

Ubuntu 下 VScode Pyqt5 GUI 编程教程

一. 配置 Pyqt5 和 Qt Designer 环境

1. 在命令行安装库

```
pip install PyQt5 -i https://pypi.tuna.tsinghua.edu.cn/simple
```

2. 在 VScode 安装“PYQT Integration”插件

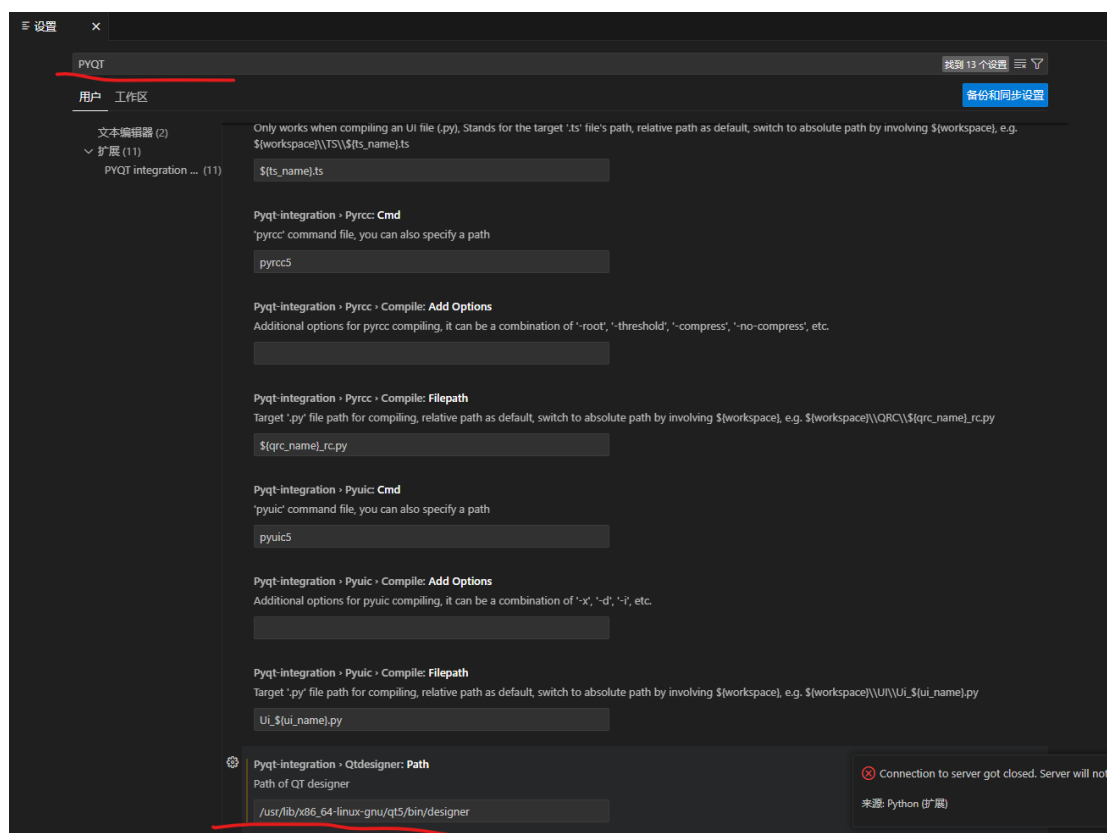


3. 配置 Qt Designer 的路径

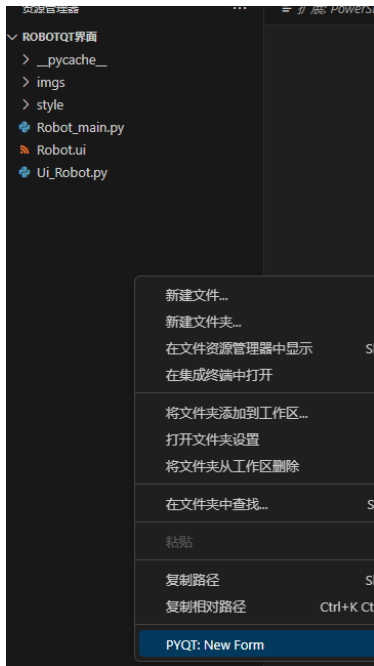
打开设置



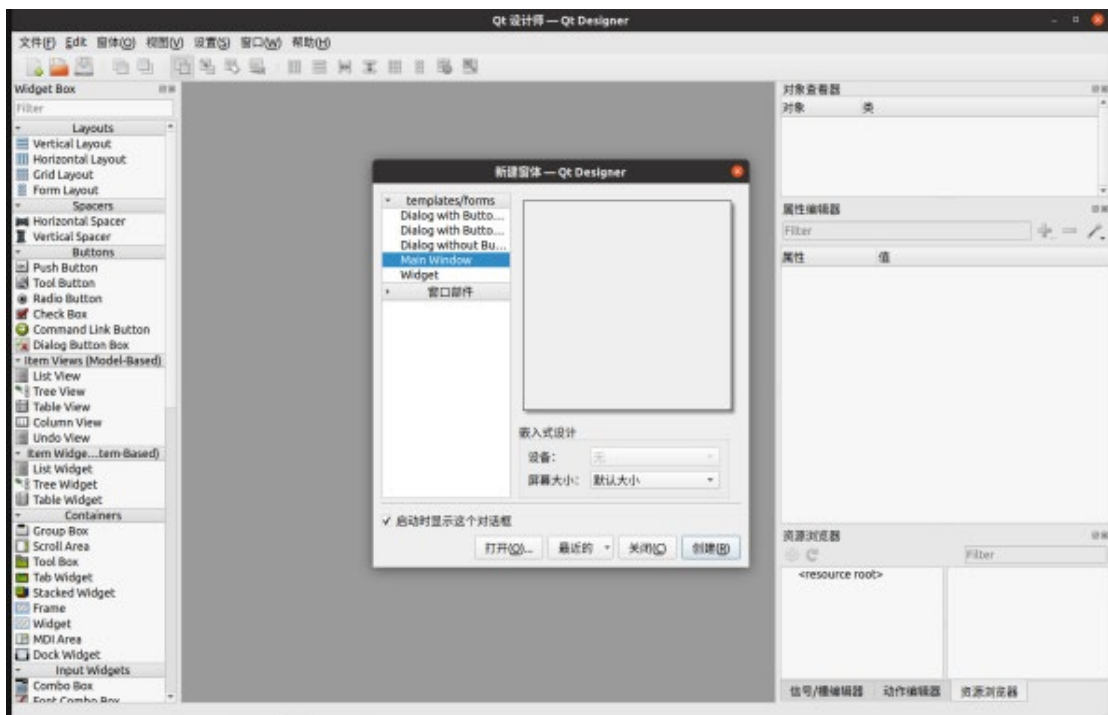
输入 pyqt，在最下一栏“Qt designer:path”里，输入路径
/usr/lib/x86_64-linux-gnu/qt5/bin/designer



4. 检查插件是否配置成功 在当前项目的资源管理栏里，右键，
会弹出如下面板。



点击 PYQT: New Form ,会弹出“Qt Designer”的设计界面，说明配置没有问题。



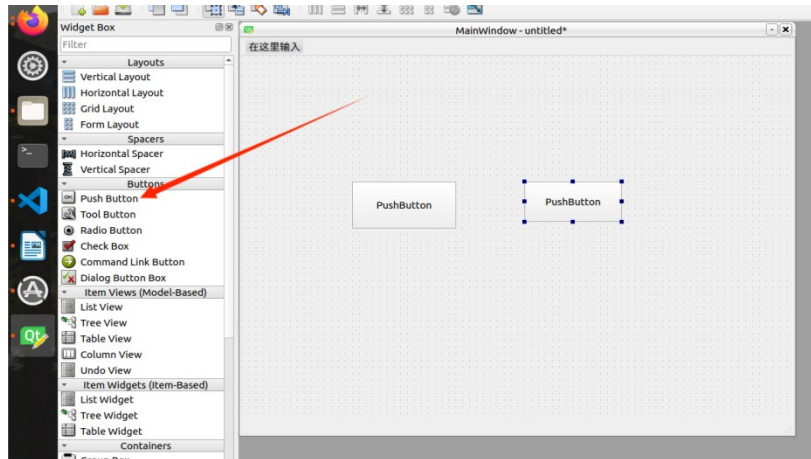
二. 编辑文件

先创建一个简单的界面并保存，再编辑文件使 QT 界面能正常运行，再开始编辑

UI 界面

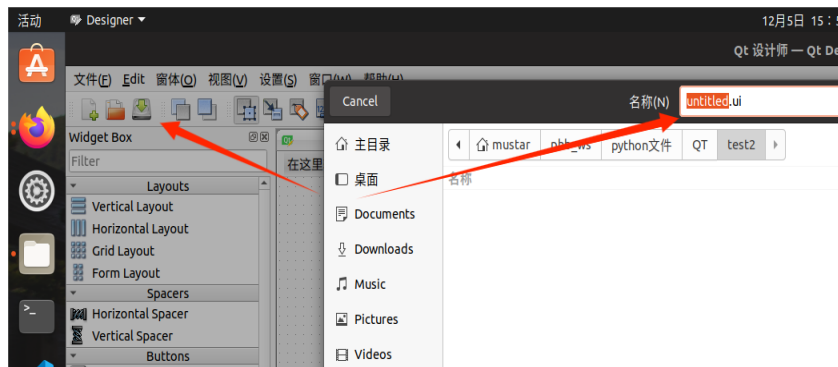
1. 测试界面

选中 Main Window 点击创建，将左边的 PushButton 拖入界面



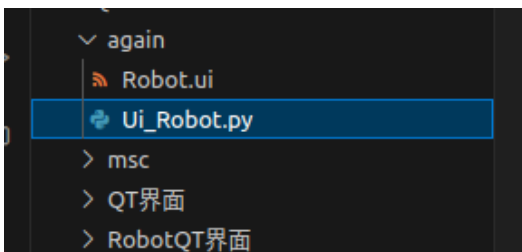
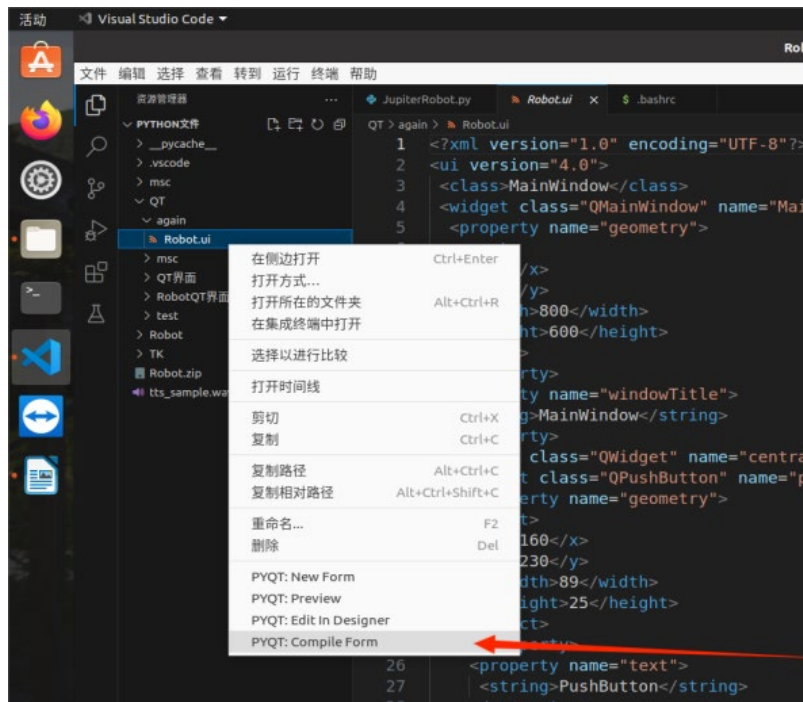
2. 保存文件

点击左上角的保存或 `ctrl+s`，首次保存会弹出下面这个界面，选择保存路径（默认为点击 PYQT: New Form 的路径）和文件名



3. ui 文件转 py 文件

右键点击刚刚创建的 .ui 文件，点击最下面的 PYQT: Compile Form，会自动生成一个 UI_XXX.py 文件。这个文件为 qt 自动生成，每次编辑完 .ui 文件后右键选择最下面的 PYQT: Compile Form 即可自动编辑。这个 py 文件不能直接运行，只是创建了窗体的类，使用需要配合后面的主程序模版。



4. 创建一个主程序模板文件

在同级目录下创建一个 py 文件，如 Main.py

下面是主程序的模版，需要修改 ui 文件改成你自己的。

Main.py 代码

```
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow

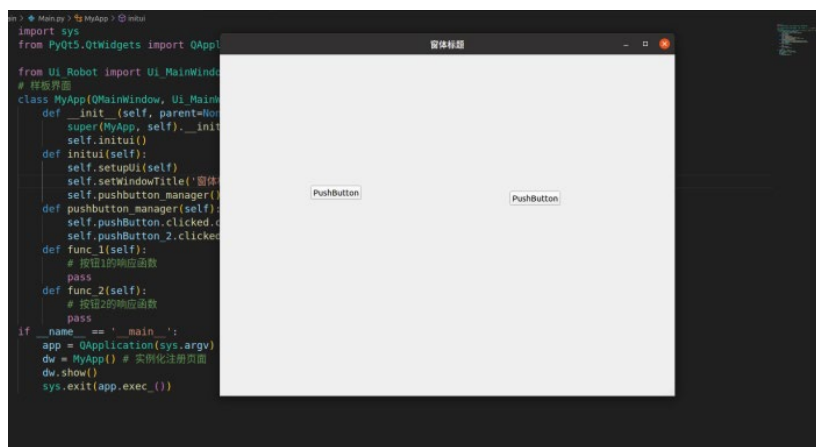
from Ui_Robot import Ui_MainWindow # 这是改成 ui 转出 py 的文件名
# 样板界面
class MyApp(QMainWindow, Ui_MainWindow): # 此处界面是主窗口，即为 QMainWindow
    def __init__(self, parent=None):
        super(MyApp, self).__init__(parent)
        self.initui()
    def initui(self):
        self.setupUi(self)
        self.setWindowTitle('窗体标题')
        self.pushButton_manager() # 绑定槽函数
    def pushbutton_manager(self):
        self.pushButton.clicked.connect(self.func_1)
        self.pushButton_2.clicked.connect(self.func_2)
```

```

def func_1(self):
    # 按钮 1 的响应函数
    pass
def func_2(self):
    # 按钮 2 的响应函数
    pass
if __name__ == '__main__':
    app = QApplication(sys.argv)
    dw = MyApp() # 实例化注册页面
    dw.show()
    sys.exit(app.exec_())

```

可以直接运行 Main.py，显示效果如下



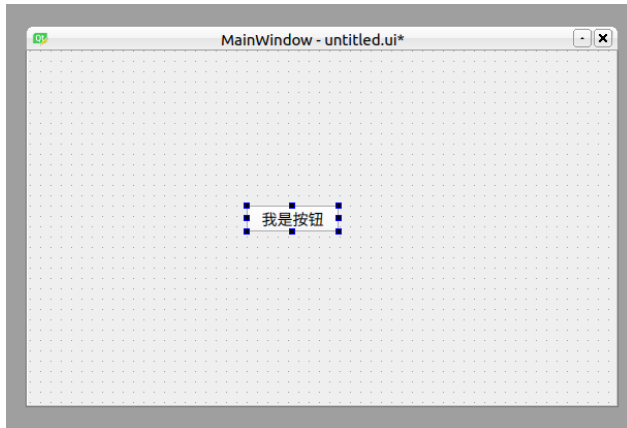
三. 编辑 UI 界面

文件能正常运行后，我们就可以开始编辑 UI 界面了，右键点击.ui 文件，选择 PYQT>Edit In Designer 就可以回到编辑界面，第一步先将界面调整至合适的大小，随后就可以加入各种控件

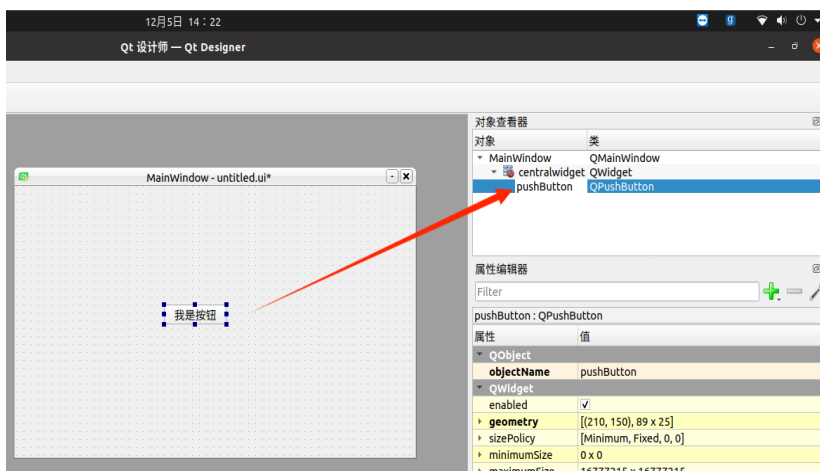
1. 常用功能介绍

(1) PushButton

按钮是制作 UI 界面最常见的控件，QT 也提供了按钮。拖入后双击可以改变按钮上显示的文字



也可以双击右侧的对象查看器中的对象，修改对象名，在按钮较多时，可以按照自己的习惯给每个按钮命名，方便后续绑定函数和管理。



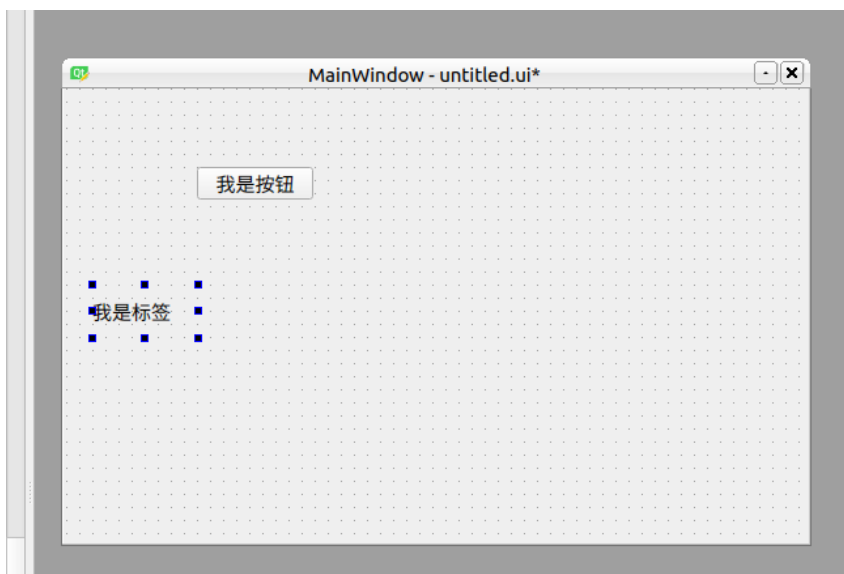
给按钮绑定函数：

```
self.pushButton.clicked.connect(self.cmd)
```

以上示例是将对象名为 pushButton 的按钮绑定点击时触发 cmd 的方法。

(2) Label

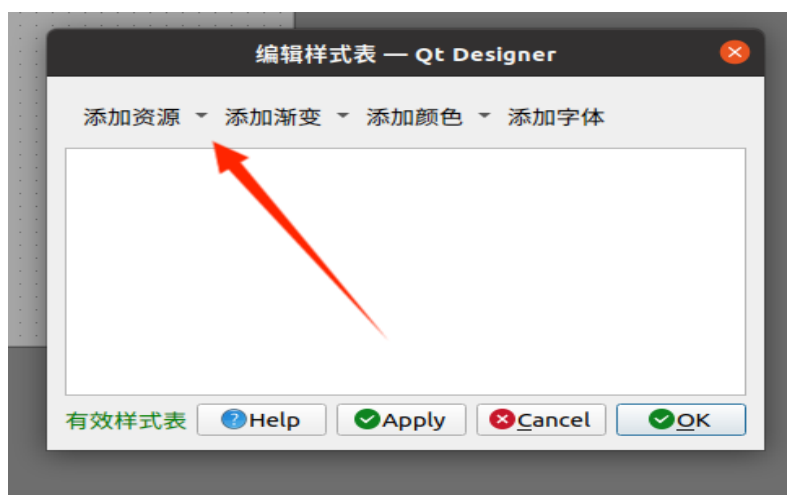
标签可以帮助我们在页面中打入标签。



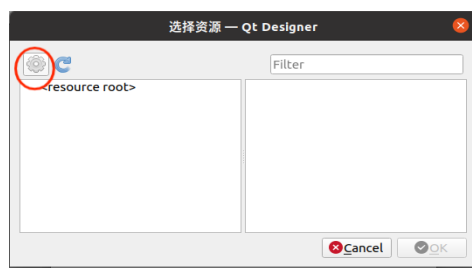
也可以右键点击标签，点击改变样式表，在添加资源中添加背景图片，使标签充当一个画布。



首先需要将标签上的文字删除，再右键点击标签，选择改变样式表，点击添加资源右边下拉菜单里的 background-image。



点击齿轮。



点击蓝色的“打开资源文件”按钮，找到需要的图片文件。

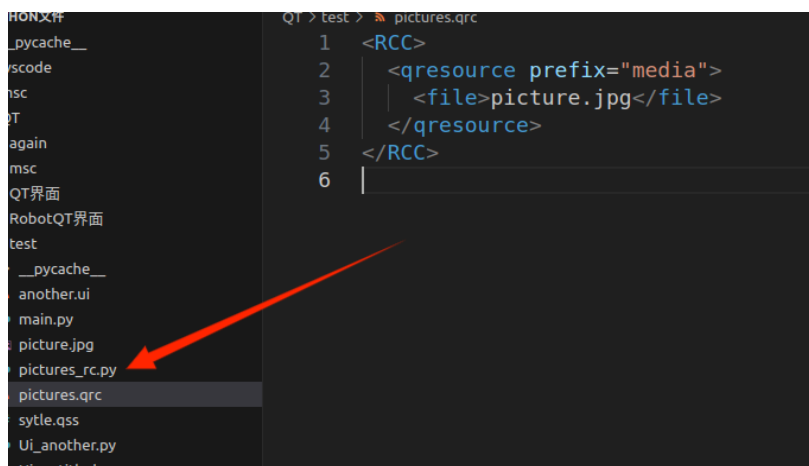


QT 编辑器只能显示.qrc 后缀的文件，所以我们需要先编写一个 qrc 文件

```
QT > test > pictures.qrc
1 <RCC>
2   <qresource prefix="media">
3     <file>picture.jpg</file>
4   </qresource>
5 </RCC>
6
```

```
<RCC>
<qresource prefix="media">
<file>picture.jpg</file>
</qresource>
</RCC>
```

再右键 qrc 文件点击 PYQT: Compile Form，会自动生成一个 py 文件



(3) Stacked Widget 分页工具 可以创建多个页面。



右上角的小三角只会在编辑时显示，运行时不会显示。



可以通过按钮绑定页码实现页面更换。

```
> test > main.py > Main_Window > Buttoncommand
1 import sys
2 from PyQt5.QtWidgets import QApplication, QMainWindow, QVBoxLayout
3 from PyQt5 import QtWidgets
4 from Ui_untitled import Ui_MainWindow as hello_ui
5
6 class Main_Window(QtWidgets.QMainWindow, hello_ui):
7
8     def __init__(self):
9         super(Main_Window, self).__init__()
10        self.setupUi(self)
11        self.buttoncommand()
12
13    def buttoncommand(self):
14        self.pushButton_2.clicked.connect(self.goto1)
15        self.pushButton_3.clicked.connect(self.goto2)
16
17    def goto1(self):
18        self.stackedWidget.setCurrentIndex(0)
19
20    def goto2(self):
21        self.stackedWidget.setCurrentIndex(1)
22
```

```
def goto1(self):
    self.stackedWidget.setCurrentIndex(0)

def goto2(self):
    self.stackedWidget.setCurrentIndex(1)
```

四. 界面美化

1. 添加样式文件

创建一个 style 文件夹，并创建一个 style.qss 的样式文件，类似于 css 文件

. 加类名可以更改整个类的样式

#加对象名可以更改某个对象的样式

```
main.py Robot_main.py # style.qss X
QT > test > # style.qss > ...
13 background-color:blueviolet;
14 } */
15
16 #MainWindow
17 {
18     background-color: ■ rgb(247, 248, 248);
19 }
20
21 #tabWidget
22 {
23     background-color:■rgb(190, 153, 224);
24
25 }
26 }
27
28 .QPushButton
29 {
30     border: 0ch;
31     background-color:■rgb(181, 155, 206);
32     border-radius: 3px;
33     border-left: 1px solid □#000000;
34     border-top: 1px solid □#000000;
35     border-right: 3px solid □#000000;
36     border-bottom: 3px solid □#000000;
37 }
```

2. 在主文件中添加更改样式代码

主文件中可以通过 . setStyleSheet 的方法改变对象样式

```
self.setWindowTitle('JupiterRobot')
self.pushButton_manager()
self.stackedWidget.setCurrentIndex(0)
self.stackedWidget_2.setCurrentIndex(0)
self.o1.setStyleSheet("background-color:#e3e1d2;color:#665657;border-bottom: 1px solid
self.pushButton.setStyleSheet("background-color:#e3e1d2;color:#665657;border: 1px soli
self.pushButton_2.setStyleSheet("background-color:#665657;color:#e3e1d2;border: 1px sc
self.pushButton.close()
self.pushButton_2.close()
self.pushButton_4.close()
```