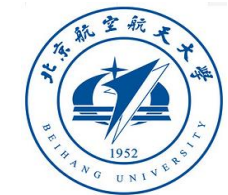
- ◆ Python内置的标准GUI库，可以快速创建GUI应用程序
- ◆ 可以在**Windows、UNIX和Macintosh**系统下使用
- ◆ 提供Python GUI设计所需的各种控件，如Frame、Label、Button、Text等
- ◆ 提供一个主要的类**Widget**（构件），Frame、Button、Text等都是Widget类的**子类**

- 注意：根据命名规范，**类名一般首字母大写，函数名一般全部小写**。**Frame、Text、Button等并不是函数**，它们是类Widget的子类。因为Python中创建对象不需要使用new关键词，所以看起来都是在调用函数



# 利用Tkinter设计GUI的一般步骤

- 利用Tkinter设计GUI的一般步骤
  - ◆ (1) 导入Tkinter模块  
`from tkinter import *`
  - ◆ (2) 创建顶层窗口  
`top = Tk()` # Tk构造器
  - ◆ (3) 创建组件, 指定这个组件的master, 即这个组件属于哪一个父窗口 (父组件)  
`t = Text(top,height=10, width=40)`
  - ◆ (4) 利用pack或grid方法将组件放置到GUI的合适位置  
`t.pack()`  
或 `t.grid(row=1,column=1)`
  - ◆ (5) 建立一个主循环, 使程序可以长期运行, 以等待用户的输入和操作  
`mainloop()` #主循环





# Tkinter的主要组件

序号	组 件	描 述
1	Button	按钮组件。用于显示各种按钮，当按钮被按下时，执行某种操作。Button 组件可以包含文本或图像
2	Checkbutton	复选框组件。又称为 <b>多选</b> 按钮，用于提供多个可能同时被选择的选项。用户通过点击按钮可以选择一个或多个选项
3	Entry	输入框组件。用于获取用户的输入文本。但仅允许输入 <b>一行</b> 文本。若希望接收多行文本的输入，则可以使用Text组件
4	Label	标签组件。用于显示提示性的 <b>文字内容</b> 或 <b>图像</b> 。只能以同一种风格显示文本，文本可以是一行或多行，也可以显示位图或图像



# Tkinter的主要组件（续）

序号	组 件	描 述
5	Menu	菜单组件。用于实现 <b>顶级</b> 菜单、 <b>下拉</b> 菜单和 <b>弹出</b> 菜单
6	Radiobutton	<b>单选</b> 按钮组件。用于为用户提供多个选项，但用户只能选择其中的一个
7	Scrollbar	滚动条组件。用于滚动一些组件（如 <b>文本框</b> 、 <b>画布</b> 和 <b>列表框</b> ）的 <b>可见</b> 范围，当内容超过可视化区域时使用。根据滚动方向可分为垂直滚动条和水平滚动条
8	Text	文本组件。用于显示 <b>一行</b> 或 <b>多行</b> 文本，文本可以是纯文本，也可以是格式化文本；还经常用作简单的文本编辑器和网页浏览器使用



北京航空航天大学  
BEIHANG UNIVERSITY

# 13.2 Tkinter常用 组件的使用方法

北京航空航天大学



## 13.2 Tkinter常用组件的使用方法

- ◆ 13.2.1 用Label组件创建标签
- ◆ 13.2.2 用Button组件创建普通按钮
- ◆ 13.2.3 用Checkbutton组件创建复选框
- ◆ 13.2.4 用Menu组件创建菜单
- ◆ 13.2.5 用Text组件创建文本框

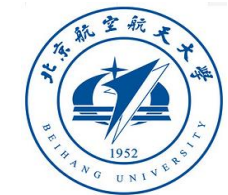
自学





## 13.2.1 用Label组件创建标签

- **Label**：**标签**组件，用于在GUI中输出提示性的**文字**内容或**图像**
  - ◆ 只能以**同一种风格**显示文本，文本可以是一**行**或多**行**，也可以显示**位图**或**图像**
  - ◆ 还可以为文本中的某个或某些字符加上**下划线**（如用来表示键盘快捷键）







# Label的用法

格式 **Label ( *master, options* )**

## ■ Label的参数

### ◆ 输入参数

**master**: 父窗口

**options**: 可选参数，共有**27**个参数，常用的参数有

✓ **text** (表示Label上显示的文本)

✓ **compound** ( = '**left**' : 指明Label在文字的左边, = '**right**' : 右边, = '**top**' : 上面, = '**bottom**' : 下面)





# Label的用法（续）

- ✓ **font** (字体大小)
- ✓ **bitmap** (设置Label上的图片)
- ✓ **anchor** (设置Label上内容在组件中的位置)
- ✓ **textvariable** (设置控制变量)：指定Label**显示**与之绑定的**Tkinter变量**（通常是一个StringVar变量）的**内容**。如果Tkinter变量的值改变，则Label上的文本随之更新
- ✓ **relief** (设置按钮被按下后释放时的外观)：默认值是**FLAT**。可设置为**SUNKEN**（凹下），**RAISED**（凸起），**GROOVE**（槽）或**RIDGE**（隆起）

用于使标签上文字动态变化

各参数的具体含义见教材表5-18

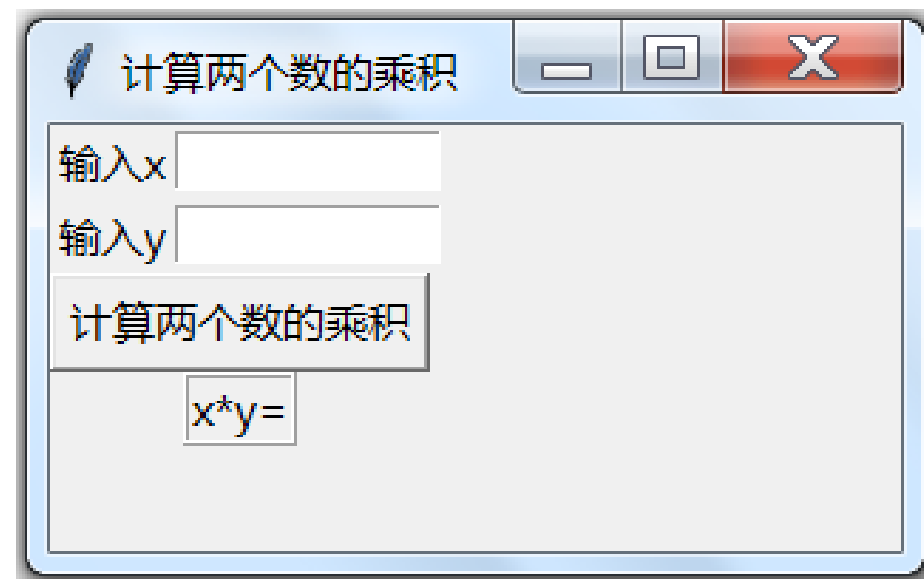




## 【例13.1】 Label 组件的高级应用

当你希望在Label上显示的文字随某个变量的变化而变化时，怎么办呢？

**【例13.1】** 设计一个GUI，计算两个数的乘积。可以通过**文本框**输入任意数，当单击“**计算**”**按钮**时，**标签**上可以显示乘数、被乘数和乘积。





## 【例13.1】设计思路

### ■ 设计思路

(1) 导入Tkinter库: `from tkinter import *`

(2) 调用**Tk构造器**创建一个顶层窗口: `top=Tk()`

使用Tk中的**geometry**函数指定顶层**窗口的大小**

**`top.geometry('300x150')`** #指定窗口宽度x高度

(3) 创建Label组件 `label_x=Label(top,text="输入x")` #提示输入x

创建文本框, 创建按钮

(4) 使用pack方法进行组件的布局

(5) 创建主循环`mainloop()`





## 【例13.1】设计难点

### ■ 关键：如何使标签上文字随输入和计算结果的不同而变化

- ◆ 创建一个Tkinter变量（字符串） var
- ◆ 创建一个标签，其textvariable参数与var绑定
- ◆ 采用**set方法**设置var的初值为 “x\*y=”

标签上文本随var的变化而变化

```
var = StringVar() #创建Tkinter变量（字符串），以便与textvariable绑定
display_label = Label( top, textvariable=var, relief=RAISED )
#参数textvariable使标签上的文本由控制变量var来控制
var.set("x*y=")      #set方法设置变量var的初值
```



## 【例13.1】设计难点（续）

- ◆ 然后在按钮cal\_button的回调函数中，通过**set方法**重新设置变量**var的值**，则var的值是随x、y以及 $x*y$ 的变化而变化的：

```
def calcuBtn(textbox_x,textbox_y):
```

```
    x = textbox_x.get(1.0,END)      #获得textbox_x的从1.0位置到末  
    尾的文本（回车符也被提取）
```

```
    y = textbox_y.get(1.0,END)
```

```
    z = float(x)*float(y)
```

```
    var.set('x='+str(x)+'y='+str(y)+'x*y='+str(z))
```



# 【例13.1】程序

## 例13.1–Label【计算乘积】.py

```
top.geometry("300x150")
```

#指定窗口宽度x高度【比较有此语句时的窗口大小】

#（1）添加组件

#创建标签

```
label_x=Label(top, text="输入x") #提示输入x
```

```
label_y=Label(top, text="输入y") #提示输入y
```

#创建文本框

```
textbox_x = Text(top, height = 1, width = 10) #用于输入x。只有一行，height设置为1即可
```

```
textbox_y = Text(top, height = 1, width = 10) #用于输入y
```

#创建按钮

```
cal_button=Button(top, text="计算两个数的乘积", command=lambda:calcuBtn(textbox_x, textbox_y))
```

#创建标签

```
var = StringVar()
```

#创建Tkinter变量（字符串），以便与textvariable绑定

```
display_label = Label( top, textvariable=var, relief=RIDGE )
```

#参数textvariable使标签上的文本由一个控制变量var来控制；参数relief设置标签边界的外观（relief=RIDGE（隆起））

```
var.set("x*y=")
```

#set方法设置变量var的初值



## 【例13.1】程序（续）

**grid\_configure()** 方法的作用和使用方法与**grid()**方法一样

# (2) 组件布局。使用grid\_configure函数将组件添加到主窗口上

```
#label_x.grid_configure(column = 1, row = 1, columnspan = 1, rowspan = 1)#第1行, 第1列, 占1列
label_x.grid(row = 1, column = 1, rowspan = 1, columnspan = 1)#第1行, 第1列, 占1列。grid方法与grid
label_y.grid_configure(row = 2, column = 1, rowspan = 1, columnspan = 1)#第2行, 第1列, 占1列
textbox_x.grid_configure(row = 1, column = 2, rowspan = 1, columnspan = 4)#第1行, 第2列, 占4列
textbox_y.grid_configure(row = 2, column = 2, rowspan = 1, columnspan = 4)#第2行, 第2列, 占4列
cal_button.grid_configure(row = 3, column = 1, rowspan = 1, columnspan = 4)#第3行, 第1列, 占4列
display_label.grid_configure(row = 4, column = 1, rowspan = 1, columnspan = 4)#第4行, 第1列, 占4列
```

# (3) 定义cal\_button的回调函数, 读取x、y, 计算两个数的乘积

```
def calcuBtn(textbox_x, textbox_y):
```

```
    x = textbox_x.get(1.0, END)
```

#获得textbox\_x的从1.0位置到末尾的文本（回车符也被提取）

```
    y = textbox_y.get(1.0, END)
```

```
    print("x is: ", x)
```

```
    print("y is: ", y)
```

```
    z = float(x)*float(y)
```

```
    print("x*y=", z)
```

```
    var.set("x="+str(x)+"y="+str(y)+"x*y="+str(z)) #set方法设置变量var的值。str(x)不要连接"\n", 因为x中已经包含了回车符
```

```
    #var.set("x="+str(x)+"\n"+"y="+str(y)+"\n"+"x*y="+str(z))#【str(x)连接"\n", 则在算式后面空了一行, 没有必要】
```

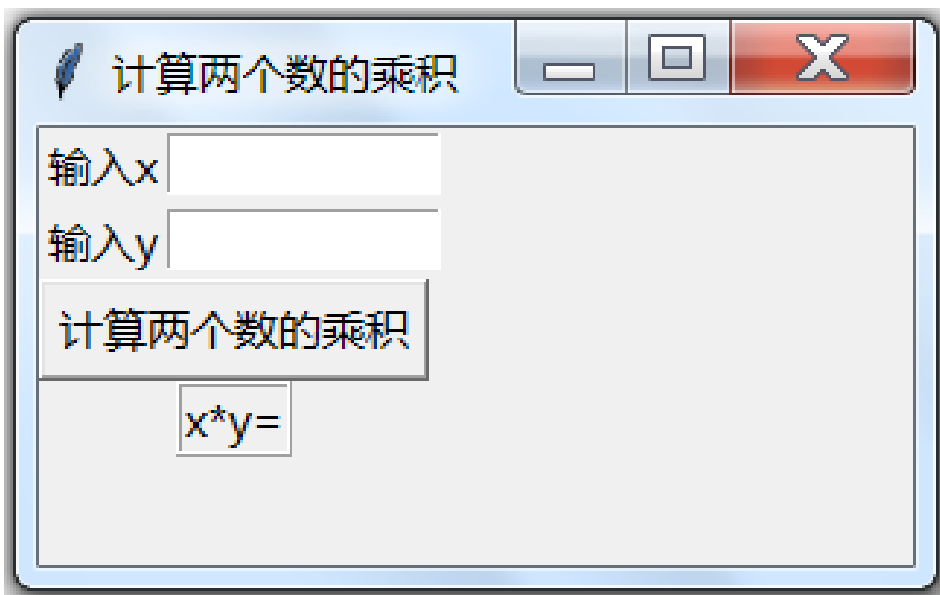
```
top.mainloop()
```

或**mainloop()**

当单击按钮时, 使  
标签上文字变化



## 【例13.1】程序运行结果



计算两个数的乘积

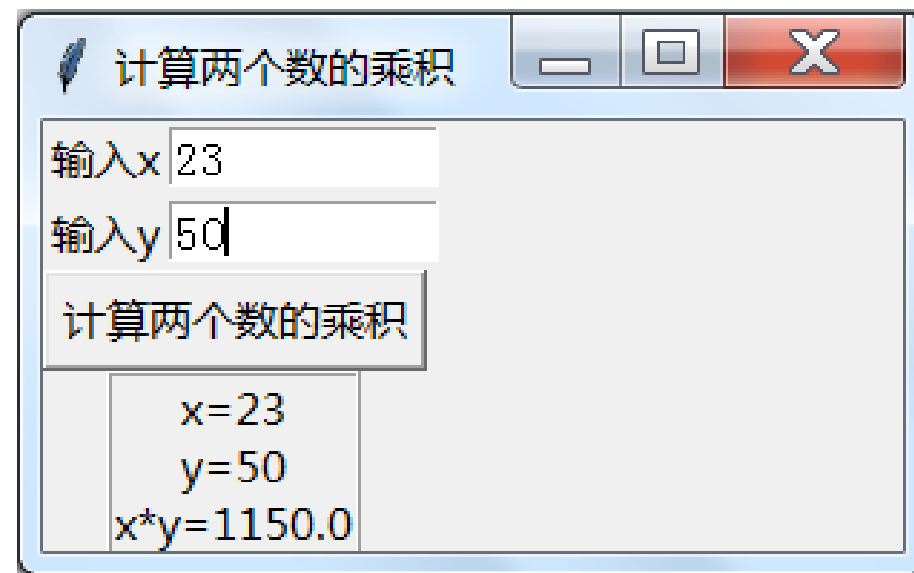
输入x

输入y

计算两个数的乘积

$x*y=$

(a) 初始界面



计算两个数的乘积

输入x

输入y

计算两个数的乘积

$x=23$   
 $y=50$   
 $x*y=1150.0$

(b) 结果界面



## 【课后练习1】

- 如果只需输入一行文本，采用**Entry组件**更合适！

**【课后练习】** 请尝试改写【例13.a1】，使用**Entry组件**代替**Text组件**，体会二者的区别。



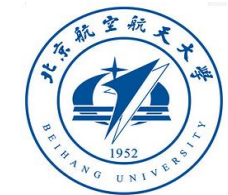


## 13.2.2 用Button组件创建普通按钮

■ **思考**：按钮有什么用途？

- **Button**：**按钮**组件，用于在GUI中添加按钮
- 按钮可以显示**文本**或**图像**，以表达**按钮被按下后**所执行的**操作**
  - ◆ 可以对按钮**绑定一个函数或方法**（称为**回调函数**），当单击按钮时，则自动调用该函数或方法
  - ◆ 同Label一样，Button只能以同一种风格显示文本，文本可以是一**行**或**多行**

但通常一行即可
  - ◆ 还可以为文本中的某个或某些字符加上下划线
  - ◆ 默认情况下，Tab按键用于在多个按钮之间切换



# Button的用法

## ■ Button的参数

格式 **Button ( *master, options* )**

***master***: 父窗口

***options***: 可选参数，共有**31**个参数，常用的参数有：

**text**表示Button上显示的文本（用单引号或双引号括起来），  
**command**指定回调函数，**font**设置字体大小，**bitmap**设置Button上的图片，**anchor**设置Button 上内容在组件中的位置，**fg**设置前景颜色，**bg**设置背景颜色等

不指定前景颜色和背景颜色时，Button的**前景**为**黑色**，**背景**为**灰色**





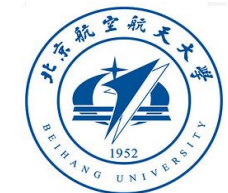
## Button的用法（续）

- ◆ 其中**27**个参数与Label完全相同，各参数的具体含义如**教材表5-18**所示

```
【例】 del_button = Button(top, text="delete",  
command = del_all, fg="red", bg="blue")  
# del_all为回调函数
```

- 文本和参数的值用**单引号**或**双引号**括起来都可以

- ◆ Button还有**4**个参数（**default**, **overrelief**, **repeatdelay**, **repeatinterval**, 如**教材表5-19**所示），这是Label所没有的



# 关于Button的几点说明

## ■ 最简单的按钮

- ◆ 只需指定按钮的**父窗口**、显示在按钮上的**内容**（文本、位图、图像）、当按钮被按下时的**回调函数**

*button1 = Button(master, text='OK', command=callback\_func)*

## ■ 回调函数的用途

- ◆ 当按钮被点击时，则调用该回调函数，执行相应的操作
- ◆ 回调函数使用“**def**”关键字来定义；**如果该函数有参数，必须一一列出（形参）**
- ◆ 然后在Button的command参数后面调用此函数时列出**实际参数（实参）**



# 关于Button回调函数的位置

注意!

## ■ 关于Button回调函数的位置

- ◆ 回调函数最好放在创建Button组件语句的**前面**定义，否则，程序运行后会提示**NameError**，提示该回调函数名尚未定义
- ◆ 如果希望将回调函数的定义放在创建Button组件语句的**后面**，则必须在创建Button时，在“command=”的后面、回调函数名称的前面加上“**lambda:**”，并在回调函数名后面必须加上一对**括号**（即使回调函数没有形参），否则单击按钮没有反应

```
button1=Button(top,text = 'Hello Button',command =  
lambda:helloButton(),relief= RAISED)
```

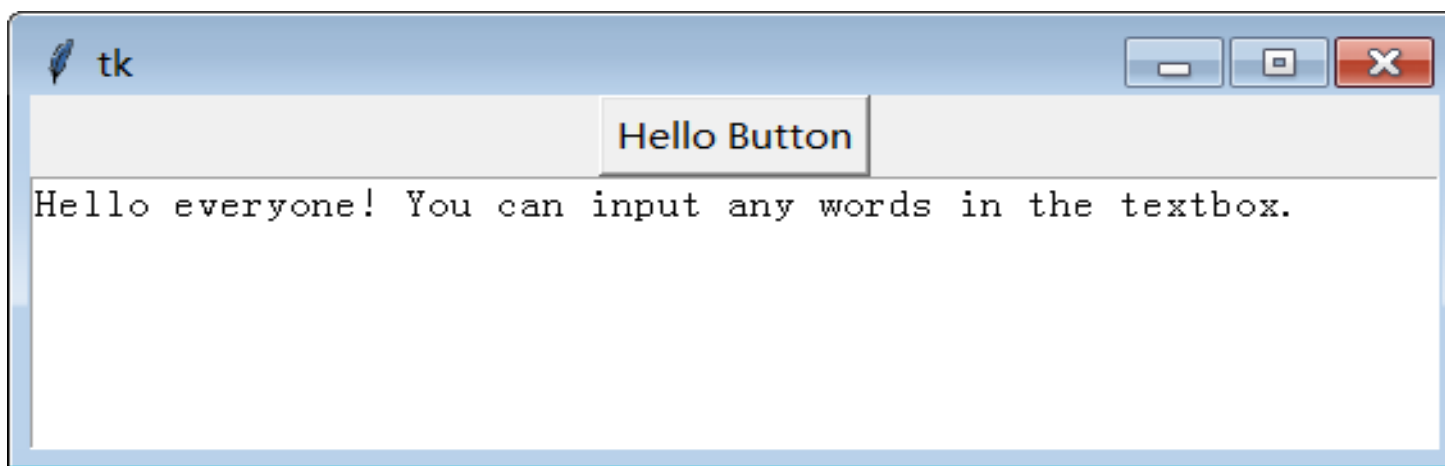
- ◆ 如果回调函数带有形参，则必须在括号中列出所有**形参**





## 【例13.2】 Button组件的简单应用

**【例13.2】** 设计一个GUI，其中包含一个文字为“Hello Button”的按钮和一个文本框。当按下此按钮时，在文本框中显示问候语。





## 【例13.2】程序

```
from tkinter import *  
top = Tk()
```

例13.2-Button.py

# (1) 定义按钮的回调函数

定义回调函数

```
def helloButton():
```

```
    print('hello')
```

```
    textbox.insert(1.0,"Hello everyone! You can input any words in the  
textbox.")
```

#在文本框中第1行开始的位置插入问候语

■ 回调函数**最好放在前面**，否则运行后提示：

NameError: name 'helloButton' is not defined

## 【例13.2】程序（续）

### # (2) 创建文本框，用于输入和输出文本

```
textbox = Text(top)
```

### # (3) 创建按钮

当按钮被点击时，调用这个函数

```
button1=Button(top,text = 'Hello Button',command = helloButton,  
relief= RAISED)
```

释放时凸起

#默认Button的前景为黑色，背景为灰色

### # (4) 组件几何布局

```
button1.pack()
```

```
textbox.pack()
```

```
top.mainloop()
```





## 13.2.3 用Checkbutton组件创建复选框

■ **Checkbutton：复选框** (checkbox) 组件，表示多个可能同时被选择的条件

自学!

- ◆ 又称为**多选按钮**，每个多选按钮有两种状态：On和Off，当选中按钮时为“ON”，不选中则为“OFF”
- ◆ 可以为多选按钮设置**回调函数**，每当单击该按钮时回调函数被调用
- ◆ 用户通过点击按钮可以选择一个或多个选项

- 当需要处理“**多选多**”问题时，可以将多个Checkbutton组件组合起来使用
- 处理“**多选一**”的问题，一般使用**Radiobutton**或**Listbox**组件





# Checkbox的用法

**格式** `Checkbox ( master, options )`

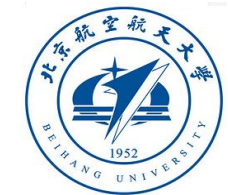
## ■ Checkbox的参数

***master***: 父窗口

***options***: 可选参数，共有**34**个参数，其中27个参数与Label完全相同。常用的参数有：

***variable***控制变量跟踪复选框状态（为**0**时表示**未勾选**复选框，为**1**时表示**勾选**复选框），***text***设置复选框文字，***bitmap***设置Checkbox上图片，

***command***当改变复选框状态时调用回调函数





# Checkbox不同于Label和Button的参数

序号	参 数	含 义
1	<b>command</b>	--指定与该按钮相关联的函数或方法 -- <b>当改变复选框状态</b> 时由Tkinter自动 <b>调用回调函数</b> --如果不设置此选项，则该按钮被按下后没有任何操作
2	<b>variable</b>	--将Checkbox跟一个Tkinter变量（一般是 <b>整型</b> 变量）关联 --当按钮 <b>被选中或不勾选</b> 时，该变量的值在 <b>1和0</b> 之间切换，或者在onvalue和offvalue所设置的值之间切换 --此切换过程是完全自动的

- variable参数是一个控制变量，用来**跟踪复选框的状态**。默认情况下，variable为**1**表示**选中**状态，为**0**表示**未选中**状态
- 也可以通过设置onvalue和offvalue的值，来自定义选中和未选中状态下variable的值（如为 “T” 和 “F” ）





# Checkbutton组件的简单应用

## ■ Checkbutton组件的简单应用

- ◆ Checkbutton组件通过**创建一个Tkinter变量**（一般是**整型**变量），并将其**与variable参数绑定**来跟踪按钮的状态
- ◆ Checkbutton组件为了实现多选功能，必须确保一组中的**每个按钮的variable参数使用不同的变量**
- ◆ 可利用**command**参数指定当复选框状态改变时调用某个**回调函数**

```
var1 = IntVar()
```

#创建一个整型变量

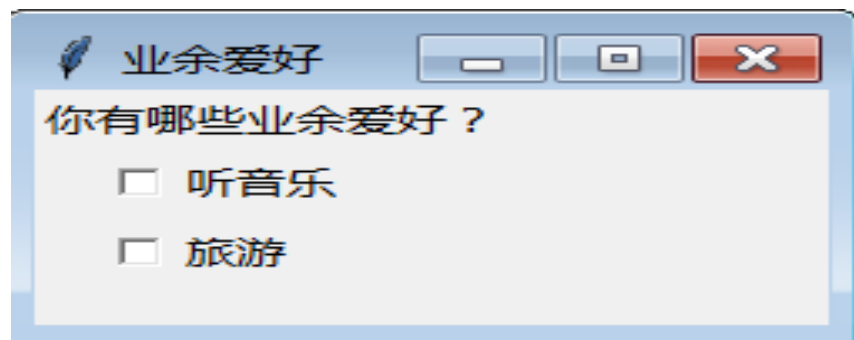
```
var2 = IntVar()
```

```
c1 = Checkbutton(top, text="我是教师", variable=var1, command=callCheckbutton)
```

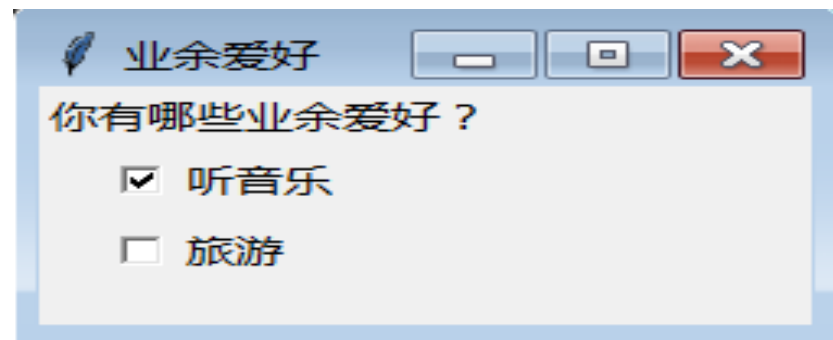
```
c2 = Checkbutton(top, text="我是医生", variable=var2, command=callCheckbutton)
```

## 【例13.3】 创建复选框示例

**【例13.3】** 设计一个简单的GUI，采用Checkbox组件实现对业余爱好的多选操作。当按钮被选中或未选中时，打印按钮当前状态下的值。



(a) 初始界面



(b) 结果界面

```
>>> ===== RESTART =====  
>>>  
checkvar1 is 1  
checkvar2 is 0
```



## 【例13.3】 程序

### 例13.3-checkbutton.py

```
from tkinter import*
```

```
top=Tk()
```

```
top.title('业余爱好')
```

```
top.geometry('200x120')
```

回调函数

```
def callCheckbutton():
```

```
    print ("checkvar1 is", checkvar1.get())
```

```
    print ("checkvar2 is", checkvar2.get())
```

#创建一个顶层窗口

#顶层窗口名称

#指定窗口宽度x高度

#定义回调函数，跟踪按钮状态

#get方法获得变量当前时刻的值

```
label=Label(top,text="你有哪些业余爱好? ")
```

#添加一个label，显示题目

```
checkvar1=IntVar()
```

#创建Tkinter整型变量，以便与variable绑定

```
checkvar2=IntVar()
```





## 【例13.3】程序（续）

#添加2个checkboxbutton

c1=Checkbutton(top,text="听音乐

当复选框状态改变时，调用这个函数

",**variable=checkvar1**,command=callCheckbutton)

c2=Checkbutton(top,text="旅游

",**variable=checkvar2**,command=callCheckbutton)

每个按钮的variable参数  
使用不同的变量

label.pack(anchor=W)

c1.pack(anchor=W,padx=20) #左对齐，与顶层窗口的左边框相距20个像素)

c2.pack(anchor=W,padx=20)

top.mainloop()



## 13.2.4 用Menu组件创建菜单

■ **菜单**中列出了一个应用程序的所有操作命令

■ **常用菜单形式**

- ◆ **顶级菜单** (top menu) —— **最顶层**的菜单 (**主菜单**)
  - ◆ **下拉菜单** (pull-down menu) —— 应用程序刚启动后, 只显示顶级菜单 (主菜单); 当用户单击某个顶级菜单时, 则弹出该顶级菜单下的所有子菜单
  - ◆ **弹出菜单** (pop-up menu) —— **快捷菜单**。当用户在程序界面上单击鼠标**右键**时, 则显示出弹出菜单 (包含一些**最常用命令**)
- ✓ 弹出菜单使用**方便快捷**, 平时它**不占据**GUI的任何**空间**





# Menu组件的用法

■ **Menu：菜单组件**，用于实现应用程序上的各种菜单

■ **Menu的参数**                      格式 **Menu ( *master, options, ...* )**

*master*: 父窗口

*options*: 可选参数，**可以不设置**。一共有**15个**参数，如果没有特殊的要求，这些参数甚至一个都不会用到

**常用的参数：**

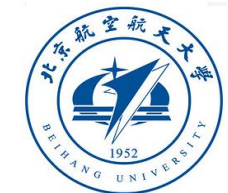
- ◆ **activebackground**: 当某选项被选中时显示的背景颜色
- ◆ **bg**: 那些未被选中的选项显示的背景颜色
- ◆ **font**: 文本选项的默认字体





# Menu组件更多的是使用它的各种方法

- 与Label、Button、Radiobutton和Checkbutton组件不同，Menu组件更多的是使用它的各种**方法**
- MENU组件拥有**22**种方法
- 一些**常用**的方法（10个）见**教材表5-23**





# Menu组件的常用方法

- ◆ **add(type, \*\*options)**: 添加一个指定类型的新元素, 包括 cascade (级联菜单)、checkboxbutton (复选框按钮)、command (命令)、radiobutton (单选按钮) 或 separator (分隔线) **作为菜单的一个可选项**
- ◆ **add\_command**: 添加一个普通的命令菜单项 (**子菜单**)
- ◆ **add\_cascade(\*\*options)**: 添加**级联菜单**
- ◆ **add\_checkbutton(\*\*options)**: 添加一个多选按钮的**菜单项**
- ◆ **add\_separator(\*\*options)**: 添加一条**分隔线**
- ◆ **insert(index, itemType, \*\*options)**: 在index参数指定的位置插入**指定类型的菜单项**





# Menu组件的add方法

## ■ **add(type, \*\*options)**

- ◆ add方法用于**在菜单中添加一个指定类型的新元素**，作为菜单的一个可选项
- ◆ **type参数**指定添加的元素类型，**字符串**，其值可以是：cascade（级联菜单）、checkboxbutton（复选框按钮）、command（菜单命令）、radiobutton（单选按钮）或separator（分隔线）
- ◆ **options参数**用于设置菜单的属性，其选项及具体含义见**教材表5-24**

```
menubar.add('cascade',label = 'Language',menu =  
filemenu)    #在menubar中添加一个级联菜单filemenu
```





# 关于add方法

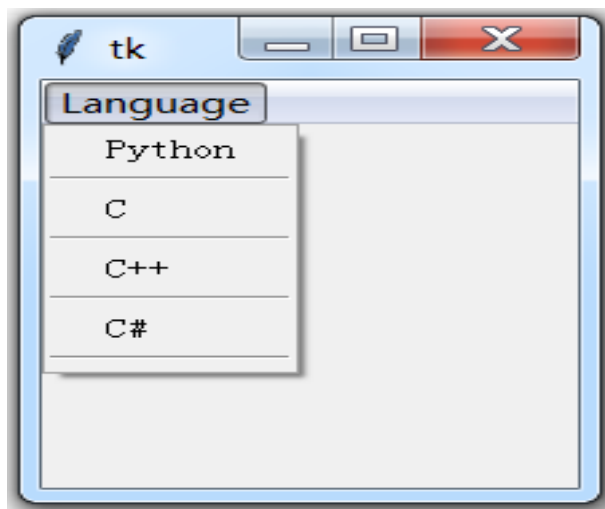
- ◆ 在add方法中，**当type参数取值为cascade、checkboxbutton、command、radiobutton或separator时**，其作用与**add\_cascade(\*\*options)、add\_checkboxbutton(\*\*options)、add\_command(\*\*options)、add\_radiobutton(\*\*options)、add\_separator(\*\*options)**方法完全相同

- 当需要添加一个级联（父）菜单、添加一个普通的命令菜单项（子菜单）或添加一条分隔线时，**建议直接使用相应的方法**，这样程序更清晰

- ◆ 如：直接使用add\_command(\*\*options)方法添加菜单命令

## 【例13.4】创建下拉菜单示例

**【例13.4】** 设计一个简单的GUI，采用Menu组件创建一个**下拉菜单**Language，它有若干子菜单项。每个子菜单都有相应的回调函数。







## 【例13.4】设计思路

### 设计思路

- ◆ 定义各子菜单的回调函数 (menu1~menu4)
- ◆ 使用**Menu**组件创建一个菜单实例作为**菜单栏**

**menubar = Menu(top)**

**top['menu'] = menubar**     #将top的menu属性设置为menubar

- ◆ 然后使用**Menu**组件创建**主菜单** (其父组件为刚创建的菜单栏)

**filemenu = Menu(menubar, tearoff = 0)**

**tearoff**: **分窗**, 0为在原窗, 1为点击虚线时分为两个窗口

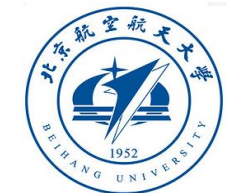
- ◆ 采用**for循环**: 使用**add\_command**方法添加所有**子菜单项**, 使用**add\_separator()**方法在每个 (或每组) 子菜单之间添加**分隔线**

**filemenu.add\_command(label = 'Python', command = menu1)**

**filemenu.add\_separator()**

- ◆ 最后使用**add\_cascade**方法在菜单栏添加一个**级联 (主) 菜单**

**menubar.add\_cascade(label = 'Language', menu = filemenu)**





## 【例13.4】程序

### 例13.4-pull-down\_menu.py

.....

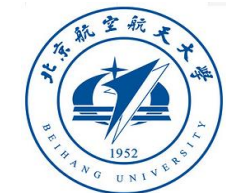
#### # (1) 定义4个子菜单的回调函数

```
def menu1():  
    print ('选择Python')  
def menu2():  
    print ('选择C')  
def menu3():  
    print ('选择C++')  
def menu4():  
    print ('选择C#')
```

#### # (2) 创建菜单栏

```
menubar = Menu(top)  
top['menu'] = menubar
```

#将top的menu属性设置为menubar





## 【例13.4】程序（续）

### # (3) 创建主菜单和添加子菜单项

```
c = [menu1, menu2, menu3, menu4] #4个子菜单的回调函数名放在一个列表中  
i = 0
```

```
filemenu = Menu(menubar, tearoff = 0) #创建主菜单，其父组件为菜单栏
```

```
for item in ['Python', 'C', 'C++', 'C#']:
```

```
    filemenu.add_command(label = item, command = c[i])
```

```
    filemenu.add_separator()
```

```
    i = i + 1
```

添加分隔线

添加子菜单项

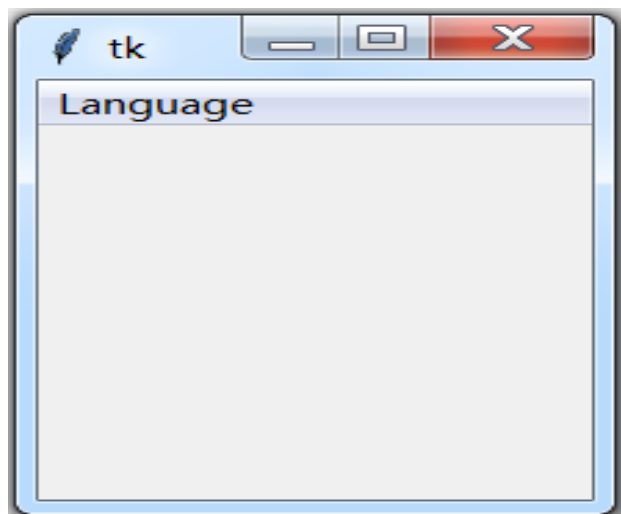
每个子菜单有不同的label和回调函数

### # (4) 在菜单栏添加一个级联（主）菜单

```
menubar.add_cascade(label = 'Language', menu = filemenu) #将menubar的menu属性指定为filemenu
```

```
top.mainloop()
```

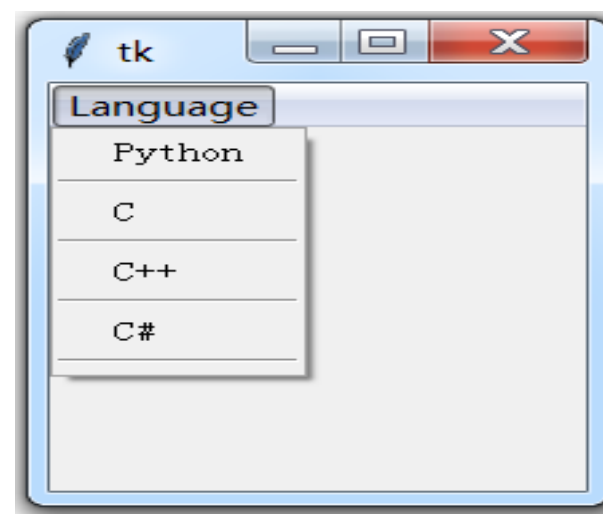
## 【例13.4】程序运行结果



当单击主菜单  
“Language” 时,



则**弹出下拉菜单**



当单击某个子菜单时，则**执行相应操作**——在Python窗口中打印对应的字符串

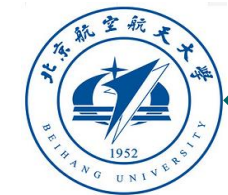
```
>>>  
选择Python  
选择C  
选择C++  
选择C#
```



## 13.2.5 用Text组件创建文本框

### 自学

- Tkinter中，**Entry**输入**单行**文本，Text处理多行数据
- **Text**是**文本**组件。用于创建文本框，使用户可以在文本框中**输入或编辑一行或多行**文本
  - ◆ 可以是**纯文本**，也可以是**格式化文本**（如采用不同字体，嵌入图片，显示链接，甚至是带CSS格式的HTML等）
  - ◆ 功能强大和灵活，主要用来输入和显示多行文本
  - ◆ 但也常被用作简单的**文本编辑器**或**网页浏览器**





# Text组件的用法

格式 ***Text ( master, options , ... )***

## ■ Text的参数

***master***: 父窗口

***options***: 可选参数，可以不设置。一共有**35**个参数。

◆ 如果没有特殊的要求，Text的参数甚至一个都不会用到

**Text的常用参数（共17个）参见教材表5-26**

■ 与Label、Button、Radiobutton和Checkbutton组件不同，**Text组件更多的是使用它的各种方法**





# Text的常用参数

- ✓ **height**设置文本框的高度（以**行数**表示，而不是像素值）
- ✓ **width**设置文本框的宽度（以**字符数**表示，而不是像素值）
- ✓ **borderwidth**（或**bd**）设置文本框边界的宽度（默认2个像素）
- ✓ **font**设置字体大小
- ✓ **background**（或**bg**）设置背景颜色

【例】 **expression1 = Text(frame, height = 4, width = 30, bd=4)**

#创建表达式输入窗口，父窗口为frame，用height、width设置界面大小，  
用bd设置文本框边界的宽度



# Text组件的常用方法

参见教材表5-27, 5-28

序号	方法	含义
1	<b>delete(start, end=None)</b>	--删除给定范围的文本或嵌入对象 --如果在给定范围内有任何Marks标记的位置, 则将Marks移动到start参数开始的位置
2	<b>edit_redo(self)</b>	-- “恢复” 上一次的 “撤销” 操作 --如果undo参数为False, 该方法无效
3	edit_separator()	--插入一个 “分隔符” 到存放操作记录的栈中, 用于表示已经完成一次完整的操作 --如果undo参数为False, 则该方法无效
4	<b>edit_undo()</b>	--撤销最近一次操作 --如果undo参数为False, 该方法无效
5	<b>get(index1, index2=None)</b>	--返回 (获取) index1到index2 (不包含) 之间的文本 --如果index2参数忽略, 则返回index1所指的那个字符 --如果文本框中包含image和window的嵌入对象, 则均被忽略 --如果包含有多行文本, 则返回的文本中自动插入换行符 ('\n')
6	image_create(index, cnf={}, **kw)	--在index参数指定的位置嵌入一个image对象。 该image对象必须是Tkinter的PhotoImage或BitmapImage实例





# Text组件的常用方法（续）

序号	方法	含 义
7	index(i)	--将i参数指定的位置以"line.column"的索引形式返回 --i参数支持任何格式的索引（如INSERT、END等）
8	<b>insert(index, text, *tags)</b>	--在index参数指定的位置插入文本。text指定要插入的字符串 --可选参数tags用于指定文本的样式
9	tag_add(tagName, index1, index2=None)	--为index1到index2之间的内容添加一个Tag（tagName参数指定Tag的名字） --若index2参数忽略，则单独为index1指定的内容添加Tag
10	tag_bind(tagName, sequence, func, add=None)	--为Tag绑定事件
11	tag_config(tagName, cnf=None, **kw)	--与tag_configure(tagName, cnf=None, **kw)一样
12	tag_configure(tagName, cnf=None, **kw)	--设置tagName的选项
13	window_create(index, **options)	--在index参数指定的位置嵌入一个window对象



# Text组件索引|Indices的使用方法

## ■ 索引|Indices的使用方法

- ◆ Index（索引）用来指向文本框中内容的**位置**。该索引是一个**字符串**，可以有不同的表示方法。Text组件的各种索引类型参见**教材表5- 29**
- ✓ **“line.column”**（行/列）：最基本的索引方式，将索引位置的行号和列号以字符串的形式表示出来
- ✓ **“line.end”**（某一行的末尾）：该行最后一个字符的位置
- ✓ **INSERT**（或**“insert”**）：Text组件中对应插入光标的位置
- ✓ **CURRENT**（或**“current”**）：对应与当前鼠标指针最接近的位置
- ✓ **END**（或**“end”**）：文本最后一个字符之后的那个位置

**注意：行号以1开始，列号以0开始**

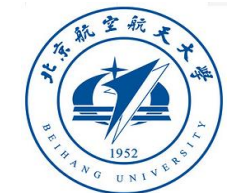




# Text组件索引Indices的使用方法（续）

## ■ 索引Indices的使用方法

- ◆ Text组件的以下**方法**都需要用到索引
- ◆ 利用索引，可向文本框中**插入**新的文本，或者**获取**任何位置的文本
  - ✓ `get(index1, index2=None)`
  - ✓ `image_create(index, cnf={}, **kw)`
  - ✓ `insert(index, text, *tags)`
  - ✓ `tag_add(tagName, index1, index2=None)`
  - ✓ `window_create(index, **options)`





# Text组件的get方法

## ■ `get(index1, index2=None)`

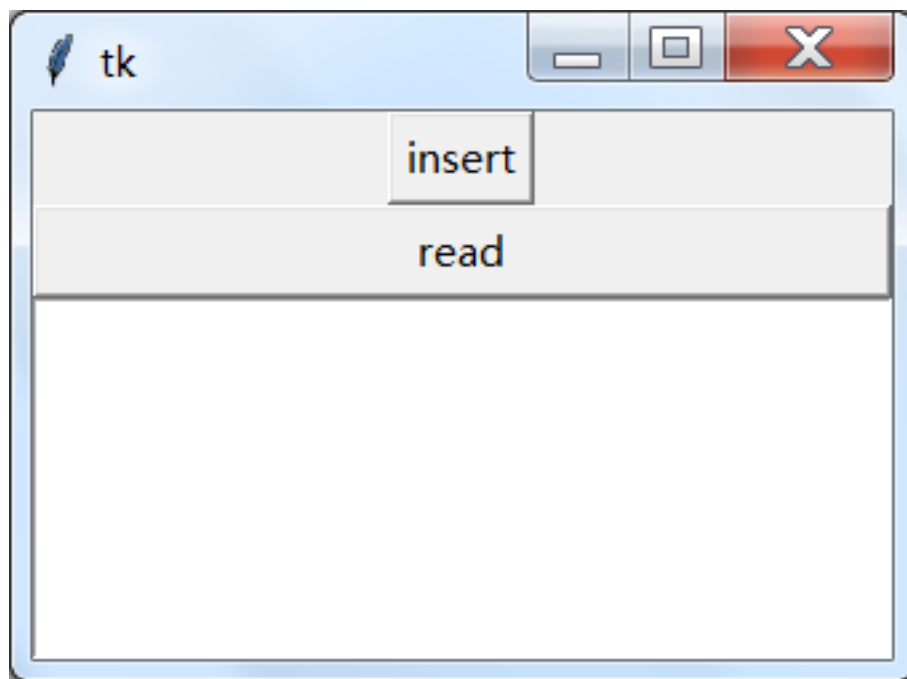
- ◆ `get`方法可以获取`index1`到`index2` (**不包含**) 之间的文本, 但不包括`index2`指定的文本。如果是**中文**, 则以**字符**为单位返回文本
- ◆ 如果文本框中包含`image`和`window`的嵌入对象, 则均被忽略
- ◆ 如果包含有**多行**文本, 则返回的文本中自动插入**换行符** (`'\n'`)
- ◆ 如果`index2`参数忽略, 则返回`index1`所指的那个字符
- ◆ 例如: 若文本框`textbox`中第2行为 “我是中国人”

```
print (textbox.get("2.0", "2.3"))    #获取第2行第0列~第3列  
                                     (不包括第3列) 之间的字符, 并打印  
则会打印 “我是中”。
```



## 【例13.5】Text的简单应用

**【例13.5】** 设计一个简单的GUI，采用Button组件创建两个按钮，采用Text组件创建一个高度为10、宽度为40的文本框。当单击一个按钮时，将一个字符串插入到文本框的第1行开始的位置；单击另一个按钮时，获取文本框中的从1.0位置到末尾的文本，并输出。

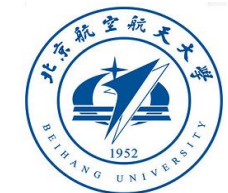




## 【例13.5】设计思路

### ■ 设计思路

- ◆ 采用Text组件的**insert方法**将文本插入到文本框指定位置  
**insert(index, text, \*tags)**
- ◆ 采用Text组件的**get方法**获取文本框中指定位置的文本  
**get(index1, index2=None)**



## 【例13.5】程序

### 例13.5-Text\_simple.py

.....

**# (1) 定义2个按钮的回调函数**

def insertBtn():

**t.insert(1.0, 'a line')** #将a line插入到文本框t的第1行开始的位置

def readBtn():

**l = t.get(1.0,END)** #获得t的从1.0位置到末尾的文本

print(l)

**# (2) 创建文本框**

**t = Text(top,height=10, width=40)**#高为10行，宽为40个字符

**# (3) 创建2个按钮**

Button(top,text='insert',**command=insertBtn**)

Button(top,text=' read ',**command=readBtn**)

当点击按钮时会触发command指定的回调函数，可以插入文本和输出文本

## 【例13.5】程序（续）

### # (4) 组件几何布局

```
insert_btn.pack()
```

#默认组件从上至下放置

```
read_btn.pack()
```

```
t.pack()
```

```
top.mainloop()
```

### ■ 文本框中的位置行数从1开始，列数从0开始计数

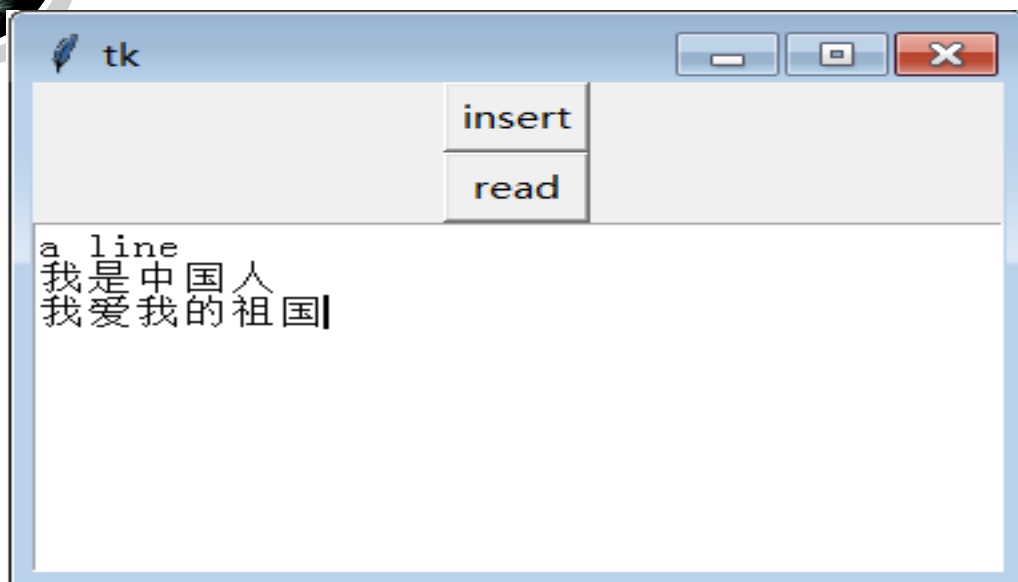
◆ 例： **`l = t.get(1.0,END)`** #获得t的从1.0位置（即第一行的开头）到末尾的文本

◆ **`l = t.get(2.0)`** #获得t的2.0位置所在的文本





## 【例13.5】的运行结果



**在指定位置插入文本和编辑文本**

```
>>>  
a line  
我是中国人  
我爱我的祖国
```

**获取文本框中的从1.0位置到末尾的文本**





# Text的使用技巧

## ■ Text的使用技巧

- (1) 使用**insert方法**向Text中添加文本
- (2) 使用**window\_create()方法**和**image\_create()方法**插入 windows对象（如按钮）或图像对象
- (3) 使用**delete()方法**删除Text组件中的内容
- (4) 实现“**恢复**”和“**撤销**”功能
- (5) Text组件索引**Indices**的使用方法
- (6) Text组件**Mark**的使用方法
- (7) Text组件**Tag**的使用方法

■ 参见**教材5.4.3.2**



# 其他有用的组件简介

## 1、标准对话框模块

- ◆ Tkinter提供了三种标准对话框模块，可以创建弹出式（pop-up）对话框窗口，分别是：**messagebox（消息对话框）**，**filedialog（文件对话框）**和**colorchooser（颜色选择对话框）**

## 2、Listbox组件

- ◆ **Listbox**是**列表框**组件。用于显示一个包含多个选项的列表框，用户可以选择其中的一项或者多项
- ◆ 可以通过参数selectmode将列表设置为单选或多选

## 3、Canvas组件

- ◆ **Canvas**是**画布**组件。画布是一个矩形区域，用于绘制图形或其他复杂布局图
- ◆ 在其中可以放置**图形、文本、组件或窗体**等





# 13.3 组件的几何布局

北京航空航天大学





## 13.3 组件的几何布局

- ◆ 13.3.1 利用pack方法布局组件
- ◆ 13.3.2 利用grid方法布局组件





# Tkinter的几何布局管理器

- Tkinter的几何布局管理器包括pack、grid和place，用来组织和**管理整个父配件（组件）区中所有子配件（组件）**的布局
- 三种方法各有特点 **适于简单布局**
  - ◆ **pack**按组件创建的先后**顺序**排列组件（默认从上至下）
  - ◆ **grid**按行/列形式排列组件：适于**规则**布局
  - ◆ **place**允许指定组件的大小和位置：适于**精确**布局





## 13.3.2 利用grid方法布局组件

- grid方法采用类似**表格**的结构组织各种组件，用其设计**对话框**和带有**滚动条**的窗体效果最好
- grid采用**行列**确定位置，行列交汇处为一个**单元格**。每一列中，列宽由这一列中最宽的单元格确定；每一行中，行高由这一行中最高的单元格决定
- 组件并不是充满整个单元格的，可以指定单元格中剩余空间的使用





# grid方法的用法

格式

**WidgetObject.grid(*options*)**

## ■ 输入参数

**options**: 可选参数，可以不设置。共有**10**个参数，参见**教材表5-34**。

常用的参数：

- ✓ **column**: 指定组件所置单元格的**列号**，自然数（起始默认值为**0**，而后累加）
- ✓ **columnspan**: 指定用多少列（跨列）显示该组件
- ✓ **row**: 指定组件所置单元格的**行号**，自然数（起始默认值为**0**，而后累加）
- ✓ **sticky**: 指定组件紧靠所在单元格的某一边角。“n”——北，“s”——南，“w”——西，“e”——东  
“nw”，“sw”，“se”，“ne”，“center”(**默认为”center”**)

■ 使用**grid\_forget()**方法，可以将组件从屏幕中**“删除”**，  
但**并没有销毁**该组件，只是使其**不可见**

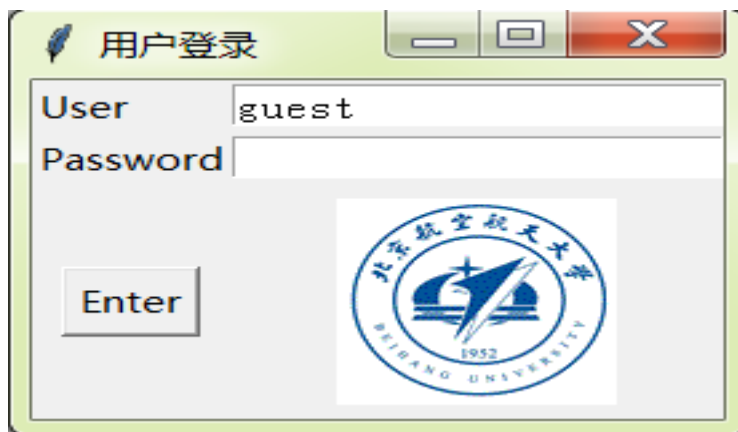




## 【例13.7】grid方法的应用

**【例13.7】** 设计一个用户登录界面，使用2个**标签**显示“User”和“Password”，使用2个**文本框**便于用户输入用户名和密码，假设初始用户名为guest。使用一个**按钮**“Enter”作为确认键。在GUI中插入一个好看的**图片**。使用**grid方法**，对组件进行布局。

当输入用户名和密码后，单击按钮“Enter”，根据用户名、密码是否与预设的匹配，弹出不同的提示信息。





## 【例13.7】设计思路

### ◆ 如何在用户名的文本框中插入初始值guest?

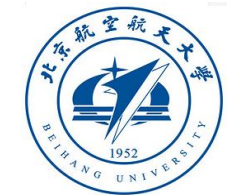
- ✓ 采用Text组件的**insert**方法

```
username_textbox.insert(INSERT,"guest")
```

### ◆ 如何在GUI中插入一幅图片?

- ✓ Label组件不仅可以显示文本，还可以显示**PhotoImage**对象
- ✓ 先使用**PhotoImage**方法创建一个图片（使其与现有的一张图片关联）  
；再创建一个Label，设置其image参数为该图片

```
photo = PhotoImage(file="BUAA.gif")  
photo_label=Label(top, image=photo)
```



## 【例13.7】设计思路（续）

### ◆ 如何获取用户名或密码文本框中的文本？

- ✓ 用Text组件的**get方法**获取文本框中指定位置的文本

```
user=username_textbox.get(1.0,END)
```

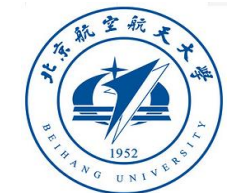
### ◆ 单击按钮“Enter”后如何显示提示信息？

- ✓ Tkinter提供了标准对话框模块**messagebox**，用于在应用程序中创建**消息对话框**
- ✓ 其中**showinfo**方法可以创建**显示信息对话框**

```
import tkinter.messagebox  
tkinter.messagebox.showinfo(title, message, options)
```

对话框名称

显示的信息



# 【例13.7】程序

例13.7-grid-Text.py

```
from tkinter import*  
import tkinter.messagebox #导入tkinter.messagebox模块。必须有此句!  
from tkinter.messagebox import*
```

```
top.title("用户登录")
```

调用messagebox模块的showinfo方法时若省略tkinter.messagebox，则必须有此句

## # (1) 创建组件

```
label1=Label(top,text="User") #显示输入用户名  
label2=Label(top,text="Password") #显示输入密码
```

```
username_textbox=Text(top,height = 1, width = 20) #用户名输入框  
username_textbox.insert(INSERT,"guest") #初始用户名给定为guest  
password_textbox=Text(top,height = 1, width =20) #密码输入框
```

```
enter_button=Button(top,text="Enter",command=lambda:enter()) #确定键
```

lambda  
表达式

```
photo = PhotoImage(file="BUAA.gif") #创建一幅图片  
photo_label=Label(top, image=photo) #Label上显示图像
```

## 【例13.7】程序（续1）

### # (2) 定义enter\_button的回调函数

```
def enter():
```

```
    user=username_textbox.get(1.0,END)
```

#提取用户名

```
    password=password_textbox.get(1.0,END)
```

#提取密码

因为回车符也被提取

```
    if ((user=="amj\n")and(password=="123\n")):
```

#如果用户名和密码正确

```
        tkinter.messagebox.showinfo('welcome','let's do a survey')
```

#显示提示框

```
    elif ((user!="amj\n")and(password=="123\n")): #如果用户名不正确，密码正确
```

```
        tkinter.messagebox.showinfo("information","Your username is wrong!")
```

- ◆ Tkinter的**messagebox**模块在应用程序中创建**消息对话框**
- ◆ 其方法**showinfo(title, message, options)** 用来**显示信息对话框**



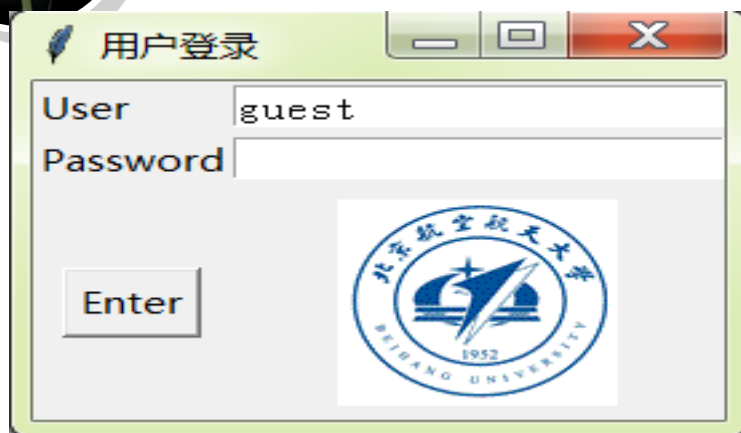
## 【例13.7】程序（续2）

```
elif ((user=="amj\n")and(password!="123\n")): #如果用户名正确，密码不正确
    tkinter.messagebox.showinfo("information","Your password is wrong!")
elif ((user!="amj\n")and(password!="123\n")): #如果用户名和密码都不正确
    tkinter.messagebox.showinfo("information","Your username and password are all
wrong!")
```

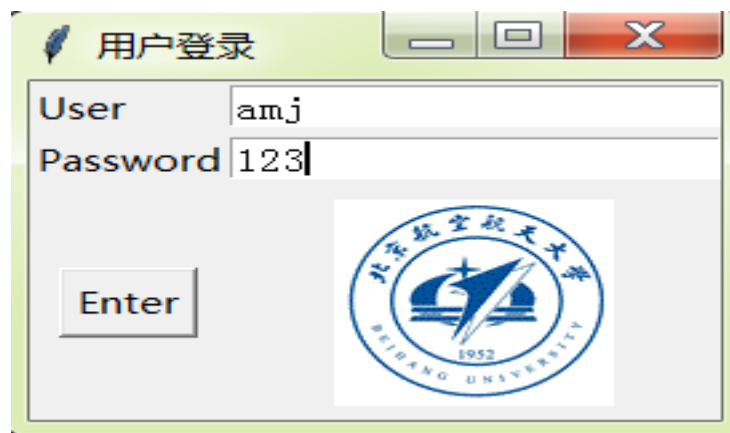
### # (3) 几何布局

```
label1.grid(row=0, sticky=W)    #第0行，默认第0列，组件紧靠所在单元格的西边
username_textbox.grid(row=0,column=1)    #第0行，第1列
label2.grid(row=1, sticky=W)    #第1行
password_textbox.grid(row=1,column=1)    #第1行，第1列
enter_button.grid(row=2,column=0,padx=5,pady=5)
photo_label.grid(row=2,column=1,columnspan=3,padx=5,pady=5)
```

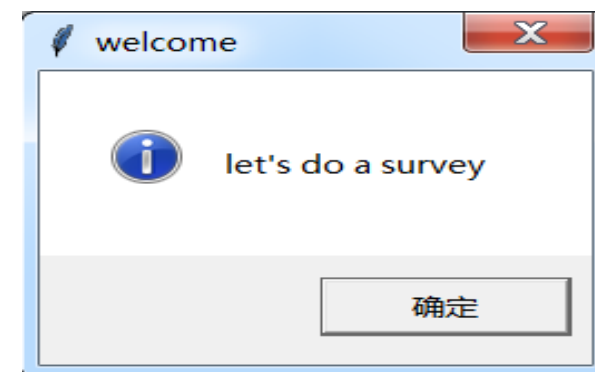
## 【例13.7】程序运行结果



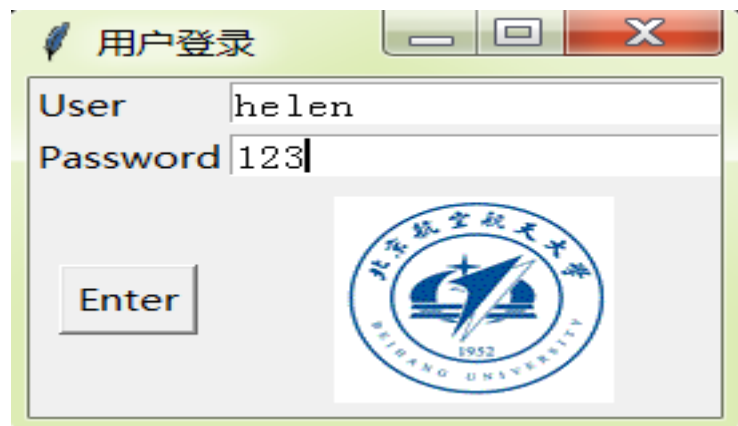
初始界面



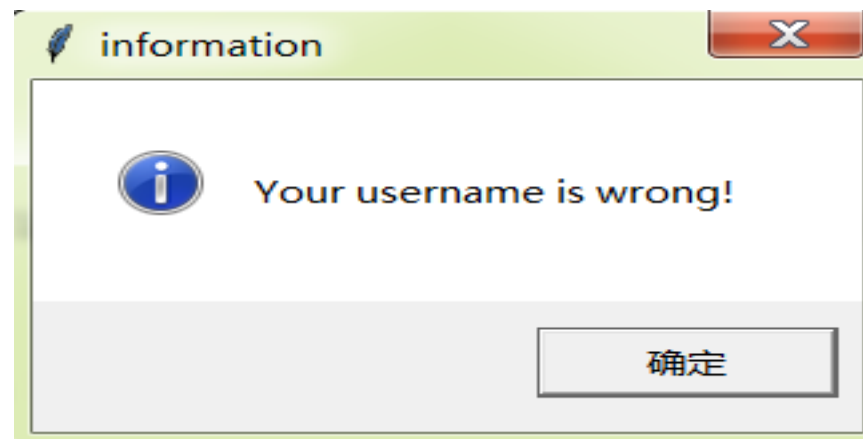
用户输入正确



单击“Enter”弹出  
信息对话框



用户名错误







# 试一试：采用Entry组件作为输入框

## ■ 改写程序，采用Entry组件作为输入框

### ◆ Entry用来创建单行文本框

```
Entry ( master, option, ... )
```

- ◆ 可选参数有show（指定始终显示的内容）、width（文本框的宽度）.....，但没有height参数，因为它只有一行文本

## 与采用Text组件有何区别？







# 例13.7-grid-entry.py主要不同

例13.7-grid-entry.py

## # (1) 创建组件

```
password_entry=Entry(top, show = "*", width =20)
```

Text没有

#密码输入框。show = "\*"表示**无论输入什么，都显示星号**

## # (2) 定义enter\_button的回调函数

```
def enter():
```

不必传递任何参数

```
    user=username_entry.get() #提取用户名
```

```
    password=password_entry.get() #提取密码
```

```
    if ((user=="amj")and(password=="123")): #如果用户名和密码正确。
```

**不能加上 "\n"**，因为提取的是输入框中文本内容，**没有回车符**

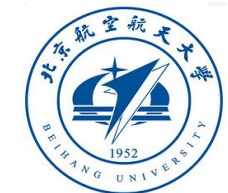
```
        messagebox.showinfo("welcome","let's do a survey")
```



# Tkinter库中函数的详细用法

- Tkinter库中函数的详细用法，请查阅Python documentation 中的相关的内容.

<https://docs.python.org/3/library/tk.html>





## 三、Pandas：数据分析库

- Pandas是基于NumPy的**数据分析**库
- 它提供了大量**高级数据结构**和能够高效便捷地操作大型数据集的工具
  - (1) **Series**：带标签的一维数组，由一组数据（各种NumPy数据类型）和与之相关的数据标签（索引）组成；
  - (2) **DataFrame**：带标签且大小可变的二维表格结构；
  - (3) **Panel**：带标签且大小可变的三维数组
- Pandas提供了大量的**函数**用于**生成、访问、修改、保存**不同类型的数据，处理缺失值、重复值、异常值，并能够结合Matplotlib绘图库进行数据可视化



# Pandas的使用

- 使用Pandas库之前，首先需要导入Pandas库和NumPy库

```
import pandas as pd  
import numpy as np
```

## 1、带标签的一维数组Series

- 如何创建Series?

index参数指定索引  
默认为0, 1, 2, .....

```
In [3]: series_obj=pd.Series([5,8,-2,1],index=['a','b','c','d'])
```

```
In [3]: series_obj
```

索引

Out[3]:

a	5
b	8
c	-2
d	1

dtype: int64



# Pandas的使用

## 2、二维数据表DataFrame

- 含有一组有序的列，每列可以是不同的值类型（如**数值**，**字符串**，**布尔值**）
- DataFrame既有行索引也有列索引。允许使用**index**参数指定**行索引**（默认为0,1,2, .....），使用**columns**参数指定**列索引**（**列名**）的顺序
- 可以使用**字典**创建DataFrame：字典的**键**指明**列名**；**值**即为该列的**数据**

In [1]:

```
data={'name':['apple','banana','pear','orange'],  
      'price':[5.0,3.2,2.5,4.5],'number':[10.0,5.0,4.0,8.0]}
```

In [2]:

```
df=pd.DataFrame(data) #创建二维数组
```

In [3]:df

Out[3]:

	name	price	number
0	apple	5.0	10.0
1	banana	3.2	5.0
2	pear	2.5	4.0
3	orange	4.5	8.0

列索引

行索引



# Pandas库的主要功能

## (1) 导入数据

- ◆ 可以导入**CSV**文件、分隔的**文本**文件、**Excel**文件、读取**SQL表/数据库**等

```
pd.read_csv(filename)           #filename如'name.csv'  
pd.read_excel(filename)        #filename如'name.xlsx'  
pd.read_sql(query, connection_object)
```

## (2) 输出数据

- ◆ 经分析处理后的数据可以写入**CSV**文件、**Excel**文件、一个**SQL**表或**JSON**格式的文件等

```
df.to_csv(filename)             #df为已创建的DataFrame  
df.to_excel("examples/ex1.xlsx") #写入Excel文件  
df.to_sql(table_name, connection_object)
```



# Pandas库的主要功能：数据表信息查看

## (3) 数据表信息查看

- ◆ 可以查看数据表基本信息（**维度**、**列名称**、数据格式、所占空间等）、每一列数据的格式、某一系列格式、空值，还可以查看前10行数据、后10行数据等

**df.head(n)**  
**df.shape**  
**df.columns**

#查看数据表的前n行  
#查看数据表的行数和列数（维度）  
#查看数据表的列名称

```
In [5]: df
Out[5]:
```

	name	price	number
0	apple	5.0	10.0
1	banana	3.2	5.0
2	pear	2.5	4.0
3	orange	4.5	8.0

```
In [6]: df.head(2)
Out[6]:
```

	name	price	number
0	apple	5.0	10.0
1	banana	3.2	5.0

查看**维度**

```
In [8]: df.shape
Out[8]: (4, 3)
```

查看**列名称**

```
In [9]: df.columns
Out[9]: Index(['name', 'price', 'number'], dtype='object')
```

- ◆ 通过**数据表.列索引**查看某一系列数据

```
In [10]: df.name
Out[10]:
```

0	apple
1	banana
2	pear
3	orange

Name: name, dtype: object

或者df[name]





# Pandas库的主要功能：加入数据

## (4) 加入数据

- ◆ 可以使用**append方法**将一个数据表中的所有**行**添加到另一个数据表的末尾，返回一个**新**的数据表（**原数据表不变**）：`append_result=df1.append(df2)`
- ◆ 也可以使用**concat方法**将一个数据表中的**所有行或列**（默认参数**axis=0**，表示按**行**添加；**axis=1**表示按**列**添加）添加到另一个数据表的末尾，返回一个**新**的数据表  
`concat_result=pd.concat([append_result, df3],axis=1)`

**【加入数据的示例】** 创建两个二维数据表df1、df2，列索引为学号、数学成绩，行索引分别为0~3、4~7。

- (1) 采用**append()方法**将df2中的**行**添加到df1的末尾，得到新的数据表。
- (2) 创建一个二维数据表df3，包含8行数据，列索引为学号、语文成绩，行索引为0~7。
- (3) 使用**concat方法**将df3中的**所有列**添加到新数据表的末尾；结果**写入Excel**文件。





# 加入数据的示例程序

pandas-加入数据.py

```
import pandas as pd
```

```
import numpy as np
```

```
data1={'学号': [1001,1002,1003,1004], '数学成绩': [50,75,89,66]}
```

```
data2={'学号': [1005,1006,1007,1008], '数学成绩': [92,73,84,99]}
```

```
df1 = pd.DataFrame(data1,index=[0,1,2,3])
```

```
df2 = pd.DataFrame(data2,index=[4,5,6,7])
```

或者df1 = pd.DataFrame(data1,  
index=range(4))

## #1、采用append()方法加入数据

```
append_result=df1.append(df2) #append()方法将df2中的行添加到df1的末尾（列数应该相同）
```

```
print("append_result: ", "\n", append_result)
```

```
print("df1不变: ", "\n", df1) #原数据表不变
```

```
print()
```

## #2、采用concat()方法增加列

```
data3={'学号': [1001,1002,1003,1004,1005,1006,1007,1008], '语文成绩': [62,80,85,70,95,80,81,89]}
```

```
df3 = pd.DataFrame(data3)
```

```
#将df3中的所有列添加到append_result的末尾（行数应该相同）。axis=1表示沿1轴连接，即按列添加
```

```
concat_result=pd.concat([append_result, df3],axis=1)
```

```
print("concat_result: ", "\n", concat_result)
```

```
concat_result.to_excel("成绩单.xlsx")
```

```
#写入Excel文件
```

# 加入数据后结果

df1:

	学号	数学成绩
0	1001	50
1	1002	75
2	1003	89
3	1004	66

df2:

	学号	数学成绩
4	1005	92
5	1006	73
6	1007	84
7	1008	99

原数据表

按行添加

append\_result:

	学号	数学成绩
0	1001	50
1	1002	75
2	1003	89
3	1004	66
4	1005	92
5	1006	73
6	1007	84
7	1008	99

df1不变:

	学号	数学成绩
0	1001	50
1	1002	75
2	1003	89
3	1004	66

使用append方法后

concat\_result:

	学号	数学成绩	学号	语文成绩
0	1001	50	1001	62
1	1002	75	1002	80
2	1003	89	1003	85
3	1004	66	1004	70
4	1005	92	1005	95
5	1006	73	1006	80
6	1007	84	1007	81
7	1008	99	1008	89

按列添加

使用concat方法后

	A	B	C	D	E
1		学号	数学成绩	学号	语文成绩
2	0	1001	50	1001	62
3	1	1002	75	1002	80
4	2	1003	89	1003	85
5	3	1004	66	1004	70
6	4	1005	92	1005	95
7	5	1006	73	1006	80
8	6	1007	84	1007	81
9	7	1008	99	1008	89

成绩单.xlsx

■ append方法和concat方法均返回一个新数据表，原数据表不变



# Pandas库的主要功能：合并/连接

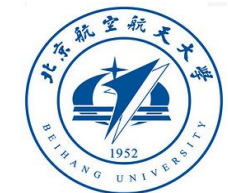
## (5) 合并/连接

- ◆ 可以把多个DataFrame对象**合并**为一个DataFrame对象
- ◆ **merge方法**用于合并两个数据集，并返回合并后数据集

格式

```
pd.merge(<数据表1>, <数据表2>, how=< 指定值 > )
```

- ◆ **缺省**选择两个数据集的**交集**合并
- ◆ 用参数“how”可以指定不同的合并方式
  - ✓ how='inner'：匹配合并，交集（取两个数据集都有的项合成为一个数据集）
  - ✓ how='left'：按照第一个数据集有的项进行合并
  - ✓ how='right'：按照第二个数据集有的项进行合并
  - ✓ how='outer'：并集



# 合并数据的示例程序

## pandas-合并.py

```
import pandas as pd
import numpy as np
```

### #1、创建二维数据表

```
data1={'学号': [1001,1002,1003,1004], '数学成绩': [50,75,89,66]}
data2={'学号': [1005,1006,1007,1008], '数学成绩': [92,73,84,99]}
df1 = pd.DataFrame(data1, index=np.array(range(4))) #采用range函数生成自动索引, 更简单
df2 = pd.DataFrame(data2, index=np.array(range(4,8)))
```

### #2、merge方法合并两个数据集, 并返回合并后数据集

```
df_outer=pd.merge(df1,df2,how='outer') #并集
print("df1与df2合并后的df_outer: ", "\n", df_outer)
```





# 合并数据后结果

df1:

	学号	数学成绩
0	1001	50
1	1002	75
2	1003	89
3	1004	66

df2:

	学号	数学成绩
4	1005	92
5	1006	73
6	1007	84
7	1008	99

原数据表

df1与df2合并后的df\_outer:

	学号	数学成绩
0	1001	50
1	1002	75
2	1003	89
3	1004	66
4	1005	92
5	1006	73
6	1007	84
7	1008	99

合并后数据

■ 返回合并后数据集，原数据表不变





# Pandas库的主要功能：筛选

## (6) 筛选

- ◆ **筛选**即从数据表中选择符合特定条件的数据，并返回筛选后数据集
- ◆ 可以使用“与”、“或”、“非”条件进行筛选
- ◆ 或者使用**query函数**进行筛选

格式

**数据表[[数据表[列名称]>= 指定值]**

```
select_df2=df1[df1['数学成绩'] >=90] #筛选出'数学成绩'列大于90的行
```



# 筛选的示例程序

## pandas-筛选.py

.....

### #1、创建二维数据表

```
data1={'学号':range(1001,1013),'数学成绩':[50,75,89,66,92,73,84,99,59,69,90,87],  
      '语文成绩':[62,80,85,70,95,80,81,89,72,80,93,91]}  
df1 = pd.DataFrame(data1)  
print("df1: ", "\n", df1)
```

### #2、筛选

```
select_df2=df1[df1['数学成绩']>=90] #筛选出'数学成绩'列大于90的行  
select_df3=df1[(df1['数学成绩']>60) & (df1['数学成绩']<90)] #筛选出90>'数学成绩'列>60的行  
select_df4=df1[df1['语文成绩']<80] #筛选出'语文成绩'列小于80的行
```

### #使用“与”进行筛选，按指定列名顺序排序

```
select_and=df1.loc[(df1['数学成绩']>=90) & (df1['语文成绩']>=90),['语文成绩','数学成绩']  
print("数学成绩和语文成绩均大于等于90的学生: ", "\n", select_and)
```

按指定列名顺序排序





# 筛选结果

df1:

	学号	数学成绩	语文成绩
0	1001	50	62
1	1002	75	80
2	1003	89	85
3	1004	66	70
4	1005	92	95
5	1006	73	80
6	1007	84	81
7	1008	99	89
8	1009	59	72
9	1010	69	80
10	1011	90	93
11	1012	87	91

原数据表

'数学成绩'大于90的学生:

	学号	数学成绩	语文成绩
4	1005	92	95
7	1008	99	89
10	1011	90	93

90>'数学成绩'>60的学生:

	学号	数学成绩	语文成绩
1	1002	75	80
2	1003	89	85
3	1004	66	70
5	1006	73	80
6	1007	84	81
9	1010	69	80
11	1012	87	91

'语文成绩'小于80的学生

	学号	数学成绩	语文成绩
0	1001	50	62
3	1004	66	70
8	1009	59	72

数学成绩和语文成绩均大于等于90的学生:

	语文成绩	数学成绩
4	95	92
10	93	90





# Pandas库的主要功能：排序

## (7) 排序

- ◆ 使用**sort\_values方法**按照特定列的值排序，其中使用**参数by**指定按哪一列的数值排序。使用**参数ascending**指定按升序或降序排序：默认**ascending=True**，表示**升序**；**ascending=False**表示降序。
- ◆ **sort\_values**方法返回一个排序后数据表，但原始数据表不变
- ◆ 如果希望更新原始数据表，则在函数的括号中最后加上 “**inplace=True**”，表示在原表上进行操作。**inplace**参数默认为False。

**格式** **<数据表>.sort\_values (by= '<列名>',ascending=False)**

```
df2=df1.sort_values(by='数学成绩',ascending=False)
#按'数学成绩'列按降序对值排序，默认返回排序后数据表
```

- ◆ 还可以使用**sort\_index方法**按照索引排序，同样可以使用参数**ascending**指定按升序或降序排序



# 排序的示例程序

pandas-排序.py

.....

## #1、创建二维数据表

```
data1={'学号': range(1001,1013),'数学成绩': [50,75,89,66,92,73,84,99,59,69,92,87],  
       '语文成绩': [62,80,85,70,93,80,81,89,72,80,95,91]}  
df1 = pd.DataFrame(data1)  
print("df1: ", "\n", df1)
```

## #2、排序

```
df2=df1.sort_values(by='数学成绩',ascending=False) #按'数学成绩'列按降序对值排序，返回排序  
后数据表  
print("df1按'数学成绩'列按降序对值排序后返回一个排序后数据表: ", "\n", df2)  
print()
```

```
#先按'数学成绩'列对数据按降序排序；如果两个值相同，再按'语文成绩'列按降序排序，原地操作  
df1.sort_values(by=['数学成绩','语文成绩'],ascending=[False,False],inplace=True)  
print("df1先按'数学成绩'列按降序对值排序、再按'语文成绩'按降序排序后: ", "\n", df1)
```

# 排序结果

df1:

	学号	数学成绩	语文成绩
0	1001	50	62
1	1002	75	80
2	1003	89	85
3	1004	66	70
4	1005	92	93
5	1006	73	80
6	1007	84	81
7	1008	99	89
8	1009	59	72
9	1010	69	80
10	1011	92	95
11	1012	87	91

原数据表

df1按'数学成绩'列按降序对值排序后返回一个排序后数据表df2:

	学号	数学成绩	语文成绩
7	1008	99	89
4	1005	92	93
10	1011	92	95
2	1003	89	85
11	1012	87	91
6	1007	84	81
1	1002	75	80
5	1006	73	80
9	1010	69	80
3	1004	66	70
8	1009	59	72
0	1001	50	62

inplace参数默认为False, 返回排序后数据表, 原数据表不变

df1先按'数学成绩'列按降序对值排序、再按'语文成绩'按降序排序后:

	学号	数学成绩	语文成绩
7	1008	99	89
10	1011	92	95
4	1005	92	93
2	1003	89	85
11	1012	87	91
6	1007	84	81
1	1002	75	80
5	1006	73	80
9	1010	69	80
3	1004	66	70
8	1009	59	72
0	1001	50	62

"inplace=True", 原地操作, 原数据表被改变



# Pandas库的主要功能：分类汇总

## (8) 分类汇总

- ◆ **分类汇总**：对同一类型的多个数据进行统计汇总
- ◆ 先按照某一**列**字段对原始数据进行**分组（分类）**，再对该列**数值相同**（如同为男生）的若干行中其他列进行**求和**、**求平均值**、**计数**等操作
- ◆ 一般是先使用**groupby()方法**进行分组（分类），再使用**sum()方法**、**mean()方法**、**len函数**进行汇总

格式

```
g=<数据表>. groupby ('列名').value      #按'列名'分组，对value列汇总  
g.mean()                                #求value列中每组的平均值
```



# 分类汇总示例

`df=pd.DataFrame({'key':['a','b','c']*3,'value':np.arange(0,9)})` #创建二维数据表

```
In [19]: df=pd.DataFrame({'key':  
['a','b','c']*3,'value':np.arange(0,9)})
```

```
In [20]: df
```

```
Out[20]:
```

	key	value
0	a	0
1	b	1
2	c	2
3	a	3
4	b	4
5	c	5
6	a	6
7	b	7
8	c	8

`g=df.groupby('key').value` #按'key'分组, 对value列汇总  
`g_mean=g.mean()` #求每组的平均值

```
In [21]: g=df.groupby('key').value
```

```
In [22]: g.mean()
```

```
Out[22]:
```

```
key
```

```
a      3
```

```
b      4
```

```
c      5
```

```
Name: value, dtype: int32
```

■ 或者两条语句写在一起:

```
df_mean=df.groupby('key')['value'].mean()
```





# 【微实例5.10】Pandas数据汇总示例

## 【微实例5.10】Pandas数据汇总示例：学生成绩汇总。

已知某小学某个小班的一个小组（10人）期中考试语文和数学成绩，存储于一个Excel文件中。

试读入该Excel文件；按性别字段进行分类，计算该小组男生和女生的人数；求各科目的平均分；计算男生和女生的人数、语文的总分和均值

	A	B	C	D	E
1	学号	性别	姓名	语文	数学
2	1	男	王刚	90	95
3	2	女	张丹	85	90
4	3	男	唐磊	80	86
5	4	女	马丽	91	83
6	5	女	洪艳	75	79
7	6	男	赵力	69	72
8	7	男	章宁	83	89
9	8	女	孙琴	88	80
10	9	男	马晨	78	75
11	10	男	杨硕	94	99

## 设计思路

- ◆ 使用read\_excel方法，读取Excel文件中的数据并创建DataFrame结构；
- ◆ 使用groupby()方法按性别进行分组，使用count()方法对'性别'字段进行计数；
- ◆ 使用mean()方法求各科目的平均分；
- ◆ 使用agg ()方法求语文的总分和均值



# 【微实例5.10】程序

## 微实例5.10-Pandas数据汇总示例.py

.....

### #1、导入Excel文件，转换为二维数据表

```
df = pd.DataFrame(pd.read_excel('score.xlsx'))  
print("df: ", "\n", df)
```

读入Excel文件

### #2、按'性别'分类，对'性别'字段进行计数

```
df_count=df.groupby('性别')['性别'].count()  
print("对'性别'字段进行计数， df_count: ", "\n", df_count)
```

### #3、按'性别'分类，分别计算各列的均值

```
df_mean=df.groupby('性别')['语文','数学'].mean()  
print("分别计算语文和数学的均值， df_mean: ", "\n", df_mean)
```





## 【微实例5.10】程序（续）

len统计语文成绩  
中男和女的个数

**#4、按'性别'分类，并分别计算男和女的人数、'语文'的总分和均值**

```
df_Chinese=df.groupby('性别')['语文'].agg([len,np.sum, np.mean])  
print("计算男和女的人数、'语文'的总分和均值， df_Chinese: ", "\n",df_Chinese)
```

**#5、按列小计（即按列求和。若某列有字符，则列出所有字符）**

```
df_sum=df.sum()  
print("df_sum: ", "\n",df_sum)
```

**#6、按列求均值（若某列有字符，则不计算）**

```
df_mean=df.mean()  
print("df_mean: \n",df_mean)
```







## 【微实例5.10】运行结果

```
df:
   学号  性别  姓名  语文  数学
0  1001  男   王刚   90   95
1  1002  女   张丹   85   90
2  1003  男   唐磊   80   86
3  1004  女   马丽   91   83
4  1005  女   洪艳   75   79
5  1006  男   赵力   69   72
6  1007  男   章宁   83   89
7  1008  女   孙琴   88   80
8  1009  男   马晨   78   75
9  1010  男   杨硕   94   99
```

读入、转换后的二维数据表

```
对'性别'字段进行计数, df_count:
  性别
女     4
男     6
Name: 性别, dtype: int64

分别计算语文和数学的均值, df_mean:
           语文     数学
性别
女  84.750000  83.0
男  82.333333  86.0
```

对'性别'字段进行计数  
计算语文和数学的均值

```
计算男和女的人数、'语文'的总分和均值,
df_Chinese:
           len  sum      mean
性别
女           4  339  84.750000
男           6  494  82.333333
```

计算男和女的人数、'语文'的总分和均值

```
df_sum:
  学号
      10055
性别      男女男女男女男男
姓名  王刚张丹唐磊马丽洪艳赵力章宁孙琴马晨杨硕
语文      833
数学      848
dtype: object
```

按列小计

