

TKinterDesigner 使用教程

开发者	Honghaier
版本号	1.2.0
更新日期	2020-03-09

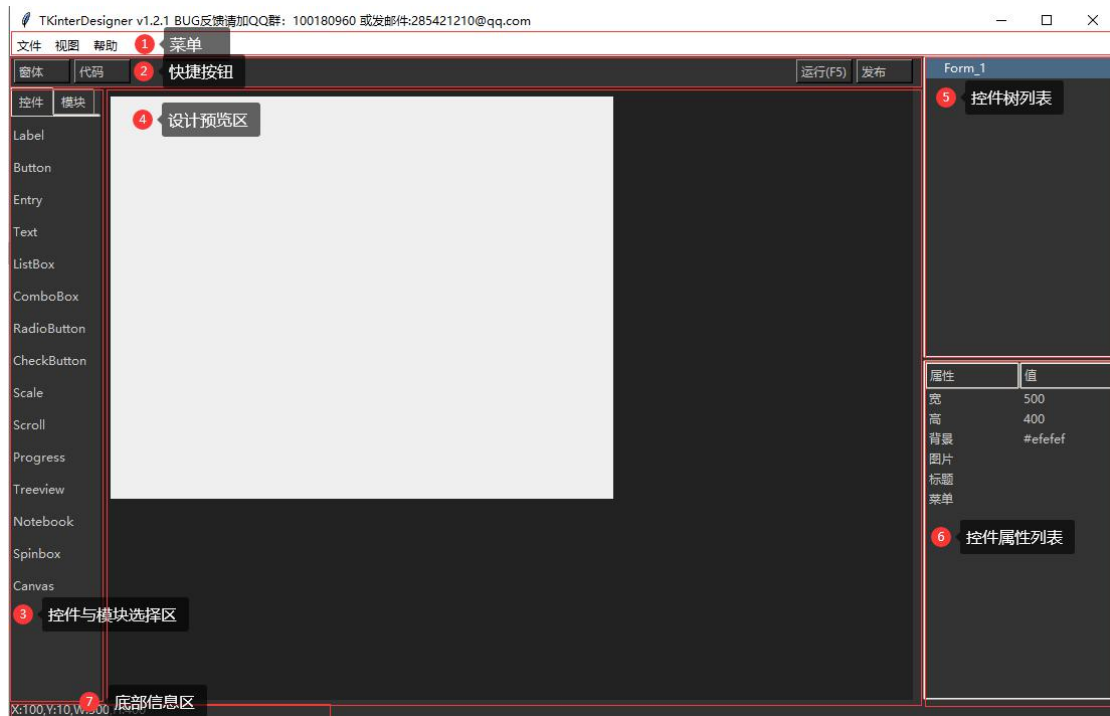
大家好，我是 Honghaier，我又回来了，这次我给大家带来一款自研的 Python 界面编辑器，希望能帮助大家在 Python 开发中提升开发效率。这款产品的名称就叫 TKinterDesigner，下面是一些简单的说明。如果大家还有什么不明白的地方，或好的建议，可以加 QQ 群：100180960，或给我发邮件：285421210@qq.com。我将持续改进，并在 GitHub 上发布新版本和插件库，地址为：<https://github.com/honghaier-game/TKinterDesigner.git>，本软件完全免费，如果好用您可以把他推荐给您的朋友和同事。

TKinterDesigner 是什么？

TKinterDesigner 是一款小巧的基于 Python TKinter 的界面编辑器，用于在进行小型快速 Python 功能开发原型时进行软件界面设计与开发。

TKinterDesigner 使用说明

一. 界面说明



编辑器由七个区域组成:

1. **主菜单**: 进行基本的文件, 视图, 帮助等菜单项操作。
2. **快捷按钮**: 快速切换窗体与代码视图, 并可以对界面进行运行状态查看和发布 EXE。
3. **控件与模块选择区**: 常用的界面控件和自定义模块, 用于拖动到界面设计区使用。
4. **设计预览区**: 用于摆放界面的主视区。
5. **控件树列表**: 当前界面中的所有控件列表, 用于进行快速访问和查看。
6. **控件属性区**: 显示和修改对应控件的属性。
7. **底部信息区**: 显示对应控件的位置, 大小等信息。

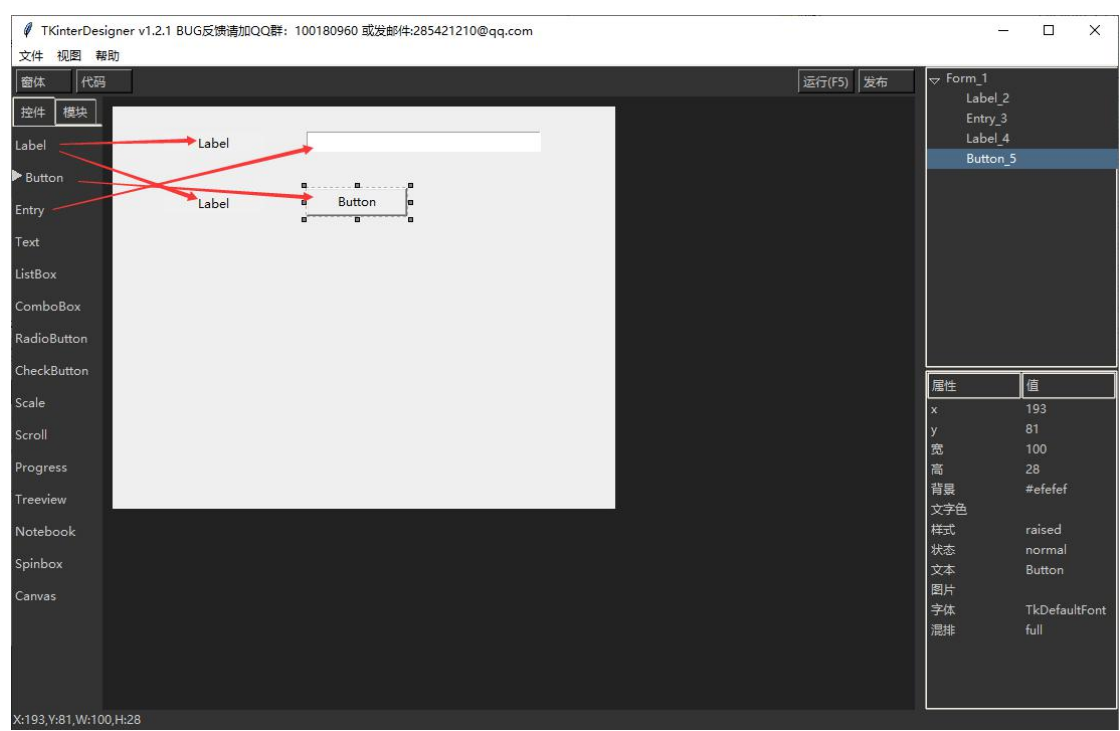
二. 功能说明

1. 界面设计

(1) 界面控件创建

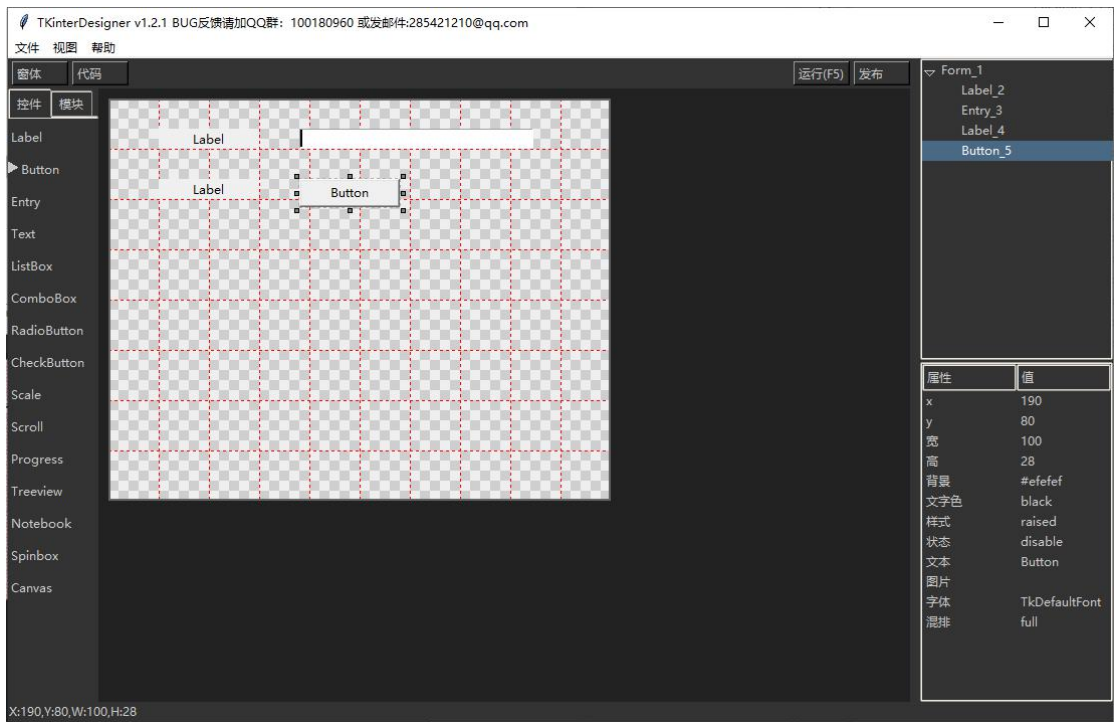
在控件选择区域中用鼠标左键选中对应的控件拖动到界面设计区的 Form 上, 即可完成

基本的控件创建，然后可以用鼠标拖动控件到相应的位置。



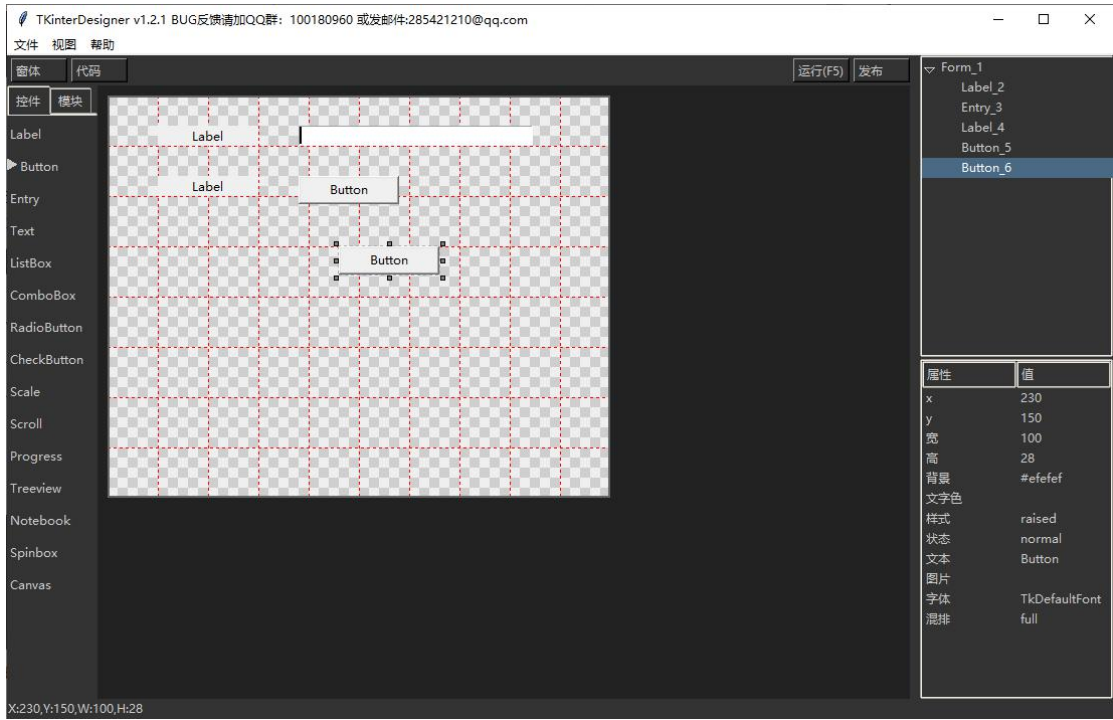
（2）界面控件摆放

如果你需要进行简单的界面摆放，首先你可以在主菜单的“视图”项下面找到“网格”，你也可以通过 **Ctrl+G** 来进行快速打开或关闭网格。这将方便你进行基本的大小观察。然后，你可以在主菜单的“视图”项下面找到“吸附”，或通过 **Ctrl+D** 来进行快速打开或关闭吸附。在开启了“网格”和“吸附”后，你可以通过鼠标直接对控件拖动来进行快速的摆放和对齐。

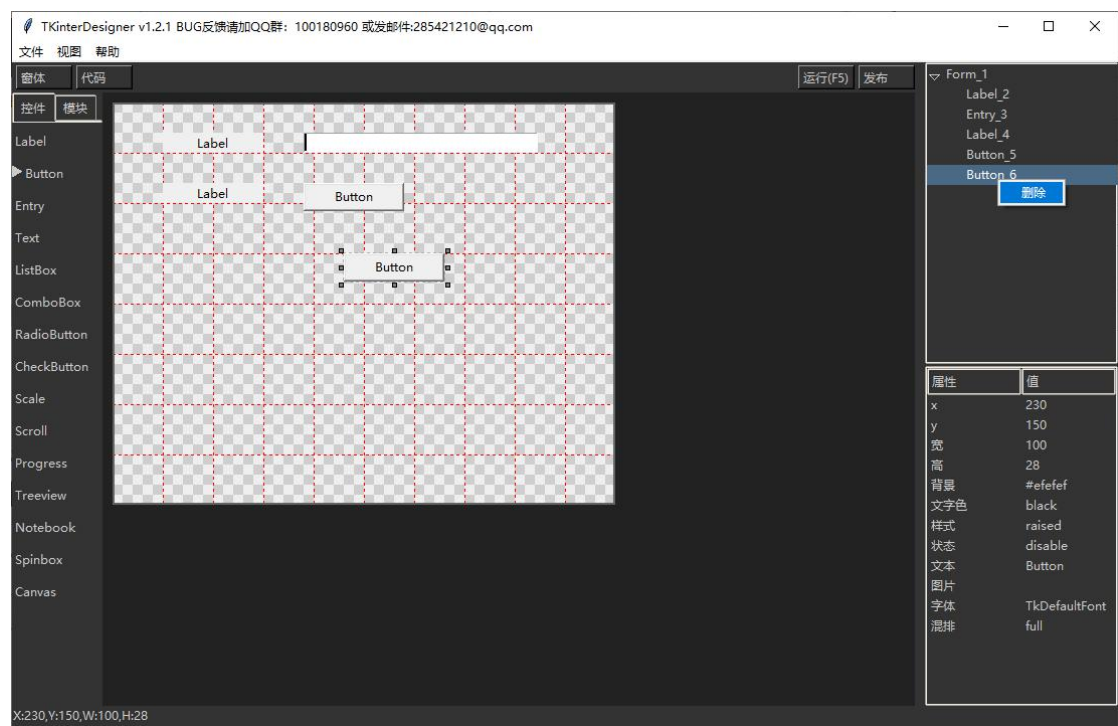


(3) 控件复制与删除

复制控件：选中一个控件后，按着 **Alt** 键并用鼠标拖动控件，即可复制出一个新的控件。



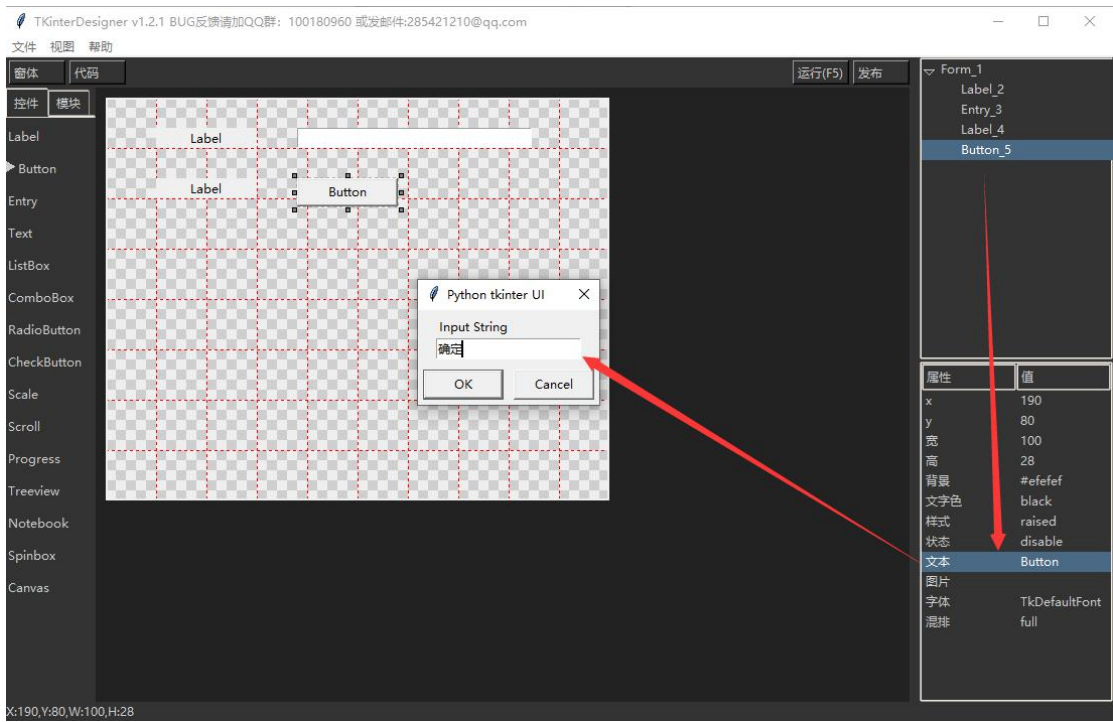
删除控件：可以通过快捷键 **delete** 或者在右上角的控件树中鼠标右键单击一个控件，在弹出菜单里来删除它。



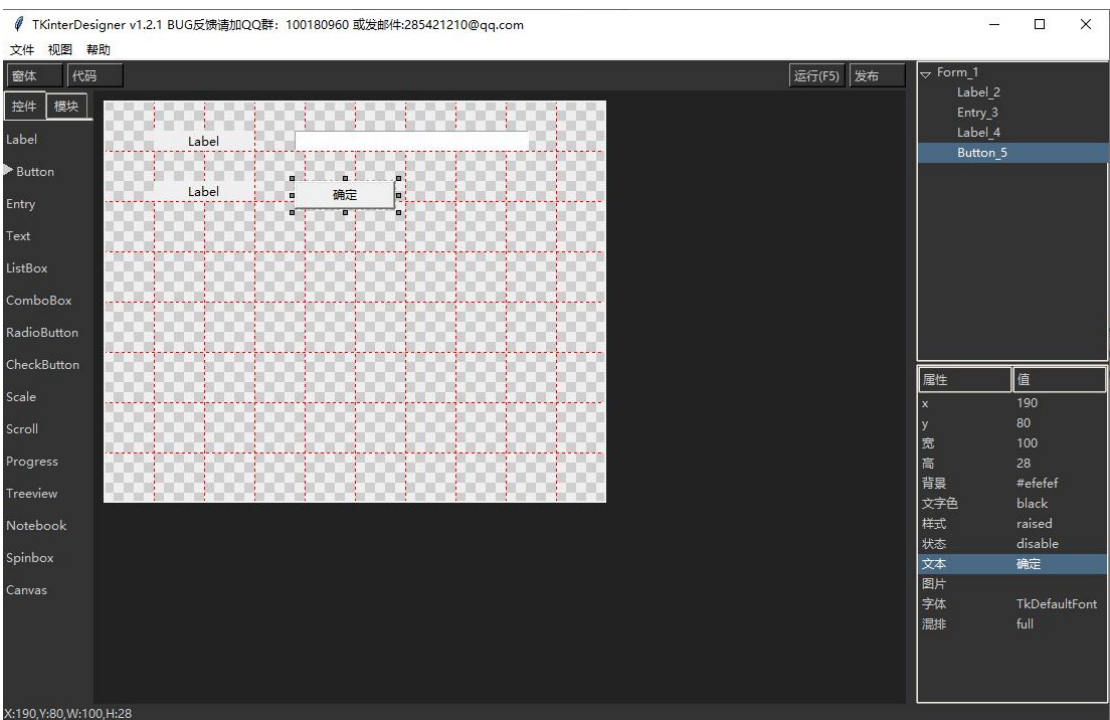
(4) 控件属性编辑

在选中相应控件后，右上角的控件列表树中，相应控件对应树项会处于高亮状态，此时我们可以在右下角的属性编辑区对相应的项进行修改。

我以控件按钮为例，以截图展示如何对文字进行修改，当我们双击属性“文本”项时，会弹出一个对话框，提示输入新的文字。



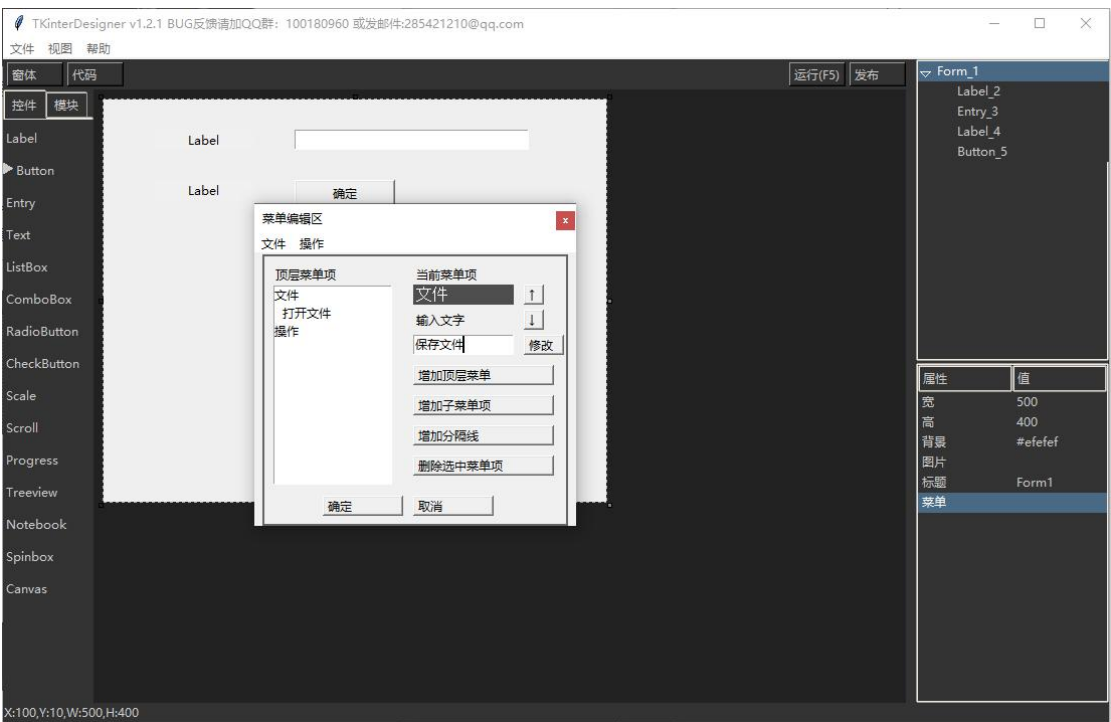
输入后，点击“确定”，即可看到相应的控件变更为当前输入的文字。



关于其它的属性，在此不进行一一介绍，大家可以自行尝试。

(5) 菜单编辑

如果需要给程序增加菜单，可以在右上角的控件树列表中选择“Form_1”，这时可以看到它的属性中有“菜单”项，双击后弹出菜单编辑对话框。

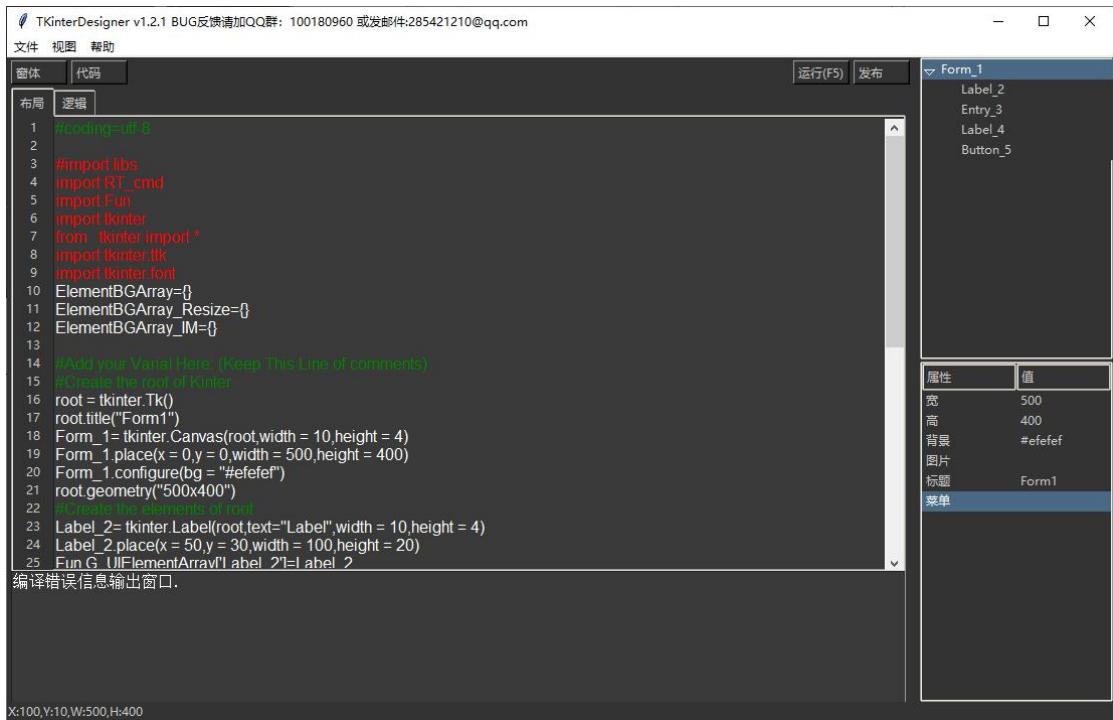


在菜单编辑对话框中，我们首先在输入文字的编辑框里输入“文件”，然后点击“增加顶层菜单”，这时可以看到在左边“顶层菜单项”的列表框里出现“文件”项，并在对话框中增加了菜单和“文件”菜单项，它可以方便你对菜单进行预览，对，这仅仅是一个预览。我们在增加了顶层菜单后，在左边“顶层菜单项”列表中点击相应的菜单项，在右边可以通过“增加子菜单项”按钮为其添加新的子项。并通过上下箭头按钮修改同一级菜单项的顺序。

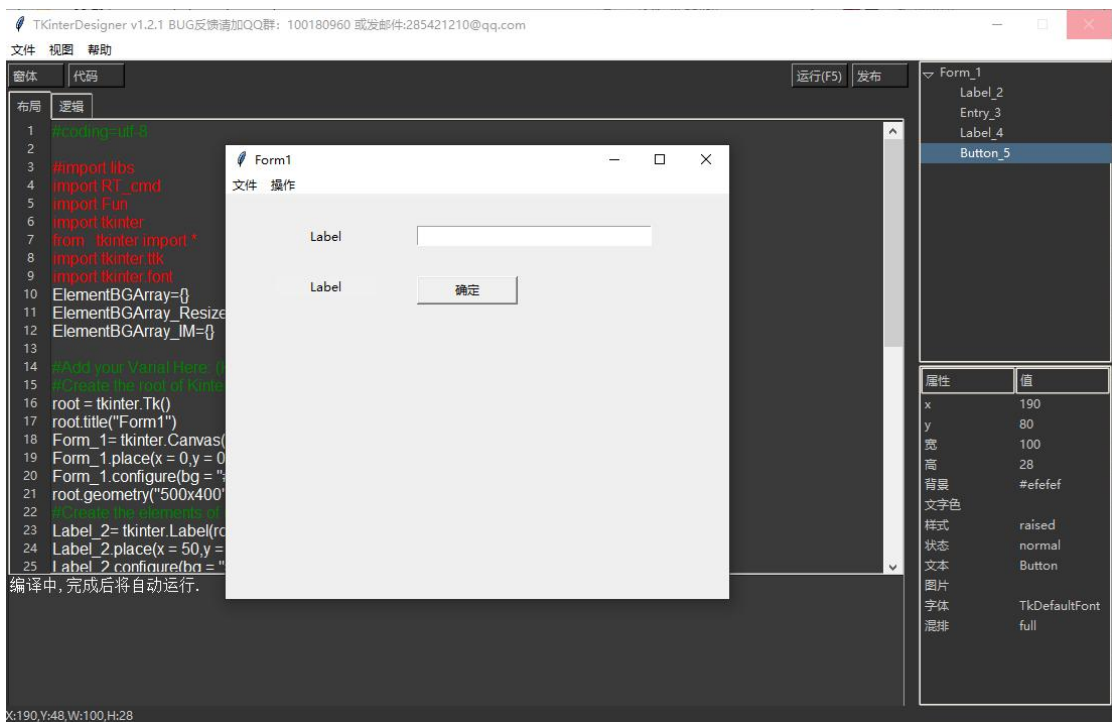
这一部分的处理略微复杂，但是我尽量使操作按钮简单明了，希望你能够明白。

2. 窗体与代码

如果我们希望查看代码，可以在快捷按钮区点击“代码”，这时我们可以看到界面的代码显示，它分为两个区，分别为“布局”和“逻辑”，其中“布局”代表的是界面布局文件的代码，“逻辑”代表的是界面逻辑文件的代码。



在对应的文本框中，我们可以手动进行代码的修改，并通过点击右上角“运行(F5)”来进行编译，如果成功，则自动执行，如果不成功，则会将错误信息输出到下面的窗口内，这样可以方便我们快速查看运行结果和查错。

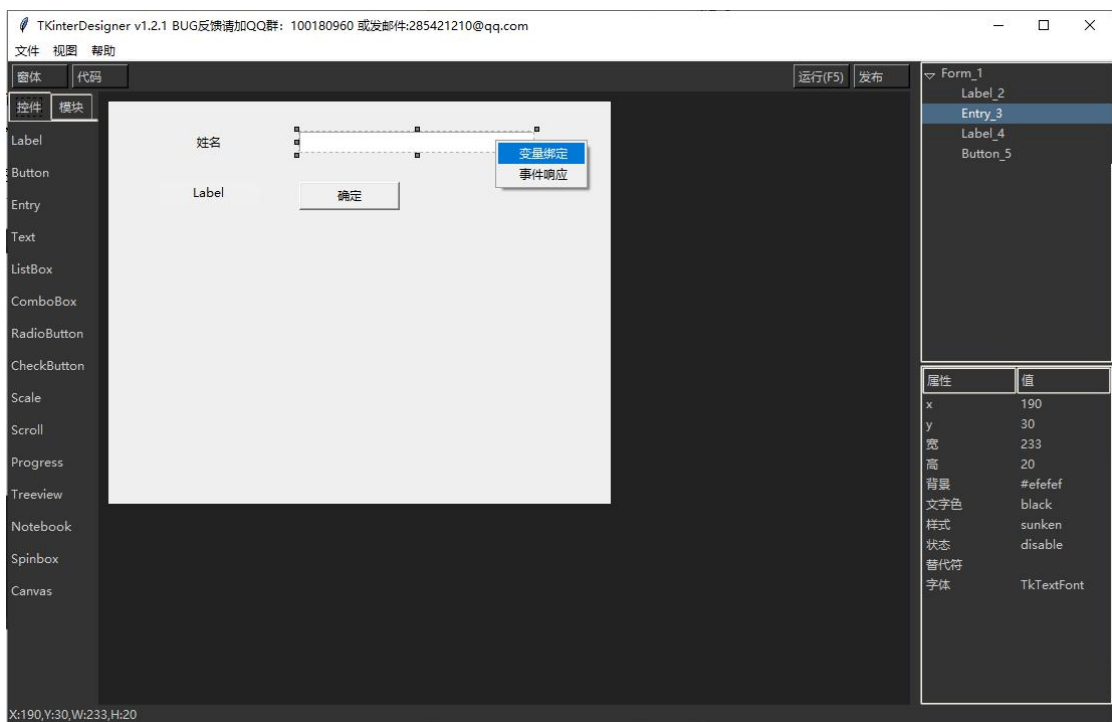


3. 控件的变量绑定

我们可以为界面中的控件，绑定一个变量，并对它进行访问和修改。

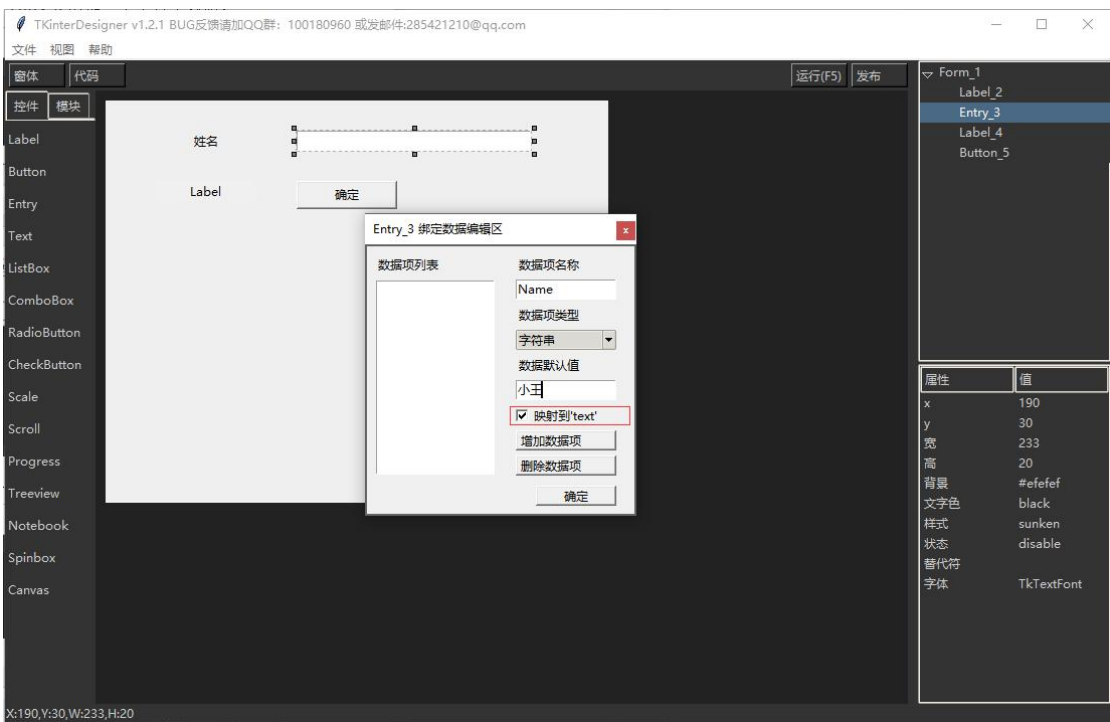
我们将 Label_2 的文本在属性框中修改为“姓名”，然后我们希望在点击 Button 的时候，修改输入框中的文字。

我们可以右键点击编辑框，在弹出菜单上选择“变量绑定”。

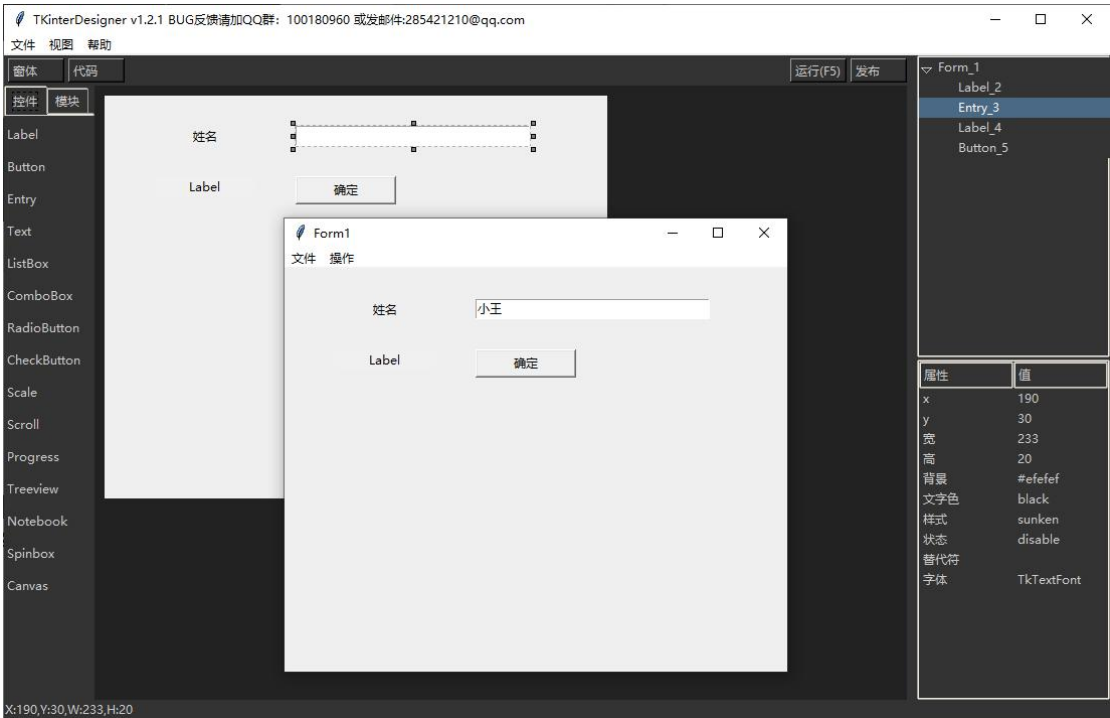


TKinterDesigner 使用教程

在弹出的对话框里为 Entry_3 增加一个绑定数据 Name,数据项类型选择“字符串”，数据默认值输入“小王”，选中“映射到'text'”，这个选项代表数据是否直接同步到 Entry_3 的 text 属性或 textvariable，点击“增加数据项”，即可为 Entry_3 增加一个字符串数据。



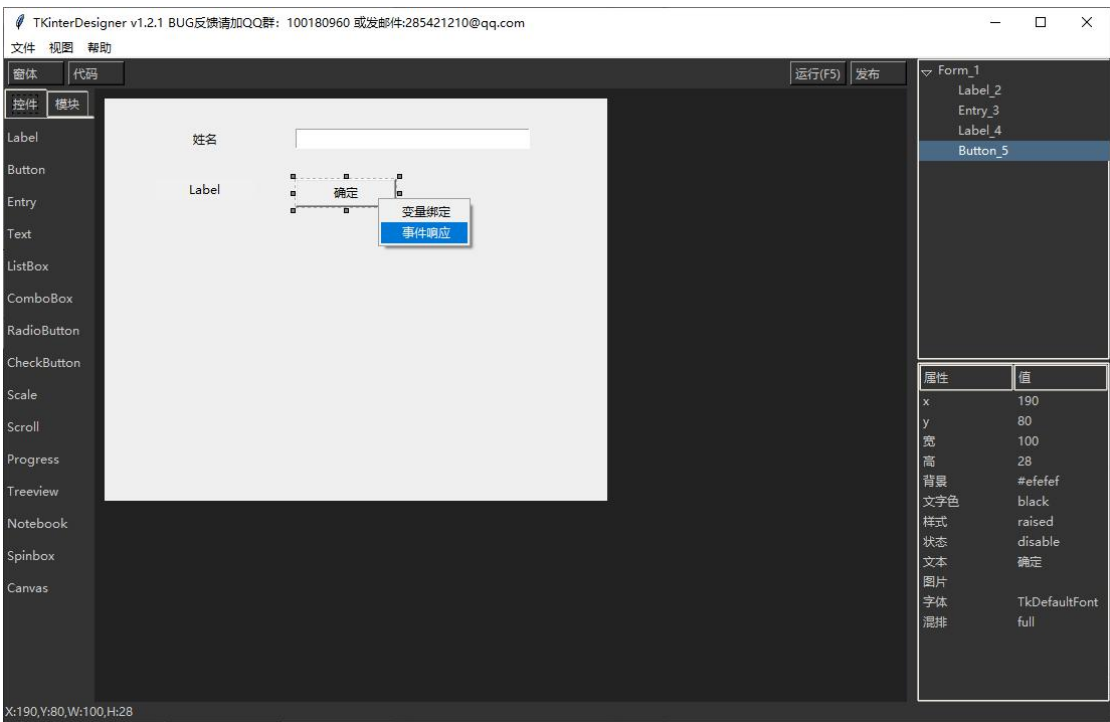
我们尝试按 F5 运行一下，可以看到窗口中的 Label 显示为“小王”。



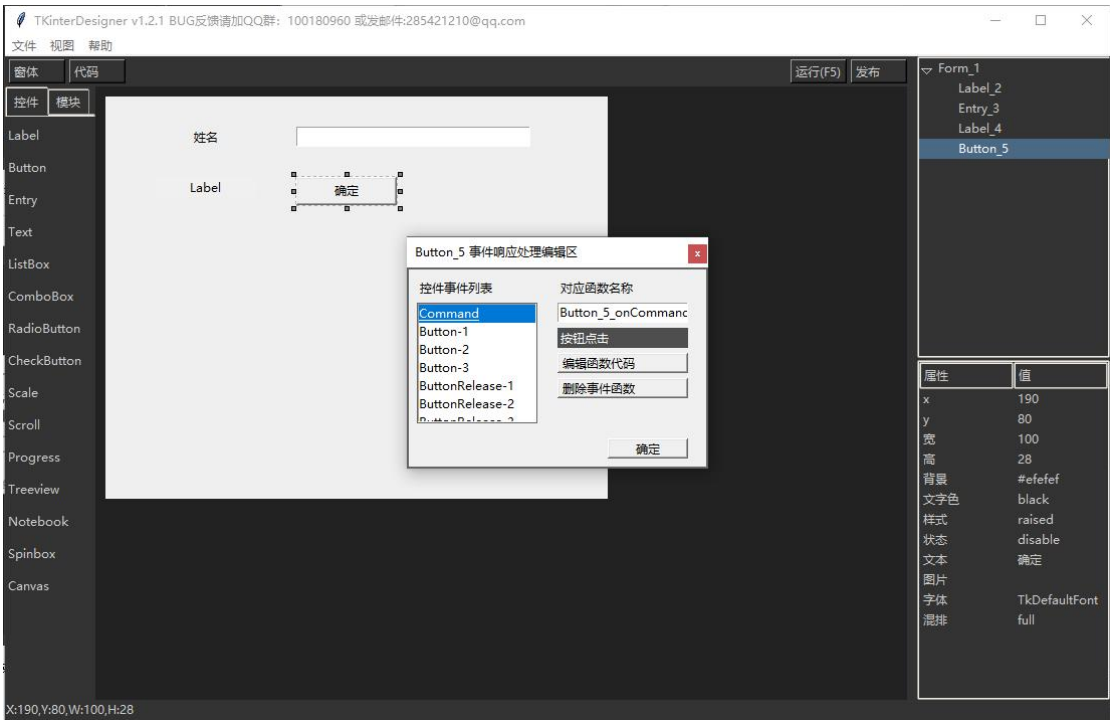
我们可以为一个控件增加多个绑定变量，并通过函数 `setUIData` 和 `getUIData` 来进行存取。

4. 控件的事件处理

我为每个控件增加了事件函数映射的向导，比如上面这个例子，我们可以在按钮上通过右键点击，在弹出的菜单点选择“事件响应”。



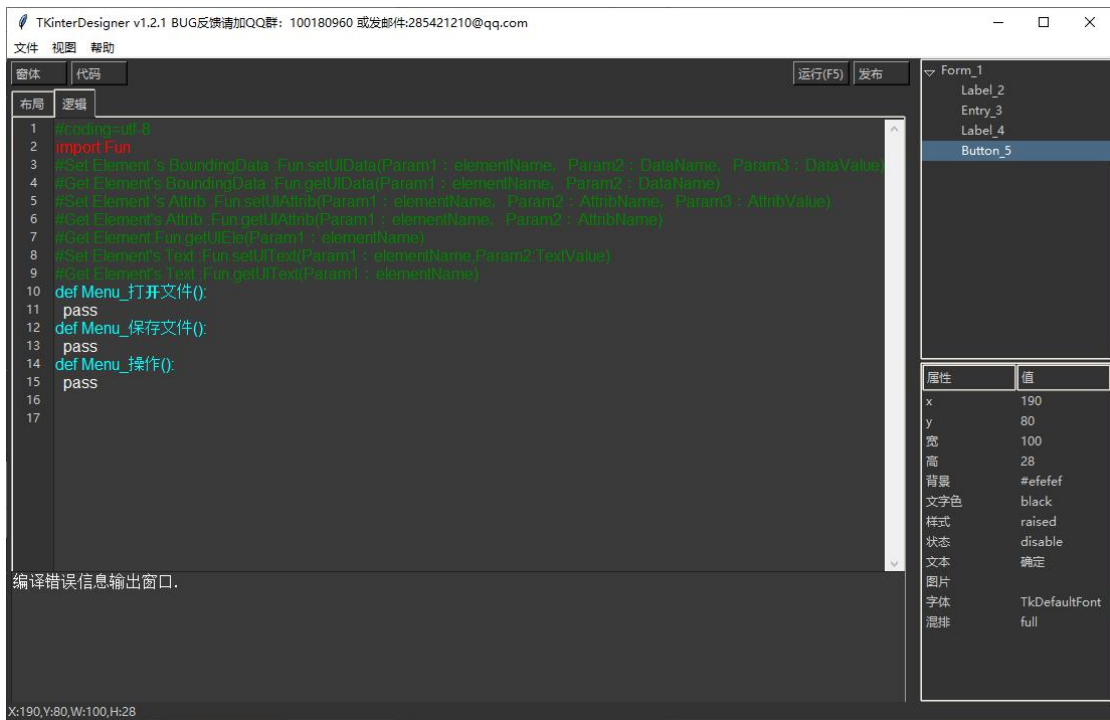
在弹出的事件响应处理编辑区里，我们可以看到相关控件所涉及到的事件列表。



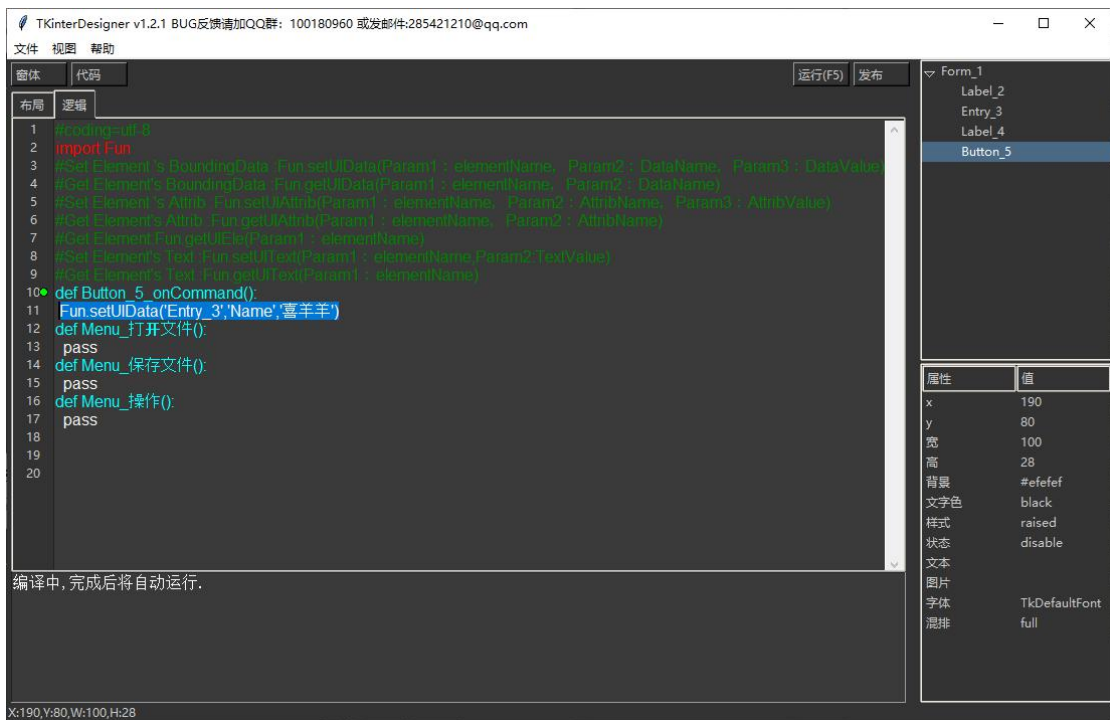
双击控件事件列表中的函数或点击“编辑函数代码”来对一个事件进行映射，这里我们

TKinterDesigner 使用教程

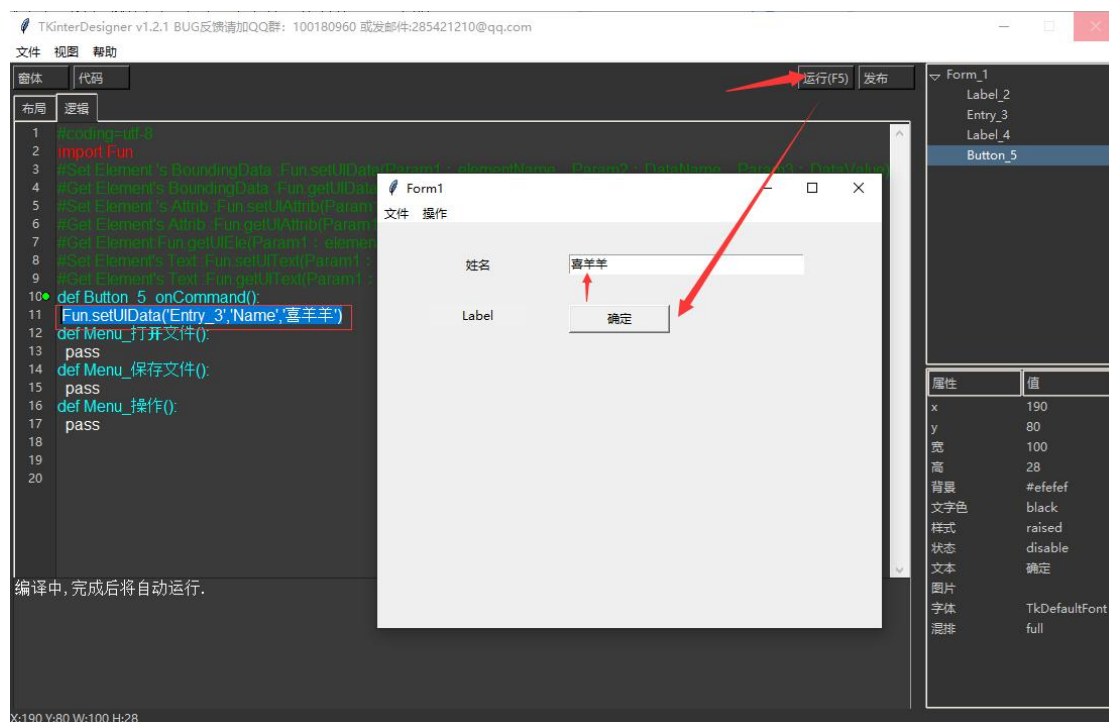
对按钮的点击事件，也就是 **Command** 进行编辑，可以进入代码编辑区：



你可以通过修改代码，比如改成这样：

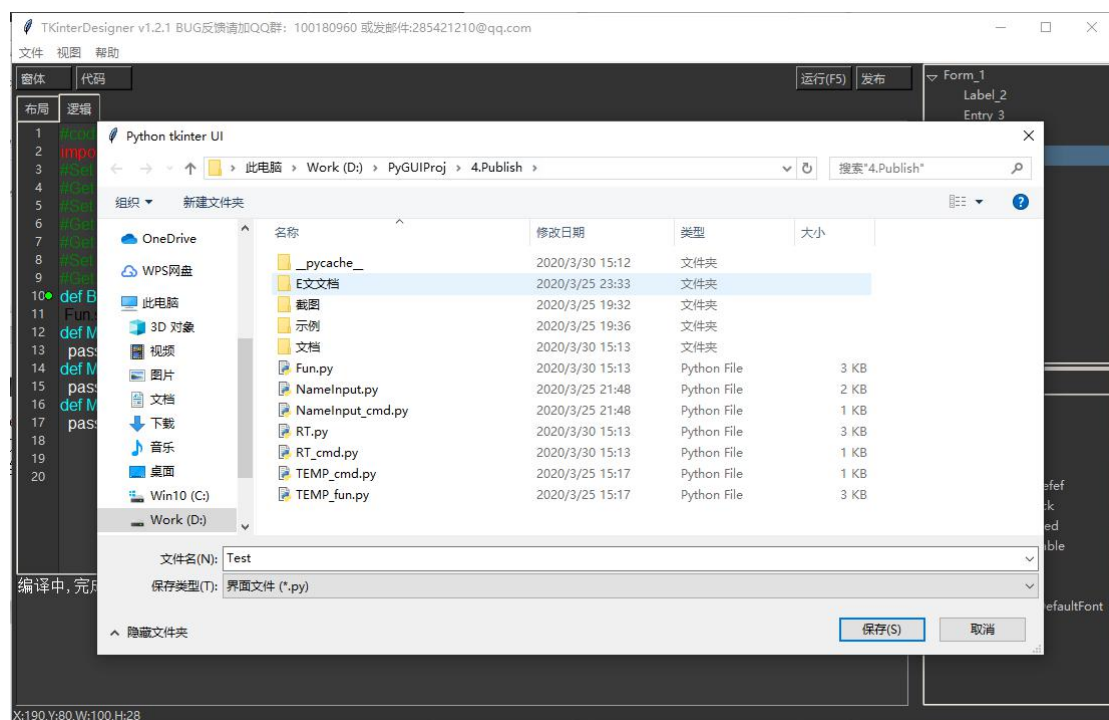


然后，你点击“运行”或按下 F5，可以在显面运行状态，点击按钮，这时你会发现，Label_2 的文本变成了‘喜羊羊’。



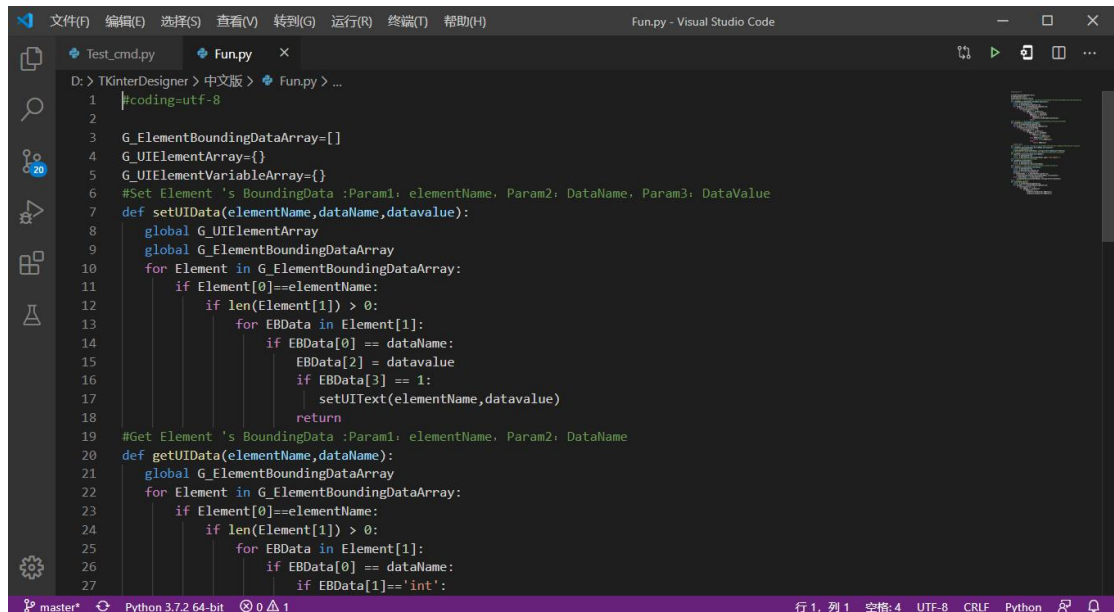
5. 文件的保存与打开

好的, 一切都弄好了, 我们只需要在主菜单的“文件”菜单项下找到“保存”, 或者 Ctrl+S, 来对文件进行保存。



在这里我们定义文件名为“Test”, 点击“保存”, 如果顺利会弹出“保存成功”提示。我们将在相应目录下可以看到生成的 Test.py, Test_cmd.py, Fun.py 三个文件, 其中 Test.py 是界面布局文件, Test_cmd.py 是对应的逻辑处理文件, 而 Fun.py 是一个库文件, 内置了我提供

的一些函数，方便对界面进行快速访问，你可以在 `Test_cmd.py` 中调用来方便获取控件或对控件的属性进行修改。



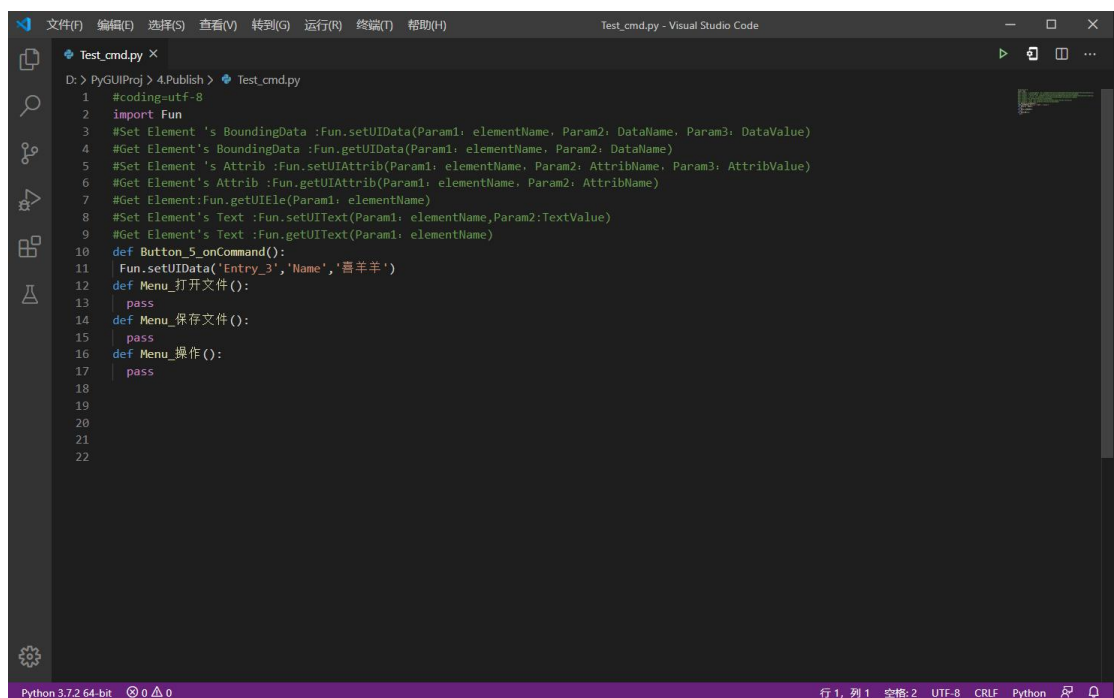
```

D:\> TKinterDesigner > 中文版 > Fun.py > ...
1  #coding=utf-8
2
3  G_ElementBoundingDataArray=[]
4  G_UIElementArray={}
5  G_UIElementVariableArray={}
6  #Set Element 's BoundingData :Param1: elementName, Param2: DataName, Param3: DataValue
7  def setUIData(elementName,dataName,datavalue):
8      global G_UIElementArray
9      global G_ElementBoundingDataArray
10     for Element in G_ElementBoundingDataArray:
11         if Element[0]==elementName:
12             if len(Element[1]) > 0:
13                 for EBDData in Element[1]:
14                     if EBDData[0] == dataName:
15                         EBDData[2] = datavalue
16                         if EBDData[3] == 1:
17                             setUIText(elementName,datavalue)
18                 return
19 #Get Element 's BoundingData :Param1: elementName, Param2: DataName
20 def getUIData(elementName,dataName):
21     global G_ElementBoundingDataArray
22     for Element in G_ElementBoundingDataArray:
23         if Element[0]==elementName:
24             if len(Element[1]) > 0:
25                 for EBDData in Element[1]:
26                     if EBDData[0] == dataName:
27                         if EBDData[1]=='int':

```

6. 逻辑处理

用文本编辑器打开 `Test_cmd.py`，我们将可以看到如下截图：



```

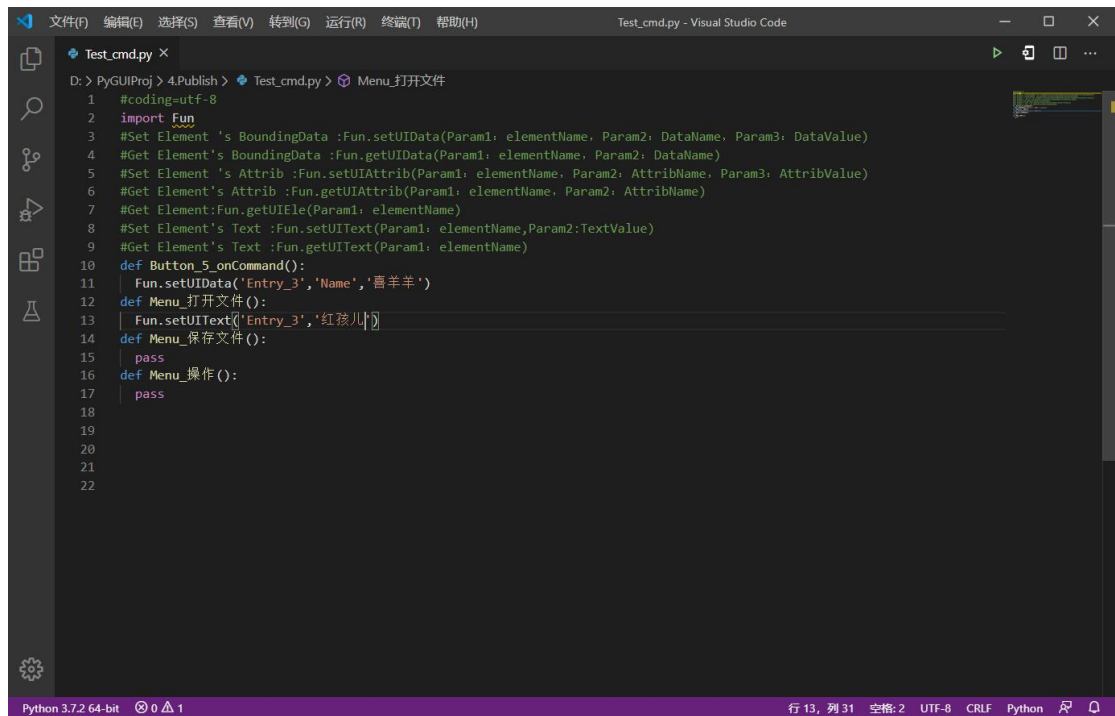
D:\> PyGUIProj > 4.Publish > Test_cmd.py
1  #coding=utf-8
2  import Fun
3  #Set Element 's BoundingData :Fun.setUIData(Param1: elementName, Param2: DataName, Param3: DataValue)
4  #Get Element 's BoundingData :Fun.getUIData(Param1: elementName, Param2: DataName)
5  #Set Element 's Attrib :Fun.setUIAttrib(Param1: elementName, Param2: AttribName, Param3: AttribValue)
6  #Get Element 's Attrib :Fun.getUIAttrib(Param1: elementName, Param2: AttribName)
7  #Get Element:Fun.getUIEle(Param1: elementName)
8  #Set Element's Text :Fun.setUIText(Param1: elementName,Param2:TextValue)
9  #Get Element's Text :Fun.getUIText(Param1: elementName)
10 def Button_5_onCommand():
11     Fun.setUIData('Entry_3','Name','喜羊羊')
12 def Menu_打开文件():
13     pass
14 def Menu_保存文件():
15     pass
16 def Menu_操作():
17     pass
18
19
20
21
22

```

有一些 `Fun` 中的函数注释，你可以直接通过“`Fun.函数名`”的形式访问它。比如我们改一下代码：

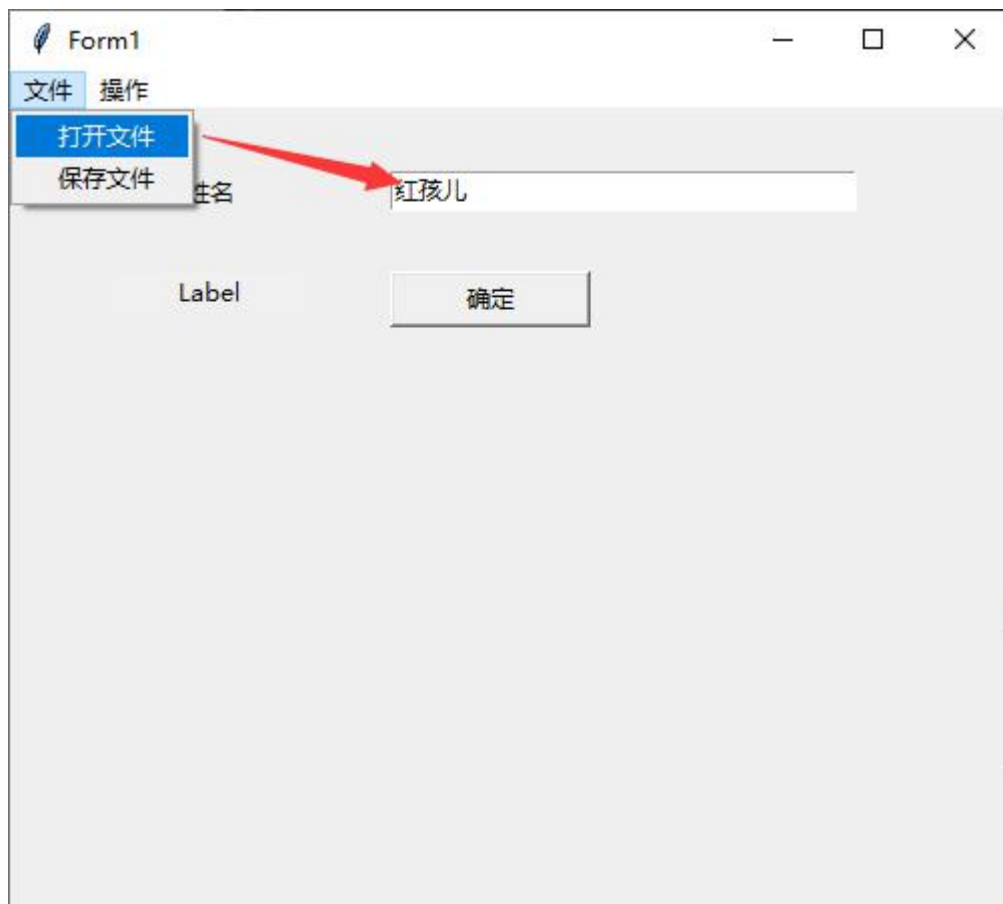
我们在预定义的菜单“打开文件”项里加入 `Fun.setUIText('Entry_3','123456')`

TKinterDesigner 使用教程



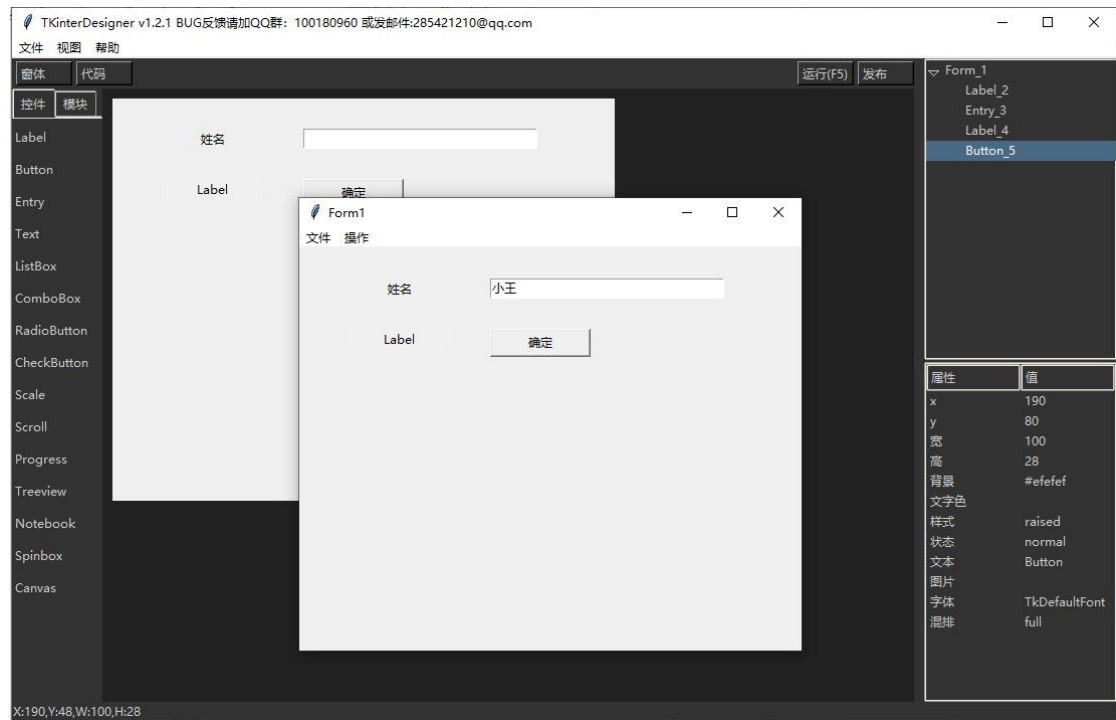
```
Test_cmd.py
D:\> PyGUIProj > 4.Publish > Test_cmd.py > Menu_打开文件
1  #coding=utf-8
2  import Fun
3  #Set Element's BoundingData :Fun.setUIData(Param1: elementName, Param2: DataName, Param3: DataValue)
4  #Get Element's BoundingData :Fun.getUIData(Param1: elementName, Param2: DataName)
5  #Set Element's Attrib :Fun.setUIAttrib(Param1: elementName, Param2: AttribName, Param3: AttribValue)
6  #Get Element's Attrib :Fun.getUIAttrib(Param1: elementName, Param2: AttribName)
7  #Get Element:Fun.getUIEle(Param1: elementName)
8  #Set Element's Text :Fun.setUIText(Param1: elementName,Param2:TextValue)
9  #Get Element's Text :Fun.getUIText(Param1: elementName)
10 def Button_5_onCommand():
11     Fun.setUIData('Entry_3','Name','喜羊羊')
12 def Menu_打开文件():
13     Fun.setUIText('Entry_3','红孩儿')
14 def Menu_保存文件():
15     pass
16 def Menu_操作():
17     pass
18
19
20
21
22
```

保存后，你可以执行 Test.py，点击菜单项“打开文件”，此时你会发现编辑框的值变为了 123456。

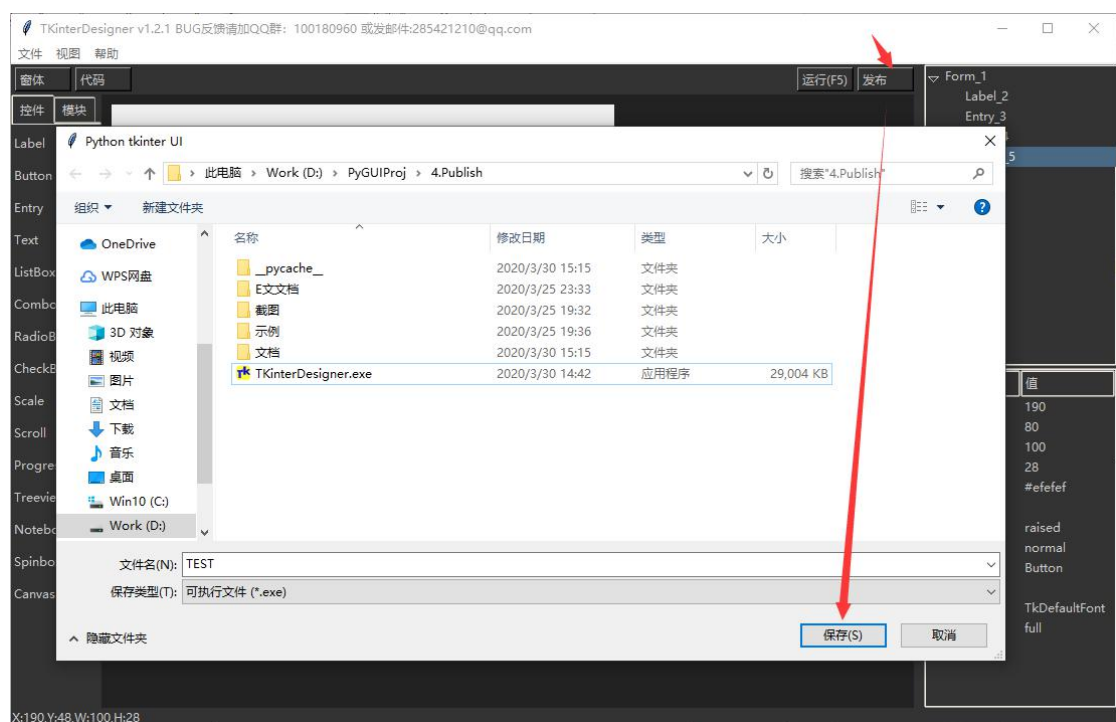


7. 运行与发布

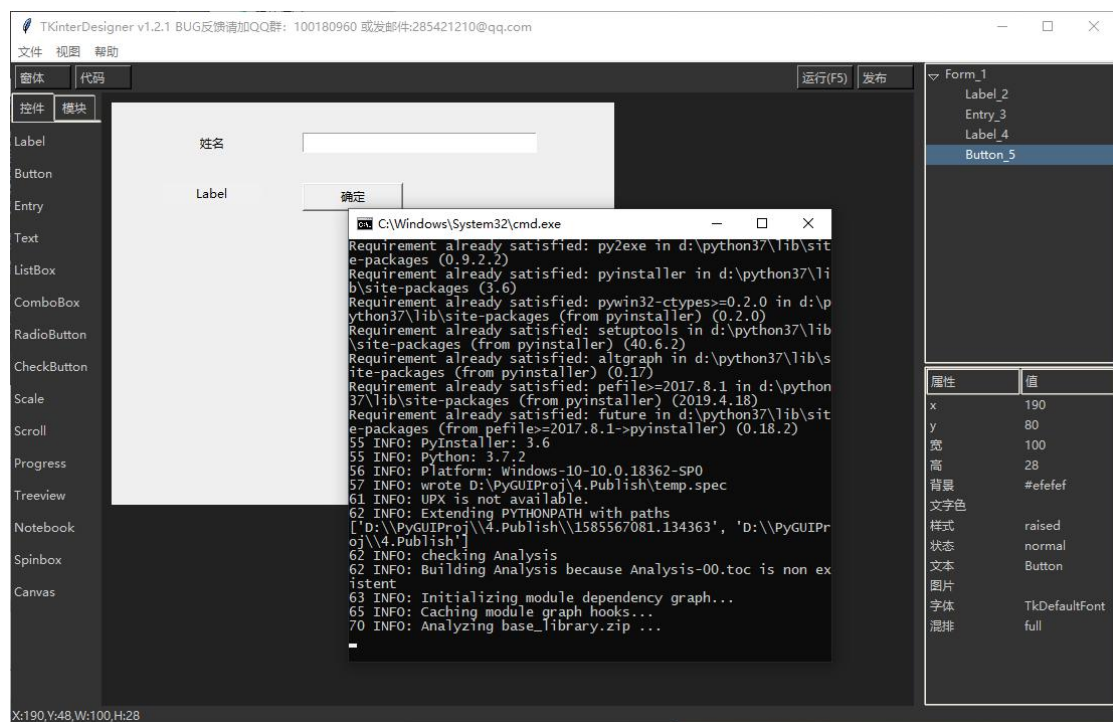
有时候，我们设计好了一个界面，我们想快速的看界面执行状态，这时我们只需要在快捷按钮区域点击“运行”按钮，即可快速的看到结果。



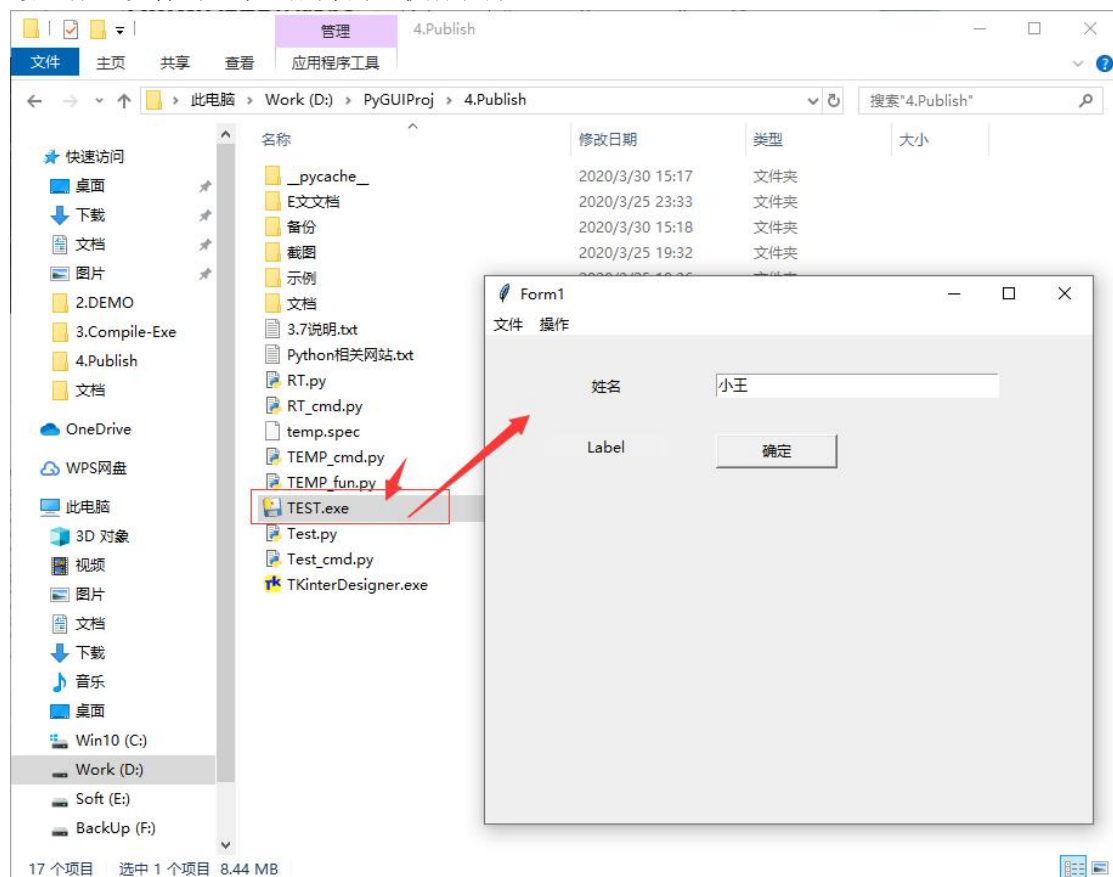
如果你已经完成它了，那你也可以点击“发布”来生成对应的执行文件。



点击“确定”后，会开始调用 pyinstaller 进行打包 EXE。



最终你可以看到一个生成好的可执行程序：



8. 自定义模块与模块库

自定义模块，就是提供给开发者一个加载使用自定义封装类的基本方法，我刚开始进行这方面的开发，相信随着越来越多的开发者加入，我们的插件库会进一步丰富起来，请及时从 [GitHub](#) 更新。

(1) 模块的编写

模块即就是一个 Python 的类，它的规则非常简单，你只需要为它设计一些属性和接口就可以了，唯一的技巧就是如果你确定需要在 TKinterDesigner 中进行基本的属性设置，那么你需要为这些属性编写 set 和 get 函数，我以一个简单的 Socket 类来做介绍：

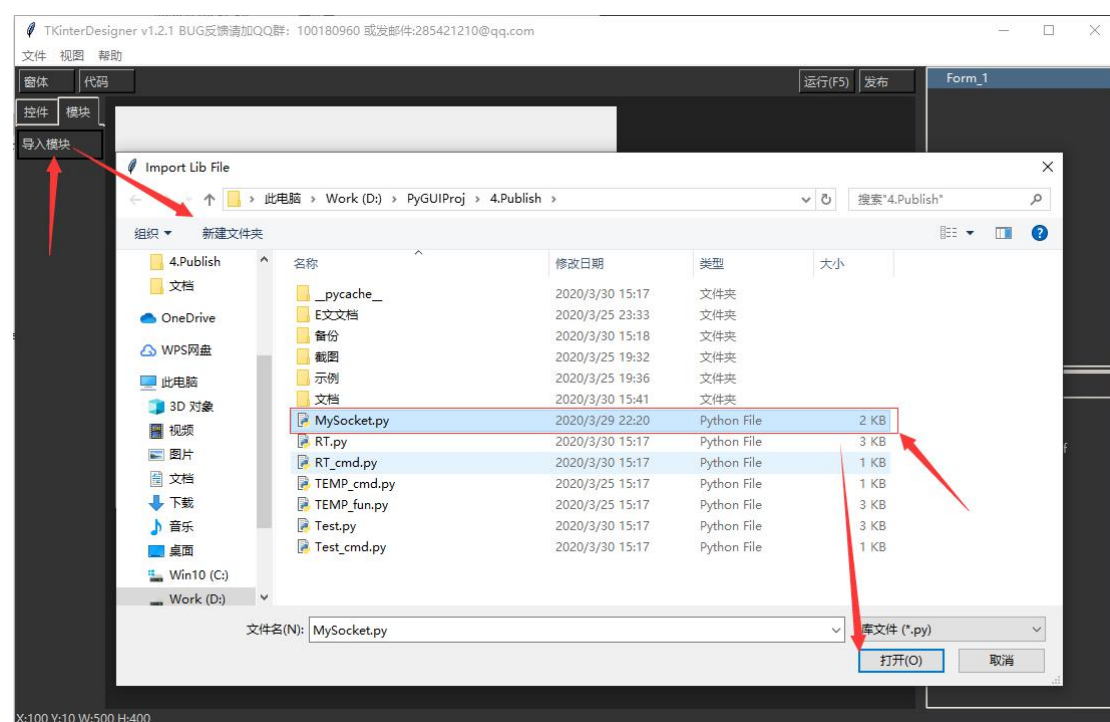
```
#coding=utf-8
import socket
import tkinter
import threading
import time

def handle_socket(ServerSocket, ListBox):
    conn, addr = ServerSocket.accept()
    text = "接受:" + str(addr)
    ListBox.insert(tkinter.END, text)
    while True:
        data = conn.recv(1024)
        text = "Receive Data:" + data.decode('utf-8')
        ListBox.insert(tkinter.END, text)
        time.sleep(3)
        send_data = 'got it'
        conn.send(send_data.encode('utf-8'))
    conn.close()

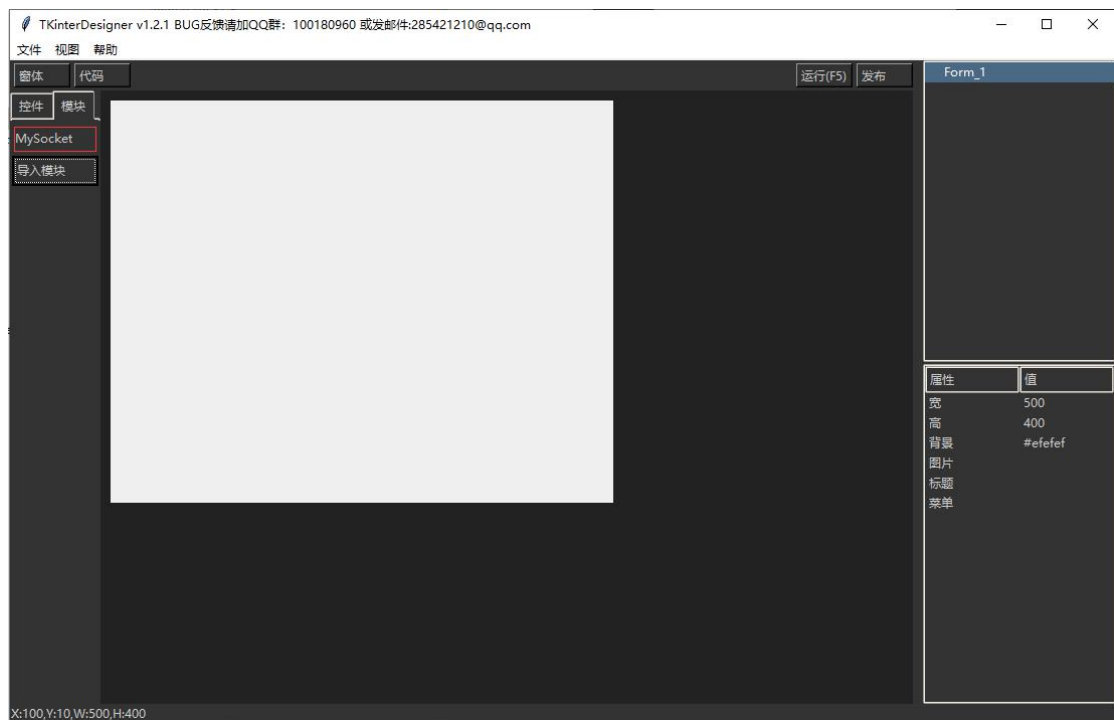
class MySocket:
    def __init__(self):
        self.s = None
        self.HOST = '127.0.0.1'
        self.PORT = 8888
    #设置 HOST
    def set_HOST(self, host):
        self.HOST = host
    #获取 HOST
    def get_HOST(self):
        return self.HOST
    #设置 PORT
    def set_PORT(self, port):
```

```
self.PORT = port
#获取 PORT
def get_PORT(self):
    return self.PORT
#创建服务器
def createServer(self,IPAddr,Port,ListBox):
    self.s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    self.s.bind((self.HOST,int(self.PORT)))
    self.s.listen(5)
    ListBox.insert(tkinter.END,"Socket 创建成功")
    client_thread = threading.Thread(target=handle_socket, args=[self.s, ListBox])
    client_thread.start()
#连接服务器
def connServer(self,IPAddr,Port):
    self.s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    self.s.connect((self.HOST,int(self.PORT)))
#发送信息
def sendMessage(self,text):
    msg = text.encode('utf-8')
    print("SendMsg:"+text)
    self.s.send(msg)
```

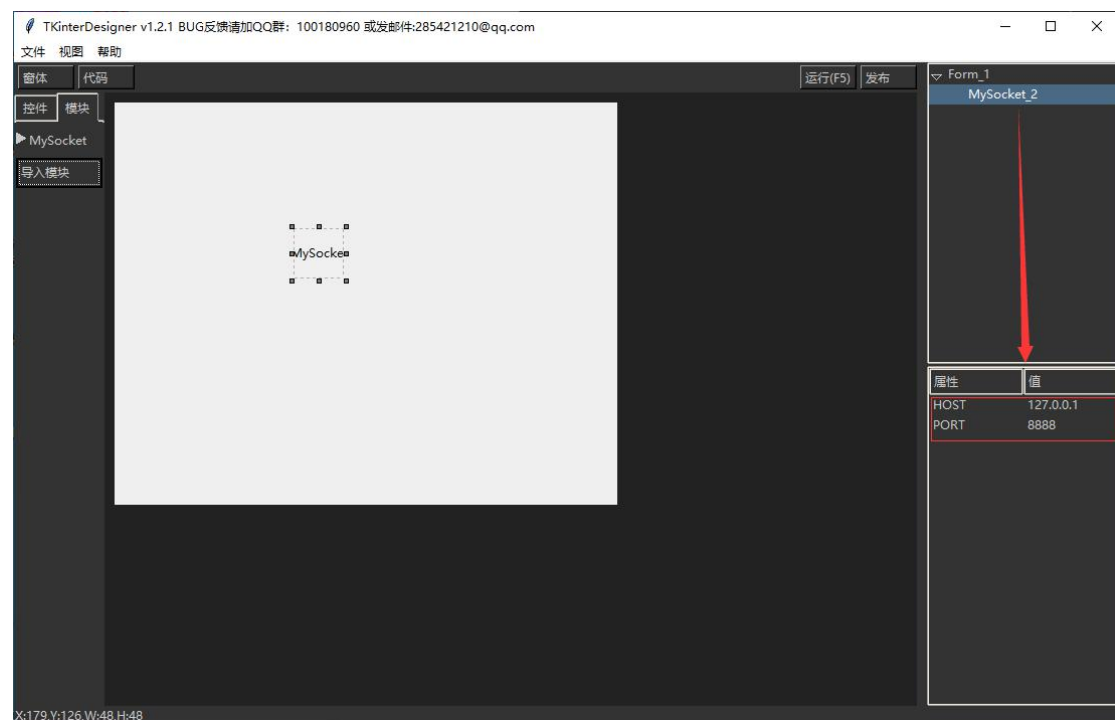
这是我随手写的一个测试代码，我们暂且就叫它“MySocket”吧。然后保存 MySocket.py 后，我们打开 TKinterDesigner。



在左边的控件选择区域，首先切换到“模块”，然后点击“导入模块”，在弹出的文件选择框中，找到我们刚刚编写的“`MySocket.py`”，打开文件后，我们会发现在模块选择区域多了一个模块“`MySocket`”：

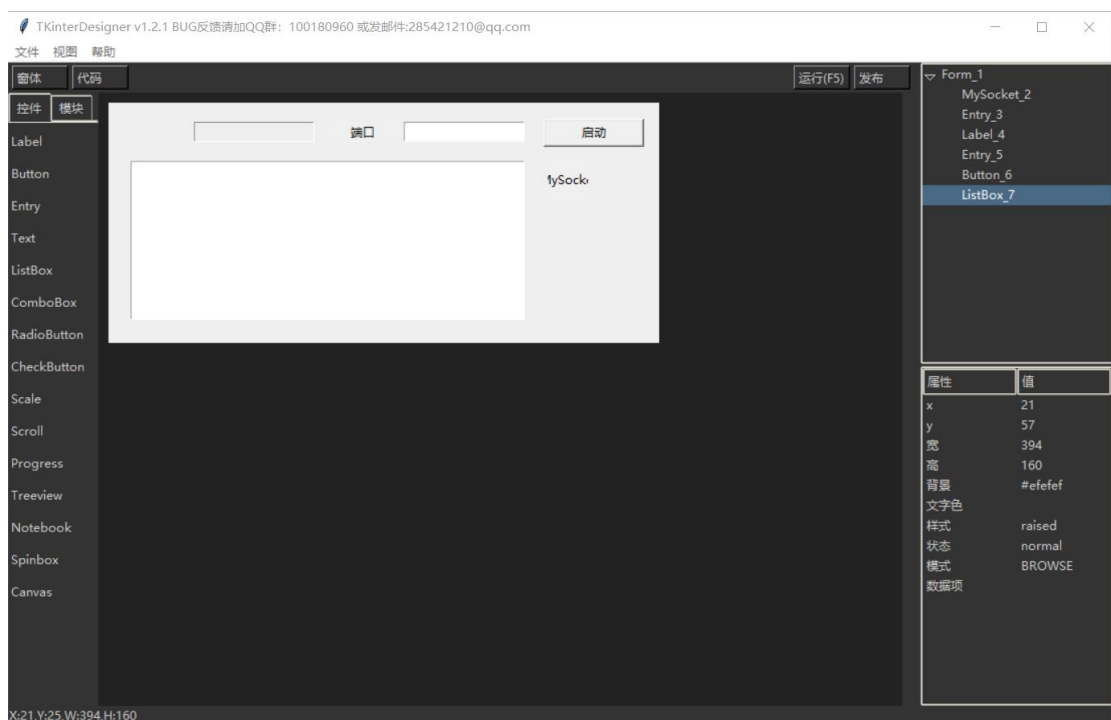


我们可以像拖放控件一样将它拖入到设计区的 `Form` 上，这时我们可以看到它被加入到控件元素树列表，并在属性列表处显示出我们要暴露的属性，供我们修改初始值。



如果我们需要在代码中访问它，可以通过“`Fun.getUIEle('MySocket_2')`”来取得这个实例化的对象。

比如：我们创建了这样一个界面：



这个界面有两个输入框，分别输入 IP 和端口，还有一个列表框用来显示消息，点击右边启动后，调用 `MySocket` 实例的创建服务器函数，我把具体的代码放在了 `examples` 的 `Server` 目录下。

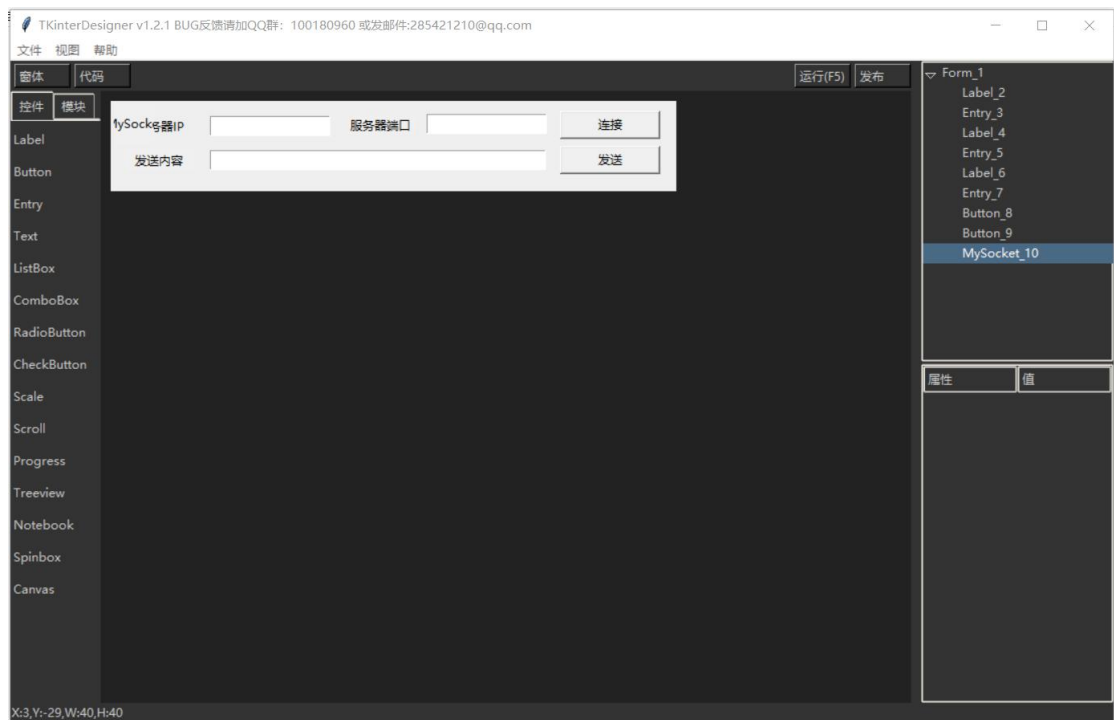
```

MySocket.py  Server_cmd.py X
D:\> PyGUIProj > 3.Compile-Exe > Server > Server_cmd.py > ...
1  #coding=utf-8
2  import Fun
3  #Set Element 's BoundingData :Fun.setUIData(Param1: elementName, Param2: DataName, Param3: DataValue)
4  #Get Element's BoundingData :Fun.getUIData(Param1: elementName, Param2: DataName)
5  #Set Element 's Attrib :Fun.setUIAttrib(Param1: elementName, Param2: AttribName, Param3: AttribValue)
6  #Get Element's Attrib :Fun.getUIAttrib(Param1: elementName, Param2: AttribName)
7  #Get Element:Fun.getUIEle(Param1: elementName)
8  #Set Element's Text :Fun.setUIText(Param1: elementName,Param2:TextValue)
9  #Get Element's Text :Fun.getUIText(Param1: elementName)
10 def Button_6_onCommand():
11     ListBox = Fun.getUIEle('ListBox_7')
12     MySocket = Fun.getUIEle('MySocket_2')
13     IPAddr = Fun.getUIData('Entry_3','IPAddr')
14     PORT = Fun.getUIData('Entry_5','Port')
15     MySocket.createServer(IPAddr,PORT,ListBox)
16
17
18

```

再做一个客户端的界面，用来向服务器发送消息：

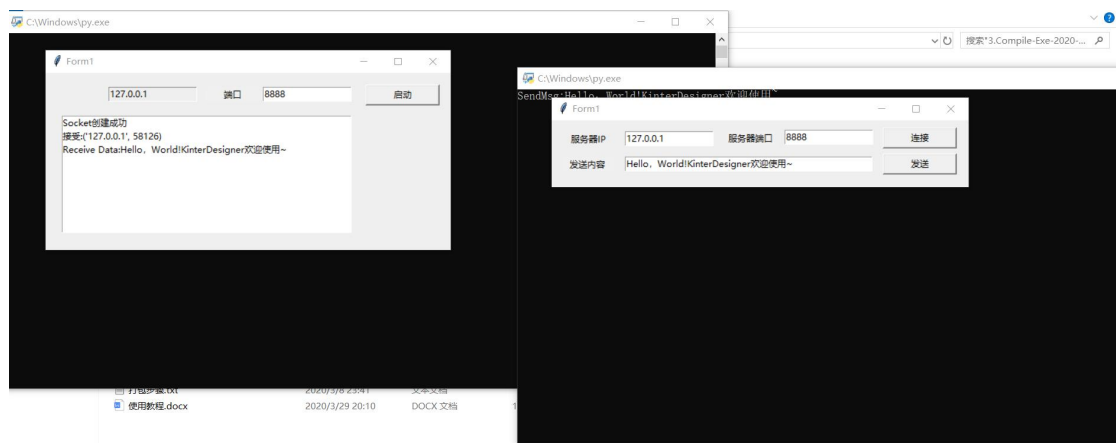
TKinterDesigner 使用教程



我们为相应的按钮设计 onCommand 消息，并实现调用 MySocket 实例的部分：

```
MySocket.py  Server_cmd.py  Client_cmd.py X
D: > PyGUIProj > 3.Compile-Exe > Server > Client_cmd.py > ...
1  #coding=utf-8
2  import Fun
3  #Set Element 's BoundingBox :Fun.setUIData(Param1: elementName, Param2: DataName, Param3: DataVa
4  #Get Element's BoundingBox :Fun.getUIData(Param1: elementName, Param2: DataName)
5  #Set Element 's Attrib :Fun.setUIAttrib(Param1: elementName, Param2: AttribName, Param3: AttribVa
6  #Get Element's Attrib :Fun.getUIAttrib(Param1: elementName, Param2: AttribName)
7  #Get Element:Fun.getUIEle(Param1: elementName)
8  #Set Element's Text :Fun.setUIText(Param1: elementName,Param2:TextValue)
9  #Get Element's Text :Fun.getUIText(Param1: elementName)
10 def Button_8_onCommand():
11     MySocket = Fun.getUIEle('MySocket_10')
12     IPAddr = Fun.getUIData('Entry_3','IPAddr')
13     PORT = Fun.getUIData('Entry_5','Port')
14     MySocket.connServer(IPAddr,PORT)
15 def Button_9_onCommand():
16     MySocket = Fun.getUIEle('MySocket_10')
17     MsgText = Fun.getUIText('Entry_7')
18     MySocket.sendMessage(MsgText)
19
20
21
```

最后它们可以运行如下：



如果你还有什么不明白的地方，你可以去更新 [Github](#) 上的代码。

最后总结一下:自定义模块的意义，是希望能不断的发掘所有人的潜力，为工具开发更多实用的类库，而我也将不仅仅只是通过界面把它导进去这么无聊，后续还有一些有趣的想法，大家可以等待下一版再来看。

三. 作者介绍

网名：火云红孩儿

作品：红孩儿工具箱，CocosVR。

教程：《Cocos 引擎深入分析系列》，《Shader 从入门到精通系列》

成就：原无限世界引擎总监，原 COCOS 引擎总监，CSDN 博客专家，Cocos 最有价值专家。

联系方式：QQ：285421210

四. 关于 TKinterDesigner

首先，因为它只是我业余时间断断续续学了两个月 Python 的一点小动机，它有很多问题，不过我会继续完善的，在此希望有兴趣的 Python 爱好者与我交流，因为我在写之前还没看完一本 Python 书，有太多的不入门。

其它，我很看好 Python 在快速化原型开发上的界面工具需求，希望 Python 越来越好。

最后，祝各位工作顺心，身体健康~

火云红孩儿

2020/03/31