【Python】Tkinter图形界面设计（GUI）

简介
作为 Python 开发者，图形用户界面（GUI）开发是必备技能之一。目前，市面上支持 Python 的“GUI 工具包”很多，各有特点，虽然大多数工具包的基础类似，但要学习一个新包并掌握其细节还是非常耗时的，因此，在选用工具包时应仔细权衡。本文将介绍 Python 自带的 GUI 工具包 TKinter。

TKinter
Python 的 GUI 库非常多，之所以选择 Tkinter，一是最为简单，二是自带库，不需下载安装，随时使用，跨平台兼容性非常好，三则是从需求出发的，Python 在实际应用中极少用于开发复杂的桌面应用，毕竟，Python 的各种 GUI 工具包都“一般得很”，不具备优势。
关于 GUI，泛泛而谈难免枯燥，鉴于此，本文将基于一系列实例来介绍 Tkinter 控件。

窗口创建与布局
做界面，首先需要创建一个窗口，Python Tkinter 创建窗口很简单，代码如下：

```
from tkinter import *
#初始化Tk()
myWindow = Tk()
#进入消息循环
myWindow.mainloop()
```

上述程序创建的窗口是非常简陋的，有待进一步美化，设置标题、窗口大小、窗口是否可变等，涉及属性有：title（设置窗口标题）、 geometry（设置窗口大小）、 resizable（设置窗口是否可以变化长 宽）。请看如下实例：

```
from tkinter import Tk
#初始化Tk()
myWindow = Tk()
#设置标题
myWindow.title('Python GUI Learning')
#设置窗口大小
myWindow.geometry('380x300')
#设置窗口是否可变长、宽，True：可变，False：不可变
myWindow.resizable(width=False, height=True)
#进入消息循环
myWindow.mainloop()
```

进一步，将窗口放置于屏幕中央，如下实例：

```
from tkinter import Tk
#初始化Tk()
myWindow = Tk()
#设置标题
myWindow.title('Python GUI Learning')
#设置窗口大小
width = 380
height = 300
#获取屏幕尺寸以计算布局参数，使窗口居屏幕中央
screenwidth = myWindow.winfo_screenwidth()
screenheight = myWindow.winfo_screenheight()
alignstr = '%dx%d+%d+%d' % (width, height, (screenwidth-width)/2, (screenheight-height)/2)
myWindow.geometry(alignstr)
#设置窗口是否可变长、宽，True：可变，False：不可变
myWindow.resizable(width=False, height=True)
#进入消息循环
myWindow.mainloop()
```

常用控件
仅有窗口并不能实现交互，还需要控件，Tkinter 提供了各种控件，如按钮、标签和文本框。在一个 GUI 应用程序中使用，这些控件通常被称为控件或者部件，目前有15种 Tkinter 部件，如下列表：

几何管理
Tkinter 控件有特定的几何状态管理方法，管理整个控件区域组织，以下是 Tkinter 公开的几何管理类：包、网格、位置。

Lable控件标签控件，基本用法为： Lable(root, option...) ，即：Label(根对象, [属性列表]),
其中属性列表如下：
Lable 控件实例
实例1：标签展示文本，代码如下：

```
from tkinter import*
#初始化Tk()
myWindow = Tk()
#设置标题
```

```
myWindow.title('Python GUI Learning')
#创建一个标签，显示文本
Label(myWindow, text="user-name",bg='red',font=('Arial 12 bold'),width=20,height=5).pack()
Label(myWindow, text="password",bg='green',width=20,height=5).pack()
#进入消息循环
myWindow.mainloop()
```

执行结果：
实例2：标签展示图标，代码如下：

```
from tkinter import*
#初始化Tk()
myWindow = Tk()
#设置标题
myWindow.title('Python GUI Learning')
#创建一个标签，显示图标
logo = PhotoImage(file="/Users/guojin/book/temp.gif")
Label(myWindow, image=logo).pack(side='left')
#进入消息循环
myWindow.mainloop()
```

运行结果：
实例3：标签图文混叠，边距控制，代码如下：

```
from tkinter import*
#初始化Tk()
myWindow = Tk()
#设置标题
myWindow.title('Python GUI Learning')
#创建一个标签，显示文本
logo = PhotoImage(file="/Users/guojin/book/temp.gif")
explanation = """At present, only GIF and PPM/PGM
formats are supported, but an interface
exists to allow additional image file
formats to be added easily."""
Label(myWindow,compound=CENTER,text=explanation,image=logo).pack(side="right")
#进入消息循环
myWindow.mainloop()
```

运行结果：

Button控件
Button 控件是一个标准的 Tkinter 部件，用于实现各种按钮。按钮可以包含文本或图像，还可以关联 Python 函数。
Tkinter 的按钮被按下时，会自动调用该函数。
按钮文本可跨越一个以上的行。此外，文本字符可以有下划线，例如标记的键盘快捷键。默认情况下，使用 Tab 键可以移动到一个按钮部件，用法如下：
Entry(根对象, [属性列表])，即Entry(root, option...)
常用的属性列表如下：
Button 实例：
实例1：创建按钮，代码如下：

```
from tkinter import*
#初始化Tk()
myWindow = Tk()
#设置标题
myWindow.title('Python GUI Learning')
#创建两个按钮
b1=Button(myWindow, text='button1',bg="red", relief='raised', width=8, height=2)
b1.grid(row=0, column=0, sticky=W, padx=5,pady=5)
b2=Button(myWindow, text='button2', font=('Helvetica 10 bold'),width=8, height=2)
b2.grid(row=0, column=1, sticky=W, padx=5, pady=5)
#进入消息循环
myWindow.mainloop()
```

运行结果：
实例2：创建按钮并绑定响应函数，输入半径，计算圆面积并输出，代码如下：

```
from tkinter import*
def printInfo():
    #清理entry2
    entry2.delete(0, END)
    #根据输入半径计算面积
    R=int(entry1.get())
    S= 3.1415926*R*R
    entry2.insert(10, S)
```

```
        #清空entry2控件
        entry1.delete(0, END)
    #初始化Tk()
    myWindow = Tk()
    #设置标题
    myWindow.title('Python GUI Learning')
    #标签控件布局
    Label(myWindow, text="input").grid(row=0)
    Label(myWindow, text="output").grid(row=1)
    #Entry控件布局
    entry1=Entry(myWindow)
    entry2=Entry(myWindow)
    entry1.grid(row=0, column=1)
    entry2.grid(row=1, column=1)
    #Quit按钮退出；Run按钮打印计算结果
    Button(myWindow, text='Quit', command=myWindow.quit).grid(row=2, column=0,sticky=W, padx=5, pady=5)
    Button(myWindow, text='Run', command=printInfo).grid(row=2, column=1, sticky=W, padx=5, pady=5)
    #进入消息循环
    myWindow.mainloop()
```

运行结果：
输入半径：
点击‘Run’计算面积：

Checkbutton控件Checkbutton 是复选框，又称为多选按钮，可以表示两种状态。用法为： Checkbutton (root, option, ...)， 其中可选属性 option 有很多，如下表所示：

以下是这个小工具的常用方法：
实例1：创建一组复选框，代码如下：

```
from tkinter import*
#初始化Tk()
myWindow = Tk()
#设置标题
myWindow.title('Python GUI Learning')
# 用来获取复选框是否被勾选，通过chVarDis.get()来获取其的状态,其状态值为int类型 勾选为1
chVarDis = IntVar()
# text为该复选框后面显示的名称, variable将该复选框的状态赋值给一个变量，当state='disa
check1 = Checkbutton(myWindow, text="Disabled", variable=chVarDis, state='disabled')
# 该复选框是否勾选,select为勾选, deselect为不勾选
check1.select()
# sticky=tk.W 当该列中其他行或该行中的其他列的某一个功能拉长这列的宽度或高度时，
# 设定该值可以保证本行保持左对齐，N：北/上对齐 S：南/下对齐 W：西/左对齐 E：东/右对齐
check1.grid(column=0, row=0, sticky=W)
chvarUn = IntVar()
check2 = Checkbutton(myWindow, text="UnChecked", variable=chvarUn)
check2.deselect()
check2.grid(column=1, row=0, sticky=W)
chvarEn = IntVar()
check3 = Checkbutton(myWindow, text="Enabled", variable=chvarEn)
check3.select()
check3.grid(column=2, row=0, sticky=W)
#进入消息循环
myWindow.mainloop()
```

实例2：绑定响应函数，代码如下：

```
from tkinter import*
def callCheckbutton():
    #改变v的值，即改变Checkbutton的显示值
    v.set('check CheckButton')
#初始化Tk()
myWindow = Tk()
#设置标题
myWindow.title('Python GUI Learning')
v = StringVar()
v.set('check python')
#绑定v到Checkbutton的属性textvariable
Checkbutton(myWindow,textvariable = v,command = callCheckbutton).pack()
#进入消息循环
myWindow.mainloop()
```

Radiobutton控件

单选按钮是一种可在多个预先定义的选项中选择出一项的 Tkinter 控件 。单选按钮可显示文字或图片，显示文字时只能使用预设字体，该控件可以绑定一个 Python 函数或方法，当单选按钮被选择时，该函数或方法将被调用。

单选按钮（Radio Button）这个名字来源于收音机（Radio）上的调频按钮， 这些按钮用来选择特定波段或预设电台，如果一个按钮被按下， 其他同类的按钮就会弹起，即同时只有一个按钮可被按下。

一组单选按钮控件和同一个变量关联。点击其中一个单选按钮将把这个变量设为某个预定义的值。一般用法为： Radiobutton(myWindow，options)，其中 option 与 Checkbutton，Button 大多重合，用法一致。

实例：创建单选按钮并绑定响应函数，代码如下：

```
from tkinter import*
#初始化Tk()
myWindow=Tk()
myWindow.title('Python GUI Learning')
v=IntVar()
#列表中存储的是元素是元组
language=[('python',0),('C++',1),('C',2),('Java',3)]
#定义单选按钮的响应函数
def callRB():
    for i in range(4):
    if (v.get()==i):
        root1 = Tk()
        Label(root1,text='你的选择是'+language[i][0]+'!',fg='red',width=20, height=6).pack()
        Button(root1,text='确定',width=3,height=1,command=root1.destroy).pack(side='bottom')
Label(myWindow,text='选择一门你喜欢的编程语言').pack(anchor=W)
#for循环创建单选框
for lan,num in language:
    Radiobutton(myWindow, text=lan, value=num, command=callRB, variable=v).pack(anchor=W)
#进入消息循环
myWindow.mainloop()
```

运行结果：

Menu控件

Menu被用来创建一个菜单，创建Menu类的实例，然后使用add方法添加命令或者其他菜单内容。使用方法如下：

Menu(root,option,…)

其中 option 列表如下：

特有函数如下：

实例：创建一个菜单组，代码如下：

```
from tkinter import *
#创建窗口
myWindow=Tk()
myWindow.title("菜单")
myWindow.geometry("400x300+300+100")
# 创建一个菜单项，类似于导航栏，顶层菜单
menubar=Menu(myWindow)
# 创建菜单项
fmenu1=Menu(myWindow)
for item in ['新建','打开','保存','另存为']:
    # 如果该菜单是顶层菜单的一个菜单项，则它添加的是下拉菜单的菜单项。则他添加的是下拉菜单的菜单项。
    fmenu1.add_command(label=item)

fmenu2=Menu(myWindow)
for item in ['复制','粘贴','剪切']:
    fmenu2.add_command(label=item)

fmenu3=Menu(myWindow)
for item in ['大纲视图','web视图']:
    fmenu3.add_command(label=item)

fmenu4=Menu(myWindow)
for item in ["版权信息","其它说明"]:
    fmenu4.add_command(label=item)

# add_cascade 的一个很重要的属性就是 menu 属性，它指明了要把那个菜单级联到该菜单项上，
# 当然，还必不可少的就是 label 属性，用于指定该菜单项的名称
menubar.add_cascade(label="文件",menu=fmenu1)
menubar.add_cascade(label="编辑",menu=fmenu2)
menubar.add_cascade(label="视图",menu=fmenu3)
menubar.add_cascade(label="关于",menu=fmenu4)
```

```
# 最后可以用窗口的 menu 属性指定我们使用哪一个作为它的顶层菜单
myWindow.config(menu=menubar)
#进入消息循环
myWindow.mainloop()
```

运行结果:

Message控件
Message 控件用来展示一些文字短消息。Message 和 Label 控件有些类似， 但在展示文字方面比 Label 要灵活，比如 Message 控件可以改变字体，而 Label 控件只能使用一种字体，它提供了一个换行对象，以使文字可以断为多行。
它可以支持文字的自动换行及对齐，这里要澄清一下前面提到的 Message 控件可以改变字体的说法: 这是说我们可以为单个控件设置任意字体, 控件内的文字都将显示为该字体，但我们不能给单个控件内的文字设置多种字体，如果你需要这么做，可以考虑使用 Text 控件。
创建一个 Message 控件的语法如下:
w = Message (root, option, ...)
其中 option 列表如下:
请看下面实例:

```
from tkinter import *
#初始化Tk()
myWindow=Tk()
#创建一个Message
whatever_you_do = "Only those who have the patience to do simple things per
msg = Message(myWindow, text = whatever_you_do)
msg.config(bg='lightgreen', font=('times', 20, 'italic'))
msg.pack( )
#进入消息循环
myWindow.mainloop()
```