thinter(Tk interface) 是python的标准GUI库，支持跨平台GUI程序开发，适合小型GUI程序编写。

wxPython是比较流行的GUI库，适合大型应用程序开发，整体设计框架类似于MFC(Microsoft Foundation Classes)微软基础类库。

PyQT

Qt是一种开源的GUI库，适合大型应用程序开发。PyQT是Qt工具包标准的Python实现。可以使用Qt Desginer界面设计器快速开发GUI应用程序。

<https://docs.python.org/3.7/library/tk.html>

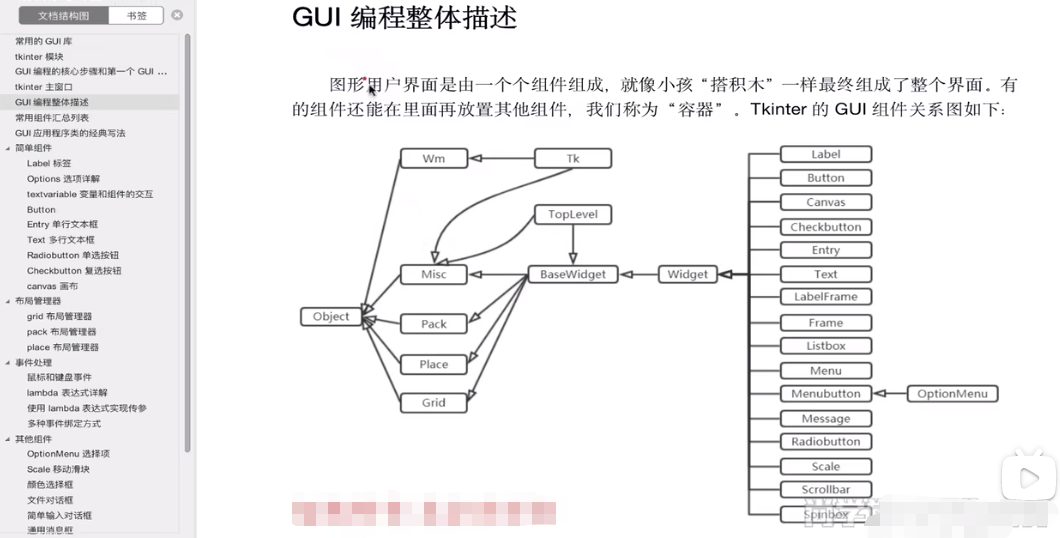
<http://effbot.org/tkinterbook>

PEP8编码规范

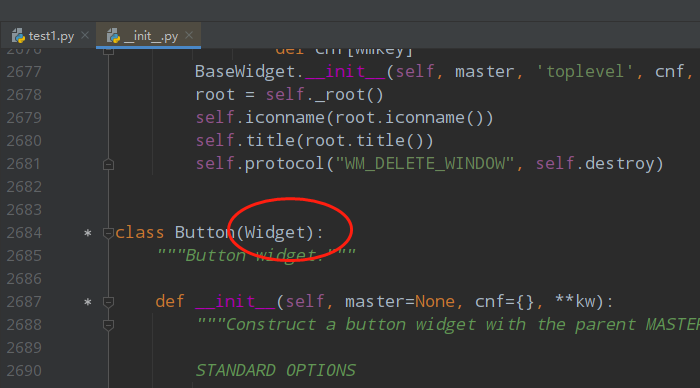
<https://www.python.org/dev/peps/pep-0008>

■thinter库

●thinter组件类的继承关系图



在pycharm中，可以按住Ctrl加组件名，来查看继承关系。例如下图，Button继承了Widget



●

常用组件汇总表



●第一个GUI程序

from tkinter import \*  
from tkinter import messagebox  
  
  
root = Tk()  
  
root.title("我的第一个GUI程序")  
# geometry(窗口宽度x窗口高度+左侧距离+上方距离) -右侧距离-下方距离  
root.geometry("500x300+200+100")  
  
btn01 = Button(root)  
btn01["text"] = "点我就送花"  
  
btn01.pack()  
  
  
def songhua(e): # e是事件对象  
 messagebox.showinfo("Message", "送你一朵小红花")  
 print("后台打印，送了一朵花")  
  
  
btn01.bind("<Button-1>", songhua) # 给btn01绑定一个鼠标点击事件  
  
root.mainloop() # 调用组件的mainloop方法，进入事件循环

●官方给的经典写法，使用面向对象的方式



from tkinter import \*  
from tkinter import messagebox  
  
  
class Application(Frame):  
 *"""一个经典的GUI程序的类的写法"""* def \_\_init\_\_(self, master=None):  
 super().\_\_init\_\_(master) # super调用父的初始化构造方法  
 self.master = master  
 self.pack()  
  
 self.createWidget()  
  
 def createWidget(self):  
 *"""在这里创建所有组件"""* # 创建一个按钮  
 self.btn01 = Button(self)  
 self.btn01["text"] = "点击送花"  
 self.btn01.pack()  
 self.btn01["command"] = self.songhua  
  
 # 再创建一个按钮  
 self.btn02 = Button(self, text="点击打招呼", command=self.sayhi)  
 self.btn02.pack()  
  
 # 创建一个退出按钮  
 self.btnQuit = Button(self, text="推出", command=root.destroy)  
 self.btnQuit.pack()  
  
 def songhua(self):  
 messagebox.showinfo("送花", "送你一朵小红花")  
 print("后台打印，送一朵花")  
  
 def sayhi(self):  
 messagebox.showinfo("打招呼", "你好哈！")  
 print("后台打印，你好")  
  
  
root = Tk()  
root.geometry("400x200+200+100")  
root.title("一个经典的GUI程序类的测试")  
app = Application(master=root)  
  
root.mainloop()

■Label组件

from tkinter import \*  
  
  
class Application(Frame):  
 *"""一个经典的GUI程序的类的写法"""* def \_\_init\_\_(self, master=None):  
 super().\_\_init\_\_(master) # super调用父的初始化构造方法  
 self.master = master  
 self.pack()  
  
 self.createWidget()  
  
 def createWidget(self):  
 *"""在这里创建所有组件"""* self.label01 = Label(self, text="测试Label组件1", width=16, height=2, bg="black", fg="white")  
 self.label01.pack()  
  
 self.label02 = Label(self, text="测试Label组件2", width=16, height=2, bg="blue", fg="yellow", font=("微软雅黑", 30))  
 self.label02.pack()  
  
 global photo1 # 定义一个全局变量。如果是局部变量，本方法执行完毕后，图像对象销毁，窗口显示不出图片  
 photo1 = PhotoImage(file="imgs/icon1.gif")  
 self.label03 = Label(self, image=photo1)  
 self.label03.pack()  
  
 txt1 = "多行文本第一行\n文本第二行\n第三行"  
 self.label04 = Label(self, text=txt1, borderwidth=1, relief="solid", justify="right")  
 self.label04.pack()  
  
root = Tk()  
root.geometry("400x400+200+100")  
root.title("测试")  
app = Application(master=root)  
  
root.mainloop()

■组件的options



■单行输入框组件entry

from tkinter import \*  
from tkinter import messagebox  
  
  
class Application(Frame):  
 *"""一个经典的GUI程序的类的写法"""* def \_\_init\_\_(self, master=None):  
 super().\_\_init\_\_(master) # super调用父的初始化构造方法  
 self.master = master  
 self.pack()  
  
 self.createWidget()  
  
 def createWidget(self):  
 *"""在这里创建所有组件"""* self.label01 = Label(self, text="用户名")  
 self.label01.pack()  
  
 # StringVar变量绑定到指定的组件  
 # StringVar变量的值发生变化，组件内容也变化  
 # 组件内容发生变化，StringVar变量的值也跟着变化  
 v1 = StringVar()  
 self.entry01 = Entry(self, textvariable=v1)  
 self.entry01.pack()  
 v1.set("admin")  
 print(v1.get())  
 print(self.entry01.get())  
  
 v2 = StringVar()  
 self.entry02 = Entry(self, textvariable=v2, show="\*")  
 self.entry02.pack()  
 v2.set("123")  
 print(v2.get())  
 print(self.entry02.get())  
  
 self.btn01 = Button(self, text="登录", command=self.login)  
 self.btn01.pack()  
  
 def login(self):  
 # 拿到输入框的输入内容  
 print("用户名：" + self.entry01.get())  
 print("密码：" + self.entry02.get())  
 messagebox.showinfo("登录成功", "欢迎您" + self.entry01.get())  
  
root = Tk()  
root.geometry("400x400+200+100")  
root.title("测试")  
app = Application(master=root)  
  
root.mainloop()

■多行文本框 Text

from tkinter import \*  
from tkinter import messagebox  
  
  
class Application(Frame):  
 *"""一个经典的GUI程序的类的写法"""* def \_\_init\_\_(self, master=None):  
 super().\_\_init\_\_(master) # super调用父的初始化构造方法  
 self.master = master  
 self.pack()  
  
 self.createWidget()  
  
 def createWidget(self):  
 *"""在这里创建所有组件"""* self.w1 = Text(root, width=40, height=12, bg="gray")  
 self.w1.pack()  
  
 # 在第1行第0列位置插入如下字符串  
 self.w1.insert(1.0, "01234567\nabcdefg")  
 # 在第2行第3列位置插入如下字符串  
 self.w1.insert(2.3, "锄禾日当午。")  
  
 Button(self, text="重复插入文本", command=self.insertText).pack(side="left")  
 Button(self, text="获取文本", command=self.returnText).pack(side="left")  
 Button(self, text="添加图片", command=self.addImage).pack(side="left")  
 Button(self, text="添加组件", command=self.addWidget).pack(side="left")  
  
 def insertText(self):  
 # INSERT索引表示在光标处插入  
 self.w1.insert(INSERT, 'ABCabc')  
 # END索引表示在最后插入  
 self.w1.insert(END, 'DEFdef')  
  
 def returnText(self):  
 print(self.w1.get(1.2, 1.6))  
 print("所有文本内容：\n" + self.w1.get(1.0, END))  
  
 def addImage(self):  
 self.photo1 = PhotoImage(file="imgs/icon1.gif")  
 self.w1.image\_create(END, image=self.photo1)  
  
 def addWidget(self):  
 b1 = Button(self.w1, text="新按钮")  
 self.w1.window\_create(INSERT, window=b1)  
  
  
root = Tk()  
root.geometry("400x400+200+100")  
root.title("测试")  
app = Application(master=root)  
  
root.mainloop()

■单选框Radiobutton

from tkinter import \*  
from tkinter import messagebox  
  
  
class Application(Frame):  
 *"""一个经典的GUI程序的类的写法"""* def \_\_init\_\_(self, master=None):  
 super().\_\_init\_\_(master) # super调用父的初始化构造方法  
 self.master = master  
 self.pack()  
  
 self.createWidget()  
  
 def createWidget(self):  
 *"""在这里创建所有组件"""* self.v = StringVar()  
 self.v.set("F")  
  
 self.r1 = Radiobutton(root, text="男性", value="M", variable=self.v)  
 self.r2 = Radiobutton(root, text="女性", value="F", variable=self.v)  
  
 self.r1.pack(side="left"); self.r2.pack(side="left")  
  
 Button(root, text="确定", command=self.confirm).pack(side="left")  
  
 def confirm(self):  
 messagebox.showinfo("测试", "选择的性别" + self.v.get())  
  
root = Tk()  
root.geometry("400x400+200+100")  
root.title("测试")  
app = Application(master=root)  
  
root.mainloop()

■多选框Checkbutton

from tkinter import \*  
from tkinter import messagebox  
  
  
class Application(Frame):  
 *"""一个经典的GUI程序的类的写法"""* def \_\_init\_\_(self, master=None):  
 super().\_\_init\_\_(master) # super调用父的初始化构造方法  
 self.master = master  
 self.pack()  
  
 self.createWidget()  
  
 def createWidget(self):  
 *"""在这里创建所有组件"""* self.condHobby = IntVar()  
 self.videoHobby = IntVar()  
  
 print(self.condHobby.get()) # 默认值是0  
 self.c1 = Checkbutton(root, text="敲代码", variable=self.condHobby, onvalue=1, offvalue=0)  
 self.c2 = Checkbutton(root, text="看视频", variable=self.videoHobby, onvalue=1, offvalue=0)  
  
 self.c1.pack(side="left"); self.c2.pack(side="left")  
  
 Button(root, text="确定", command=self.confirm).pack(side="left")  
  
 def confirm(self):  
 messagebox.showinfo("提示", "敲代码的是" + str(self.condHobby.get()) + "\n看视频是" + str(self.videoHobby.get()))  
  
  
root = Tk()  
root.geometry("400x400+200+100")  
root.title("测试")  
app = Application(master=root)  
  
root.mainloop()