创建一个窗口

```
from tkinter import *

window = Tk()
window.title("First Window")
window.mainloop()
```

最后一行我们调用了`mainloop`函数，这个函数将让窗口等待用户与之交互，直到我们关闭它。如果忘记调用`mainloop`函数的话，将不会向用户显示任何内容（没有窗口）。

添加一个标签组件

为了给之前的例子增加一个标签组件，我们可以使用`Label`类：

```
lbl = Label(window, text="Hello")
```

我们可以通过`grid`函数设置其在窗口的位置：

```
lbl.grid(column=0, row=0)
```

完整代码：

```
from tkinter import *

window = Tk()
window.title("First Window")
lbl = Label(window, text="Hello")
lbl.grid(column=0, row=0)
window.mainloop()
```

值得注意的是`lbl`没有调用`grid`函数的话是不会显示的。

设置标签字体大小

我们可以使用`font`参数设置标签字体大小：

```
lbl = Label(window, text="Hello", font=("Arial Bold", 50))
```

`font`参数不光可以在标签组件中用，其他组件也可以使用！

可是，现在窗口貌似太小了，连窗口的标题都看不全，如何设置窗口大小呢？

设置窗口大小

我们可以用`geometry`函数来设置窗口大小：

```
window.geometry("350x200")
```

以上代码将会把窗口设置成350个像素宽，200个像素高。

添加一个按钮组件

让我们给窗口增加一个按钮组件，它的创建和添加方式和标签组件差不多：

```
btn = Button(window, text="Click Me")
btn.grid(column=1, row=0)
```

完整代码如下所示：

```
from tkinter import *

window = Tk()
window.title("First Window")
window.geometry("350x200")
lbl = Label(window, text="Hello")
lbl.grid(column=0, row=0)
#lbl.place(x=15, y=5, width=50, height=20)
btn = Button(window, text="Click Me")
btn.grid(column=1, row=0)
window.mainloop()
```

更改按钮前景和背景颜色

我们可以用`fg`参数设置按钮或其他组件的前景色。

我们可以用`bg`参数设置按钮或其他组件的背景色。

```
btn = Button(window, text="Click Me", bg="orange", fg="red")
```

现在，如果点击按钮，什么都不会发生，因为我们没有写处理点击事件的代码。

处理按钮点击事件

首先，我们编写一个当按钮点击后需要执行的函数：

```
def clicked():
    lbl.configure(text="Button was clicked!")
```

然后，我们注明一下点击时要调用的函数：

```
btn = Button(window, text="Click Me", command=clicked)
```

完整代码如下所示：

```
from tkinter import *

window = Tk()
window.title("First Window")
window.geometry("350x200")
lbl = Label(window, text="Hello")
lbl.grid(column=0, row=0)

def clicked():
    lbl.configure(text="Button was clicked!")

btn = Button(window, text="Click Me", command=clicked)
btn.grid(column=1, row=0)
window.mainloop()
```

添加一个文本框

在之前的例子中我们了解了如何添加一些简单组件，现在我们将通过`Tkinter`的`Entry`类获取到用户输入。我们可以这样用`Entry`类创建一个文本框：

```
txt = Entry(window, width=10)
```

然后可以用`grid`函数像之前那样添加到窗口中。

```
from tkinter import *

window = Tk()
window.title("First Window")
window.geometry("350x200")
lbl = Label(window, text="Hello")
lbl.grid(column=0, row=0)
txt = Entry(window, width=10)
txt.grid(column=1, row=0)
def clicked():
    lbl.configure(text="Button was clicked!")

btn = Button(window, text="Click Me", command=clicked)
btn.grid(column=2, row=0)
window.mainloop()
```

此时点击按钮，标签组件内的内容没有变化，如何将文本框中输入的信息在标签组件中显示呢?

我们可以用`get`函数获取到文本框中输入的信息，然后如下更改`clicked`函数来设置窗口大小:

```
def clicked():
    res = "Welcome to " + txt.get()
    lbl.configure(text=res)
```

如果我们在文本框中输入信息并点击按钮组件，标签组件将会显示`Welcome to 文本框输入信息`。

以下是完整代码:

```
from tkinter import *

window = Tk()
window.title("First Window")
window.geometry("350x200")
lbl = Label(window, text="Hello")
lbl.grid(column=0, row=0)
txt = Entry(window, width=10)
txt.grid(column=1, row=0)
def clicked():
    res = "Welcome to " + txt.get()
    lbl.configure(text=res)

btn = Button(window, text="Click Me", command=clicked)
btn.grid(column=2, row=0)
window.mainloop()
```

但每次我们运行代码后，我们都需要通过点击文本框来设置输入焦点才能输入信息，有什么办法可以自动设置输入焦点吗?

设置输入焦点

很简单，我们只需要调用`focus`函数来设置窗口大小:

```
txt.focus()
```

当我们运行代码后，会发现可以直接在文本框中输入信息而不需要点击文本框。

添加一个组合框

为了添加一个组合框，可以使用`Combobox`类：

```
from tkinter.ttk import *
combo = Combobox(window)
```

然后可以给组合框添加一些值。

```
from tkinter import *
from tkinter.ttk import *

window = Tk()
window.title("First Window")
window.geometry("350x200")
combo = Combobox(window)
combo['values'] = (1,2,3,4,5,"Text")
combo.current(1)
combo.grid(column=0, row=0)
window.mainloop()
```

如上所示，我们可以用元组设置组合框选项。

我们可以通过传递期望被选中选项的索引给`current`函数用以设置被选中的选项。

我们可以通过`get`函数获取到被选中的选项。

```
combo.get()
```

添加复选框

我们可以用`Checkbutton`类来创建一个复选框组件：

```
chk = Checkbutton(window, text="Choose")
```

能通过传递值设置复选框的状态：

```
from tkinter import *
from tkinter.ttk import *

window = Tk()
window.title("First Window")
window.geometry("350x200")
chk_state = BooleanVar()
chk_state.set(True) # Set check state
chk = Checkbutton(window, text="Choose", var=chk_state)
chk.grid(column=0, row=0)
window.mainloop()
```

上例我们用的是`BooleanVar`变量用来设置复选框的状态，也可以使用`IntVar`变量进行设置。

```
chk_state = IntVar()
chk_state.set(1) # Check
chk_state.set(0) # Uncheck
```

添加单选框

添加单选框可以用`Radiobutton`类创建一个文本框：

```
rad1 = Radiobutton(window, text="First", value=1)
```

value 设置组的， value值一样时，选中一个则都会选中。取消一个则都会取消。

我们需要给每个单选框设置不同的值，否则会不起作用。

```
from tkinter import *
from tkinter.ttk import *

window = Tk()
window.title("First Window")
window.geometry("350x200")
rad1 = Radiobutton(window, text="First", value=1)
rad2 = Radiobutton(window, text="Second", value=2)
rad3 = Radiobutton(window, text="Third", value=3)
rad1.grid(column=0, row=0)
rad2.grid(column=1, row=0)
rad3.grid(column=2, row=0)
window.mainloop()
```

当然，我们可以给这些单选框设置`command`参数指定一个函数，当用户点击它们时就会运行该函数。

```
rad1 = Radiobutton(window, text="First", value=1, command=clicked)

def clicked():
    # Do what you need
    pass
```

获取单选框值

为获取到选中单选框的值，我们可以将`IntVar`变量传给单选框的`variable`参数，之后用`IntVar`变量的`get`函数就可以获取到其值啦！

```
from tkinter import *
from tkinter.ttk import *

window = Tk()
window.title("First Window")
selected = IntVar()
lbl = Label(window, text="Show Value")
def clicked():
    lbl.configure(text=selected.get())
rad1 = Radiobutton(window, text="First", value=1, variable=selected)
rad2 = Radiobutton(window, text="Second", value=2, variable=selected)
rad3 = Radiobutton(window, text='Third', value=3,
command=clicked,variable=selected)

btn = Button(window, text="Click Me", command=clicked)
rad1.grid(column=0, row=0)
rad2.grid(column=1, row=0)
rad3.grid(column=2, row=0)
btn.grid(column=4, row=0)
lbl.grid(column=0, row=1)
window.mainloop()
```

添加文本区 （讲到这里了）

添加文本区可以用`ScrolledText`类创建一个文本框：

```
from tkinter import scrolledtext  # scrolled text
txt = scrolledtext.ScrolledText(window, width=40, height=10)
```

我们需要指定一个文本区的宽度和高度，否则它会占住整个窗口：

```
from tkinter import scrolledtext  # scrolled text
txt = scrolledtext.ScrolledText(window, width=40, height=10)
txt.grid(column=0, row=2)

window.mainloop()
```

用以下方法可以在文本区中插入文本：

```
txt.insert(INSERT, "Text goes here")
```

用以下方法可以将文本区中的文本删除：

```
txt.delete(1.0, END)
```

创建消息框

我们可以按如下方式创建一个消息框：

```
from tkinter import messagebox
messagebox.showinfo("Message title", "Message content")
```

我们创建一个按钮，当它被点击时显示一个消息框：

```
from tkinter import *
from tkinter import messagebox

window = Tk()
window.title("First Window")
window.geometry("350x200")
def clicked():
    messagebox.showinfo("Message title", "Message content")
btn = Button(window, text="Click here", command=clicked)
btn.grid(column=0, row=0)
window.mainloop()
```

添加Spinbox

`Spinbox`是输入控件；与`Entry`类似，但是可以指定输入范围值。

```
spin = Spinbox(window, from_=0, to=100)
```

通过`from_`和`to`参数指定范围，也可以用`width`参数指定控件宽度。

```
from tkinter import *

window = Tk()
window.title("First Window")
window.geometry("350x200")
spin = Spinbox(window, from_=0, to=100, width=5)
spin.grid(column=0, row=0)
window.mainloop()
```

也可以指定某些特定的值，而不是整个范围：

```
from tkinter import *

window = Tk()
window.title("First Window")
window.geometry("350x200")
spin = Spinbox(window, values=(3,8,11), width=5)
spin.grid(column=0, row=0)
window.mainloop()
```

这样，`Spinbox`控件就只会显示3个数字即3，8，11。

可以用如下方式给`Spinbox`控件设置默认值：

```
var = IntVar()
var.set(36)
spin = Spinbox(window, from_=0, to=100, width=5, textvariable=var)
```

添加进度条

我们可以用`Progressbar`类创建进度条：

```
from tkinter.ttk import Progressbar
bar = Progressbar(window, length=200)
```

设置一下进度条的值：

```
bar['value'] = 70
```

改变进度条的颜色可以按如下步骤进行：

```
from tkinter import *
from tkinter.ttk import Progressbar
from tkinter import ttk
window = Tk()
window.title("First Window")
window.geometry('350x200')
style = ttk.Style()
style.theme_use('default')
style.configure("black.Horizontal.TProgressbar", background='black') #
起个名字后面用
bar = Progressbar(window, length=200,
style='black.Horizontal.TProgressbar')
bar['value'] = 70
bar.grid(column=0, row=0)
window.mainloop()
```

添加文件对话框

我们可以按如下方式创建一个文件对话框：

```
from tkinter import filedialog # file dialog
file = filedialog.askopenfilename()
```

当你选择一个文件并点击打开，`file`变量将会保存该文件的路径。

如果想一次选择多个文件并打开，我们可以用：

```
files = filedialog.askopenfilenames()
```

用`filetypes`参数指定文件对话框的文件类型，只需在元组中指定扩展名即可。

```
file = filedialog.askopenfilename(filetypes = (("Text files","*.txt"),
("all files","*.*")))
```

`askdirectory`函数可以让我们请求目录：

```
dir = filedialog.askdirectory()
```

可以用`initialdir`参数指定打开的初始目录：

```
from tkinter import *
from tkinter import filedialog
import os

window = Tk()
window.title("First Window")
window.geometry('350x200')
def clicked():
    file =
filedialog.askopenfilenames(initialdir=os.path.dirname(__file__))
    print(file)
btn = Button(window, text="Click Me", command=clicked)
btn.grid(column=0, row=0)
window.mainloop()
```

用列表管理信息

```
from tkinter import *
from tkinter.ttk import *
from tkinter import ttk

window = Tk()
window.geometry('400x300')
dataTreeview = ttk.Treeview(window, show='headings', column=
('id','name', 'sex', 'python_score', 'database_score',
'c_language_score'))
# 设置表头的大小和位置
dataTreeview.column('id', width=15, anchor="center")
dataTreeview.column('name', width=20, anchor="center")
dataTreeview.column('sex', width=20, anchor="center")
dataTreeview.column('python_score', width=60, anchor="center")
dataTreeview.column('database_score', width=60, anchor="center")
dataTreeview.column('c_language_score', width=40, anchor="center")
# 设置表头的别名（给用户看的）
dataTreeview.heading('id', text='编号')
dataTreeview.heading('name', text='姓名')
dataTreeview.heading('sex', text='性别')
dataTreeview.heading('python_score', text='Python成绩')
dataTreeview.heading('database_score', text='数据库成绩')
dataTreeview.heading('c_language_score', text='C成绩')
dataTreeview.place(x=0, y=0, width=350, height=250)

window.mainloop()
```

拓展

```
from tkinter import *
from tkinter.ttk import *

window = Tk()

tree1 = Treeview(window, columns=('qy', 'dz')) # columns 纵列
# columns:设置树状列表中列的名称，将所有列名称用一个元组表示

tree1.column('#0', width=90, anchor=CENTER, stretch=False) # 树根
stretch 不伸展
# anchor：文字在cell里的对齐方式； stretch: 布尔值，表示列的宽度是否随整个部件
的改动而变化；width：列宽，单位是像素。
tree1.column('qy', width=90, anchor=CENTER)
tree1.column('dz', width=160, anchor=CENTER)
# 定义3个栏目的宽度，对齐方法，宽度是否窗体变化

tree1.heading('#0', text='')
tree1.heading('qy', text='区域')
tree1.heading('dz', text='地址')
# 定义3个栏目的表头文字

sf1 = tree1.insert('', END, text='广东', open=True)
sf2 = tree1.insert('', END, text='湖南', open=True)
# 在根节点‘’下添加2个子节点：广东，湖南

tree1.insert(sf1, END, text='广州市', values=('海珠区', '阅江中路380号'))
tree1.insert(sf1, END, text='深圳市', values=('南山区', '华侨城侨香路11
号'))
tree1.insert(sf1, END, text='东莞市', values=('南城区', '元美东路3号济亨
网'))
# 在广州（sf1）节点下，插入3条记录：#0栏 = text,其它栏 = values()

tree1.insert(sf2, END, text='长沙市', values=('雨花区', '韶山中路108号'))
tree1.insert(sf2, END, text='湘潭市', values=('岳塘区', '书院路42号云峰工
作室'))
tree1.insert(sf2, END, text='衡阳市', values=('蒸湘区', '祝融路名都花园B9
栋107室'))
# 在湖南（sf2）节点下，插入3条记录：#0栏 = text,其它栏 = values()

tree1.insert(sf2, END, text='长沙市', values=('岳麓区', '梅溪湖路复兴小区
709号'))
tree1.insert(sf1, END, text='广州市', values=('白云区', '下塘西路545号'))
# 以同样方法插入2条记录，它们会根据父节点找到自己的位置
```

```
tree1.pack(fill=BOTH, expand=True)
# 滚动条相关
window.mainloop()
```