第7章 GUI编程

李宇

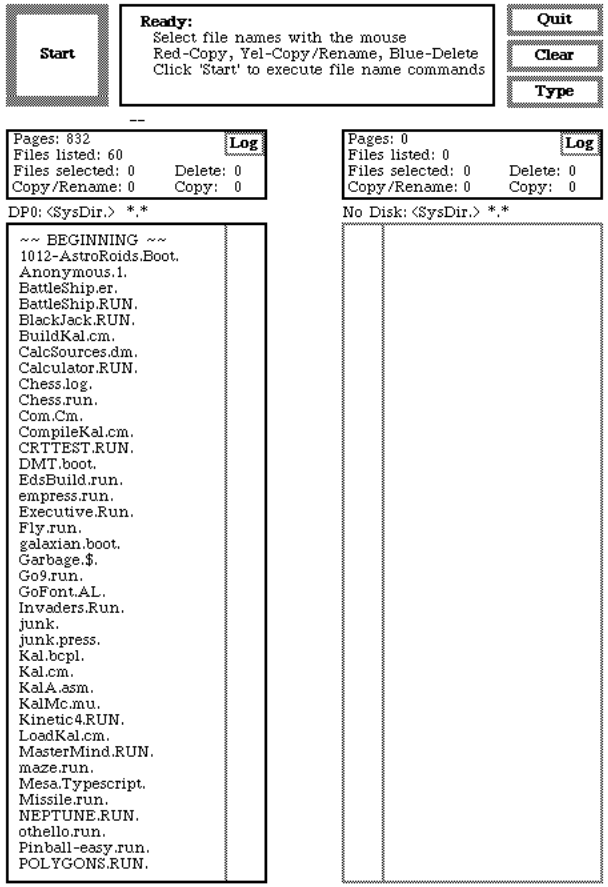
东北大学-计算机学院-智慧系统实验室

liyu@cse.neu.edu.cn

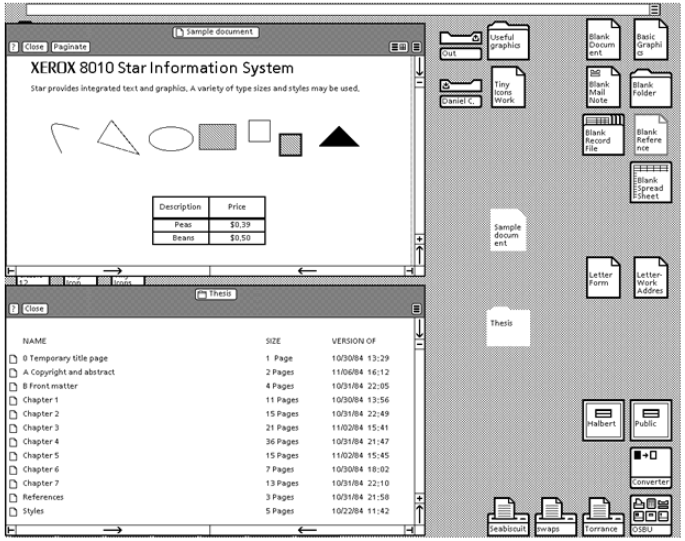
GUI(Graphical User Interface)

- GUI：用户图形界面
  - 早期计算机屏幕
    - 显示纯文本
    - 由键盘输入命令行来控制
  - 1973年，Xerox推出了第一款GUI：计算机 Alto
  - 1979年，Steve Jobs参观时看到了Alto
  - 1984年为Macintosh发布了GUI操作系统
  - 1985年微软推出了windows1.0

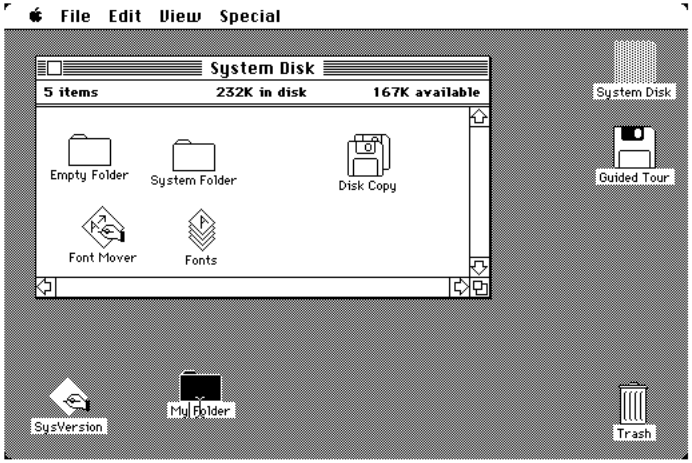
早期的GUI系统



Alto (1973)



Xerox 8010 Star(1981)



Mac OS System 1.0(1984)

Tkinter库

- Python的自带GUI库
  - 不需下载安装
  - 简单
  - 适合开发小工具
  - 支持16个**核心**窗口控件/组件
    - Button（按钮）、Canvas（画布）、checkbutton（复选框）、Entry（单行文本框）、Frame（框架）、Label（标签）、Listbox（列表框）、Menu（菜单）、Menubutton（菜单按钮）、Message（消息框）、Radiobutton（选择菜单）、Scale（进度条）、Scrollbar（滚动条）、Text（多行文本框）、Toplevel（顶层）、MessageBox（消息框）
  - 每个典型的窗口组件类提供了约150种方法

创建主窗口及Label控件

```
import tkinter as tk  # 使用tkinter前需要先导入

window = tk.Tk()  #实例化一个窗口
window.title('My Window') #给窗口起名字
window.geometry('500x300') #设置窗口的长和宽
l = tk.Label(window, text='你好! this is Tkinter',
bg='green', font=('Arial', 12), width=30,
height=2)  #创建一个标签，设置标签的主体是window窗口，设置标签的参数

l.pack()  #放置标签，自动调节尺寸
window.mainloop()  #主窗口循环显示，窗口对象必须调用mainloop函数，是窗口的关键!
```

Button控件

- 按钮可以是文本或者图像，将按钮与一个Python函数或方法关联，当按钮按下时，tkinter自动调用这个函数或方法

```
var = tk.StringVar()
l = tk.Label(window, textvariable=var, bg='green', fg='white',
font=('Arial', 12), width=30, height=2)
on_hit = False
def hit_me():
    global on_hit
    if on_hit == False:
        on_hit = True
        var.set('you hit me')
    else:
        on_hit = False
        var.set('')
b = tk.Button(window, text="hit me", command=hit_me)
b.pack()
```

字体颜色

自定义函数，点击按钮时调用

把自定义函数名传给command参数

另一种事件处理机制 bind()

- 除了通过给command属性传入指定的动作，Tkinter还可能使用方法bind()

```
import tkinter as tk
window = tk.Tk()
window.title('My Window')
window.geometry('500x300')

def printLabel(event):#定义事件函数
    x1=tk.Label(window,text='我是一个Label!',background='pink')
    x1.pack()

b = tk.Button(window,text='单击左键试试')#定义一个按钮
b.bind('<Button-1>', printLabel)#单击鼠标左键，绑定printLabel()函数
b.pack()

window.mainloop()
```

属性	描述
activebackground	按钮按下时背景颜色。默认是系统指定的颜色。
activeforeground	按钮按下时前景颜色。默认是系统指定的颜色。
anchor	采用何种方式锚定文字或者图片。默认是CENTER（居中模式）。可以选择如下方式： N, NE, E, SE, S, SW, W, NW, or CENTER.
background bg	按钮的背景颜色。默认是系统指定颜色
bitmap	bitmap形式显示按键。如果设置了images属性，则忽略bitmap属性。
borderwidth bd	按钮的边缘宽度。一般是1~2个像素值。
command	设置回调函数。当按钮被按下时，会调用该函数。如果该属性没有设置，按下按钮时，不会有任何动作发生。
compound	在按钮上同时显示文字和图片。默认的模式是如果提供了图片，会只显示图片。但是，如果将选项设为： CENTER：在图片中间叠加显示文字 BOTTOM：在图片下方显示文字 LEFT：在图片左边显示文字 RIGHT：在图片右边显示文字 TOP：在图片顶部显示文字 NONE：不显示文字
cursor	当鼠标移动经过按钮的时候，显示光标
default	取值有normal,active和disabled三个。
disabledforeground	按钮被禁止使用时，按时上的文字的颜色
font	按钮上文字的字体。只能选择一种字体显示。
foreground fg	按钮上文字或者位图的颜色
height	设置按钮的高度。如果是显示文字，数值是文字单位。如果是显示图片，数值单位为像素。如果没有设置，系统自动计算按钮的高度。
highlightbackground	当按钮失去焦点的时候，显示按钮边框的高亮颜色
highlightcolor	当按钮获得焦点的时候，显示按钮边框的高亮颜色
highlightthickness	设置高亮边框的宽度
image	设置按钮显示的图片。如果该选项被设置，会取代text或bitmap选项。
justify	当按钮有多行文字时，设置文字的对齐方式。可设的数值有： LEFT,RIGHT,CENTER
overrelief	当鼠标移动经过的时候，按钮显示浮雕效果。如果没有设定，会使用relief中的值。
padx	在水平方向上，按钮边框和文字或图像之间的填充（pad）
pady	在垂直方向上，按钮边框和文字或图像之间的填充（pad）
relief	按钮3D美化效果。通常情况下，按钮被按下时，是SUNKEN效果，释放时是RAISED效果。其他的可选项包括：GROOVE，RIDGE以及FLAT
state	按钮的状态，包括：NORMAL, ACTIVE 或者 DISABLED
takefocus	标识用户是否能够使用Tab键选中按钮。取值是True，False或者None
text	显示在按钮上的文字。如果使用了bitmap或者image，该选项被忽略
textvariable	将tkinter变量与按钮相关联。如果变量的内容发生变化，按钮的文字也随之更新。
underline	标识在那个字符下显示下划线。默认值是-1，标识没有字符下面显示下划线
width	设置按钮的宽度。参考height属性
wraplength	确定按钮上文字超过多长时，文字会被折叠成多行。单位是像素。默认值是0.

Entry控件

- 单行文本输入域，输入显示一行文本，用来收集键盘输入（用户名、密码）

```
import tkinter as tk

window = tk.Tk()
window.title('My Window')
window.geometry('500x300')
#设定输入框控件entry
e1 = tk.Entry(window, show='*', font=('Arial', 14))
e2 = tk.Entry(window, show=None, font=('Arial', 14))

#放置控件
e1.pack()
e2.pack()

window.mainloop()
```

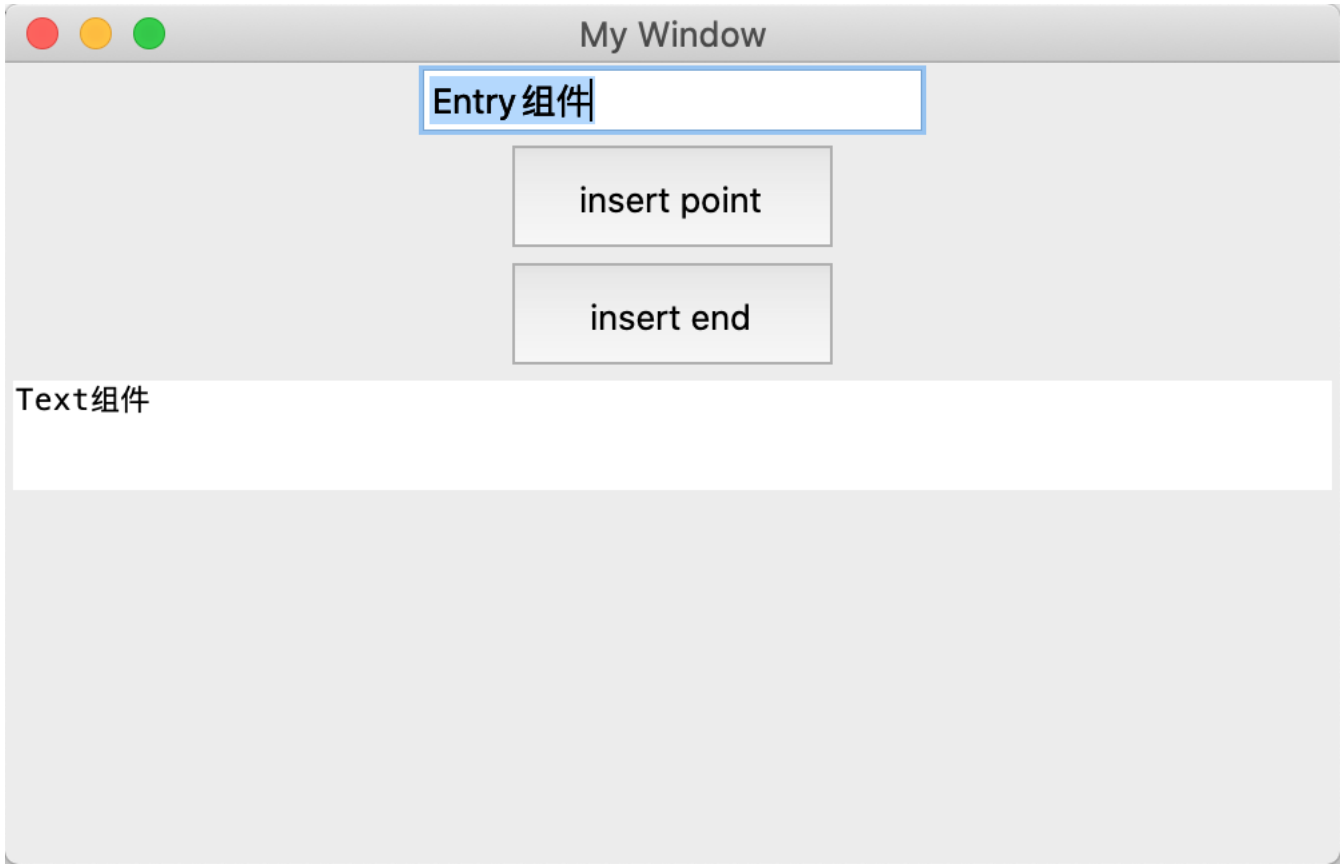
#用关键字show='*'显示成密文

#用关键字show=None显示成明文

Text控件

- Text：多行文本区域，收集/显示键盘输入的多行文字

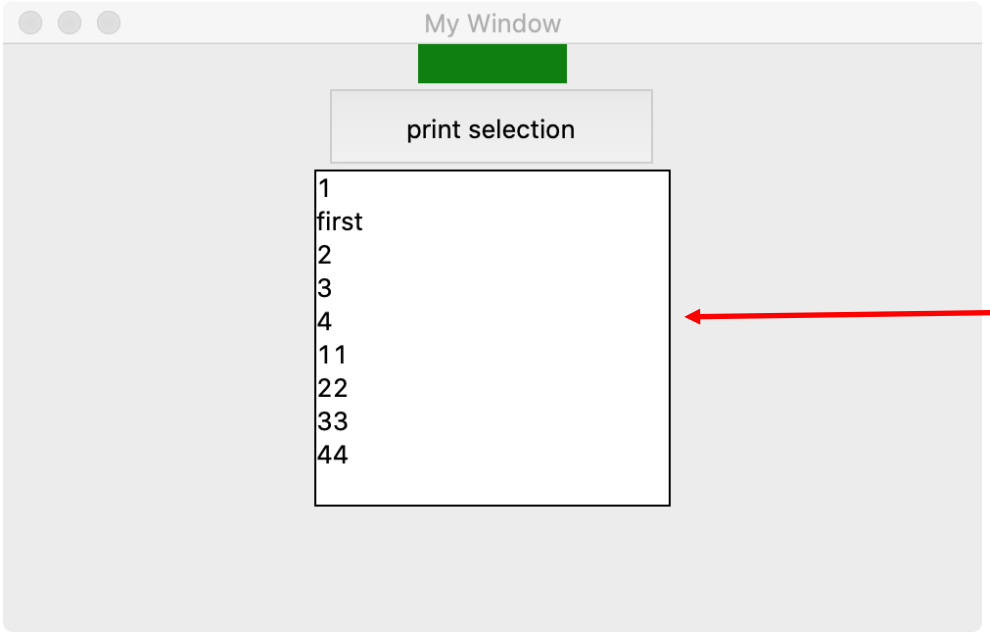
```
t = tk.Text(window, height=3)
t.pack()
```



ListBox控件

- ListBox控件:

```
var2 = tk.StringVar()
var2.set(('1','2',3,4)) # 为变量var2设置值
# 创建Listbox
lb = tk.Listbox(window, listvariable=var2) #将var2的值赋给Listbox
lb.pack()
```



RadioButton控件

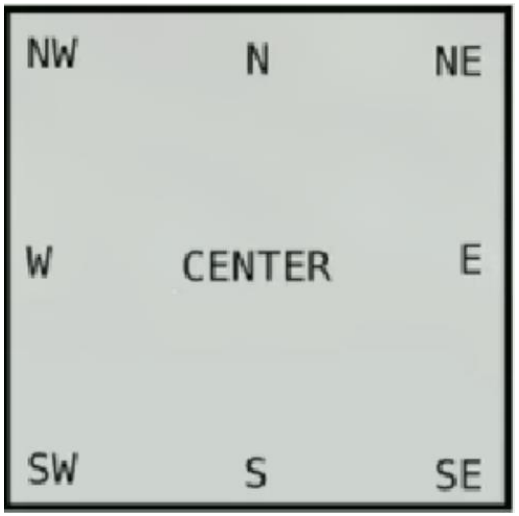
```
var = tk.StringVar()# 定义var用来将radiobutton的值和Label的值联系起来
l = tk.Label(window, bg='yellow', width=20, text='empty')
l.pack() #刚开始时label显示‘empty’

# 定义选项触发函数功能，config()方法配置label里的text
def print_selection():
    l.config(text='you have selected ' + var.get())

# 创建三个radiobutton选项和其响应函数
r1 = tk.Radiobutton(window, text='Option A', variable=var,
value='A', command=print_selection)
r2 = tk.Radiobutton(window, text='Option B', variable=var,
value='B', command=print_selection)
r3 = tk.Radiobutton(window, text='Option C', variable=var,
value='C', command=print_selection)
```

Canvas控件

- 提供绘图功能(直线、椭圆、多边形、矩形等)，可以包含图形或位图，用来绘制图表和图，创建图形编辑器

```
canvas = tk.Canvas(window, bg='green', height=200, width=500)
# 说明图片路径，并导入图片到画布上
image_file = tk.PhotoImage(file='pic.gif')
image = canvas.create_image(250, 0, anchor='n',image=image_file)
```

图片锚定点位置参数图：

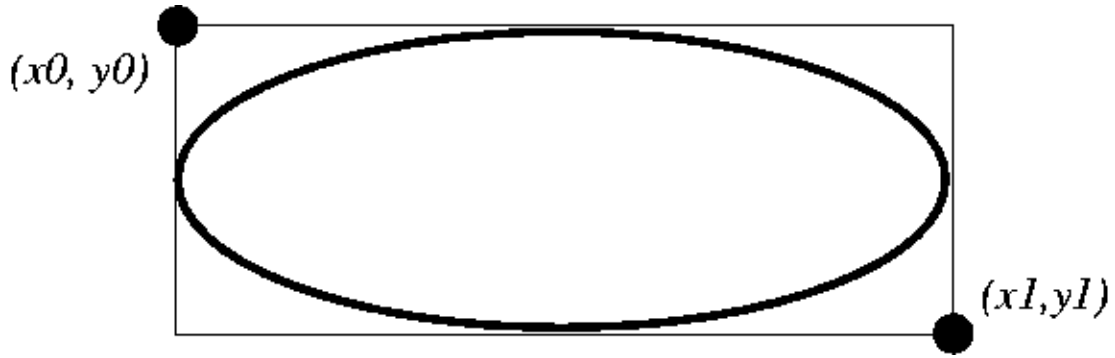


Canvas控件

```
x0, y0, x1, y1 = 100, 100, 150, 150
line = canvas.create_line(x0-50, y0-50, x1-50, y1-50)      # 画直线
oval = canvas.create_oval(x0+120, y0+50, x1+120, y1+50, fill='yellow')  # 画圆 黄色
arc = canvas.create_arc(x0, y0+50, x1, y1+50, start=0, extent=180)
# 画扇形 从0度打开收到180度结束
rect = canvas.create_rectangle(330, 30, 330+20, 30+20)              # 画矩形正方形
canvas.pack()

# 触发函数，用来一定指定图形
def moveit():
    canvas.move(rect, 2, 2) # 移动正方形rect，按每次（x=2, y=2）步长进行移动

# 定义一个按钮用来移动指定图形的在画布上的位置
b = tk.Button(window, text='move item',command=moveit).pack()
```

create_oval的原理

- 根据两点确定的矩形，然后画一个唯一的内切的椭圆
  - oval = canvas.create_oval(x0+120, y0+50, x1+120, y1+50, fill='yellow’)



  - create_arc()方法也是用同样的原理来确定圆弧的

Menu

```
# 在图形界面上创建一个标签用以显示内容并放置
l = tk.Label(window, text='我是一个标签', bg='green')
l.pack()

# 定义一个函数功能，用来代表菜单选项的功能
counter = 1
def do_job():
    global counter
    l.config(text='点了 '+ str(counter) + '下菜单')
    counter += 1

# 创建一个菜单栏，这里我们可以把他理解成一个容器，在窗口的上方
menubar = tk.Menu(window)

# 在创建的menubar里创建一个File菜单项（默认不下拉，下拉内容包括New，Open，Save，Exit功能项）
filemenu = tk.Menu(menubar, tearoff=0)
# 将上面定义的空菜单命名为File，放在菜单栏中，就是装入那个容器中
menubar.add_cascade(label='File', menu=filemenu)

# 在File中加入New、Open、Save等小菜单，即我们平时看到的下拉菜单，每一个小菜单对应命令操作。
filemenu.add_command(label='New', command=do_job)
filemenu.add_command(label='Open', command=do_job)
filemenu.add_separator()     # 添加一条分隔线
filemenu.add_command(label='Exit', command=window.quit) # 用tkinter里面自带的quit()函数

# 创建菜单栏完成后，配置让菜单栏menubar显示出来
window.config(menu=menubar)
# 主窗口循环显示
window.mainloop()
```

Frame控件

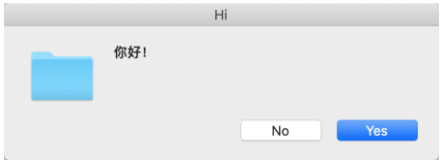
- Frame-框架，用来承载放置其他GUI元素，是一个容器，是一个在 Window 上分离小区域的部件, 它能将 Window分成不同的区,然后存放不同的其他部件. Frame可以再分成两个Frame

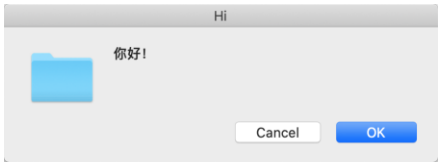
```
tk.Label(window, text='I am a Label', bg='red', font=('Arial', 16)).pack()    #创建+放置标签，一步到位

# 创建一个主frame，长在主window窗口上
frame = tk.Frame(window)
frame.pack()

# 创建第二层框架frame，长在主框架frame上面
frame_l = tk.Frame(frame)# 第二层frame，左frame，长在主frame上
frame_r = tk.Frame(frame)# 第二层frame，右frame，长在主frame上
frame_l.pack(side='left')
frame_r.pack(side='right')

# 创建三组标签，为第二层frame上面的内容，分为左区域和右区域，用不同颜色标识
tk.Label(frame_l, text='on the frame_l1', bg='green').pack()
tk.Label(frame_l, text='on the frame_l2', bg='green').pack()
tk.Label(frame_l, text='on the frame_l3', bg='green').pack()
tk.Label(frame_r, text='on the frame_r1', bg='yellow').pack()
tk.Label(frame_r, text='on the frame_r2', bg='yellow').pack()
tk.Label(frame_r, text='on the frame_r3', bg='yellow').pack()

# 窗口循环显示
window.mainloop()
```

MessageBox

- 消息框，也就是弹窗， 需要定义一个触发功能，来触发这个弹窗

```
import tkinter as tk

window = tk.Tk()
window.title('My Window')
window.geometry('500x300')

# 定义触发函数功能
def hit_me():
    #tkinter.messagebox.showinfo(title='Hi', message='你好！')                # 提示信息对话窗
    tkinter.messagebox.showwarning(title='Hi', message='有警告！')          # 警告对话窗
    # tkinter.messagebox.showerror(title='Hi', message='出错了！')            # 错误对话窗
    # print(tkinter.messagebox.askquestion(title='Hi', message='你好！'))    # 选择对话窗
    # print(tkinter.messagebox.askyesno(title='Hi', message='你好！'))       # return 'True'/'False'
    # print(tkinter.messagebox.askokcancel(title='Hi', message='你好！'))    # return 'True'/'False'
# 在图形界面上创建一个按钮用以显示内容并放置
tk.Button(window, text='hit me', bg='green', font=('Arial', 14), command=hit_me).pack()

window.mainloop()
```





窗口组件的部署方式

- pack()
  - 把控件放在其主控件中
  - 会按照上下左右的方式排列

```
tk.Label(window, text='P', fg='red').pack(side='top')     # 上
tk.Label(window, text='P', fg='red').pack(side='bottom')  # 下
tk.Label(window, text='P', fg='red').pack(side='left')    # 左
tk.Label(window, text='P', fg='red').pack(side='right')   # 右
```

窗口组件的部署方式

- grid()
  - grid 是方格, 所以所有的内容会被放在这些规律的方格中

```
for i in range(3):
    for j in range(3):
        tk.Label(window, text="标签"+str(i)+str(j)).grid(row=i,
column=j, padx=10, pady=10)
```

创建一个三行三列的表格，padx 就是单元格左右间距，pady 就是单元格上下间距

窗口组件的部署方式

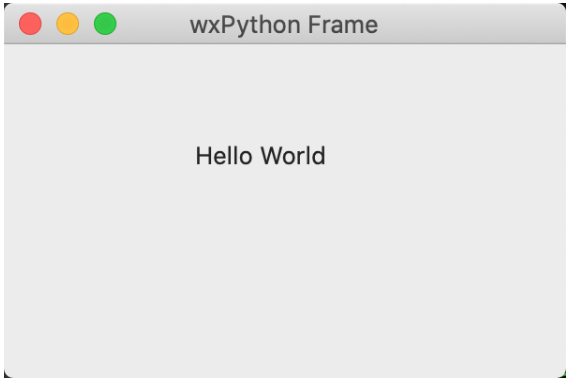
- place()
  - 用x, y, anchor参数给组件**精确定位**

```
import tkinter as tk
window = tk.Tk()
window.title('My Window')
window.geometry('500x300')

# place 放置方法（精准的放置到指定坐标点的位置上）
l=tk.Label(window, text='Pl', font=('Arial', 20), )
l.place(x=50, y=100, anchor='nw')

window.mainloop()
```

另一个GUI模块wxPython

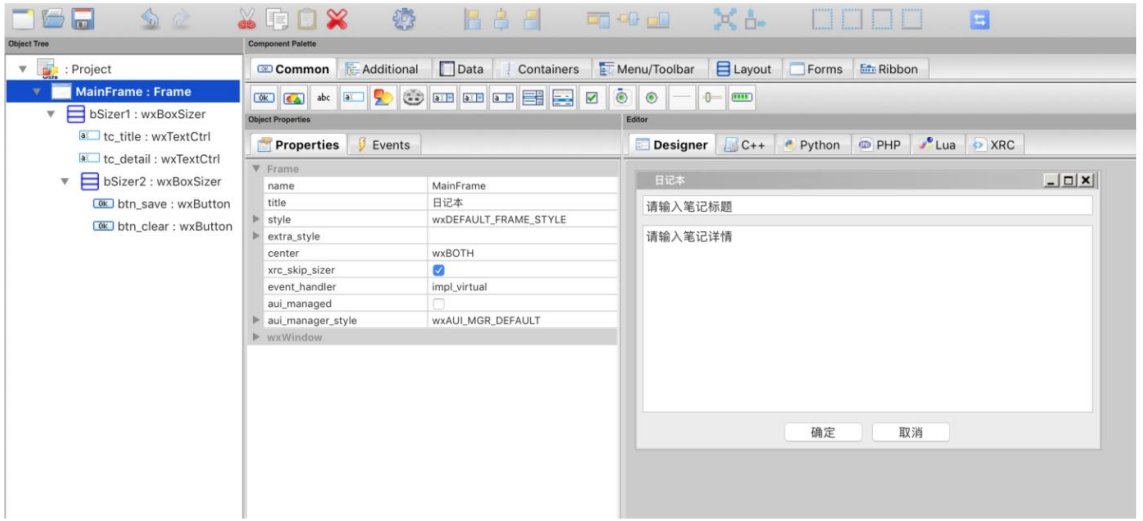
- 安装命令：pip install -U wxPython
  - 适用于windows和mac

```
import wx  #导入wx模块
app = wx.App() #定义Application类的对象
#创建一个顶级窗口作为wx.Frame类的对象。 标题和大小参数在构造函数中给出。
window = wx.Frame(None, title = "wxPython Frame", size = (300,200))
#虽可以在Frame对象中添加其他控件，但无法管理布局。 因此，将Panel对象放入Frame
panel = wx.Panel(window)
#添加一个StaticText对象，在窗口内的所需位置显示“Hello World”。
label = wx.StaticText(panel, label = "Hello World", pos = (100,50))
# 通过show()方法激活框架窗口
window.Show(True)
# Application对象的主事件循环
app.MainLoop()
```



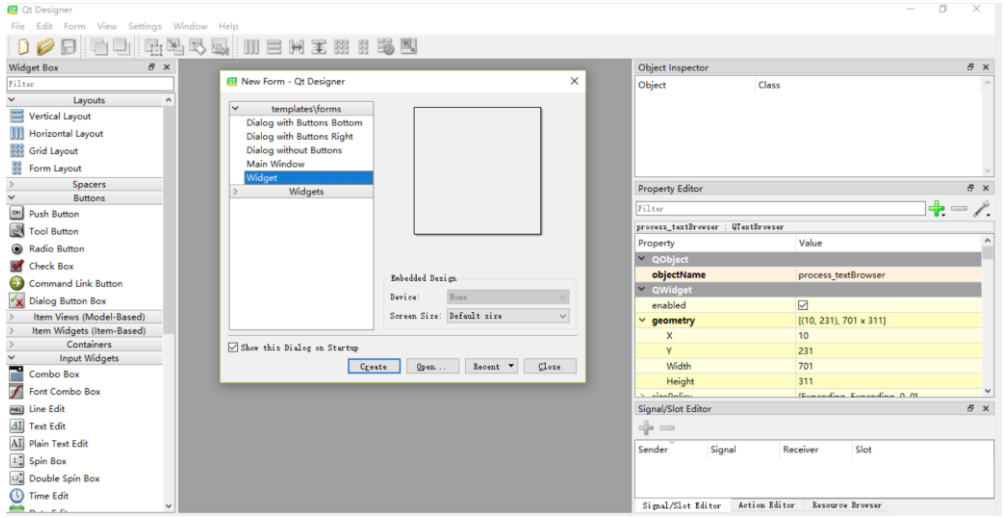
一些wxPython的GUI工具

- wxFormBuilder
  - 免费开源，跨平台，可将GUI界面生成C++，Python，PHP或XML代码
  - 自动生成部分代码，帮助你完成界面设计
- wxDesigner
- wxGlade
- BoaConstructor
- gui2py



PyQt

- PyQt5是Qt5的Python版本，功能强大复杂，提供QT Designer设计UI （GPL V3协议，开源，商用收费）
  - 使用pip工具安装PyQt5工具。在命令行里执行 **pip install PyQt5**
  - 安装Qt Designer图形界面开发工具。执行**pip install PyQt5-tools**
  - 需配置环境变量PyQt5-tools的安装目录添加到系统环境变量path



感谢