链滴

PyQt5 快速开发与实战（一）

作者：Ricky2020
原文链接：https://ld246.com/article/1612327176642
来源网站：链滴

第一章 认识PyQt5

1.1 PyQt框架简介

1. Python支持的GUI控件集

- PyQt

1. 用户创建GUI应用程序的跨平台工具包。它将Python和Qt库成功融合在一起。Qt库是目前强大的GUI库之一

2. **PyQt5**官网

- Tkinter
- wxPython
- Kivy
- PyGUI
- Libavg

1.1.1 PyQt5 的特点

1. 基于高性能的Qt的GUI控件集
2. 能够跨平台运行在主流系统上
3. 使用信号/槽机制进行通讯
4. 对Qt库的完全封装
5. 可以使用Qt成熟的IDE进行图形界面设计，并自动生成可以执行的Python代码

6. 提供了一整套种类繁多的窗口控件

1.1.2 Qt和PyQt的关系

1. PyQt是Qt框架在Python语言上的实现

2. PyQt向Python程序员提供了完整的Qt应用程序接口的函数，几乎可以用Python做任何Qt想做的情

3. 基于Python语言开发Qt的代码将会是原来使用C++开发的50%~60%

4. Qt的信号/槽机制相比于回调函数(callback)机制更加安全简洁

1.1.3 其他图形界面开发库的介绍

1. Tkinter

- Tkinter是绑定了Python的Tk GUI工具集
- 就是Python包装的tcl代码，通过内嵌在python解释器内部的Tcl解释器实现的。
    1. 调用Tkinter库
    2. 将其转换为Tcl命令
    3. Tcl解释器进行命令的解释执行

2. wxPython

- wxPython是Python对跨平台的GUI工具集的wxWidgets(用C++编写)的包装，作为Python的个扩展模块来实现
- wxPython是比较流行的Tkinter的一个代替品，跨平台性比较良好

3. PyGTK

- PyGTK是Python对GTK+GUI库的一系列包装，在windows系统上不友好，毕竟是使用的GTK的UI库

4. PySide----目前已经停止更新了，最新的才支持Qt 4.8版本

- PySide目前是由Qt官方在维护，是Python对跨平台的GUI工具集Qt的另一个包装，捆绑在Python当中

1. 上述的前三个库，没有 Qt Designer---可视化创建UI文件，可以通过工具快速编译为Python文件
2. 最后一个库也已经停止更新了

1.1.4 PyQt4/PyQt5

0. PyQt5不再向下兼容使用PyQt4编写的程序，因为PyQt5进行了重大变动

1. PyQt5不再去支持Python2.6之前的版本了，官方只提供了Python3.x版本的安装包

2. PyQt5对一些模块进行了重构，废弃了好一些模块

3. PyQt5对网页的支持与时俱进，PyQt4对网页的支持的基础引擎是WebKit，对互联网新生事物尤是JavaScript的支持不是很完美，PyQt5则是使用了谷歌团队的chromium内核引擎，基本上完美支互联网的新生事物

4. PyQt5仅支持新式的信号和槽，对旧式的不再去支持
5. PyQt5不再去支持Qt5.0中标记已经废弃或者过时的API
6. PyQt5在程序需要时才去释放GIL,而PyQt4是在执行完毕程序后强制释放GIL
7. 官方对PyQt4不再会有重大的更新和维护了，所以目前主流的还是使用PyQt5

1.2 PyQt5环境搭建

1.2.1 在Windows下搭建PyQt5环境

1. 安装Python环境　　建议3.5.3+
2. 安装PyQt以及其常用工具　　建议5.9+

1. pip install -i https://pypi.douban.com/simple --trusted-host pypi.douban.com PyQt5-tools
    + 其实推荐直接安装这个就完事了，因为先安装下面的那个在安装这个，还是会根据兼容性重新安PyQt5的
    + 如果直接先安装这个报错的话，那么就先安装下面的PyQt5再去安装这个(一般不会报错的)
2. pip install -i https://pypi.douban.com/simple --trusted-host pypi.douban.com PyQt5

3. Eric　　建议6.17

1.2.3 在Windows环境下PyQt5的安装测试

1. cmd 进入安装PyQt5的python环境
2. import PyQt5 不报错说明安装成功
3. help(PyQt5) 查看PyQt5所依赖的模块

1.2.4 安装其他Python模块

pip install xxxx

1.2.5 使用PyQt5的API文档

1. dir()可以查看一个类或者对象的相关属性
2. help()可以查看一个类或者对象的相关说明文档

为了方便查阅说明文档，我们可以使用代码将其输出写入文件中

```
import sys
from PyQt5.QtWidgets import QWidget
out = sys.stdout
sys.stdout = open('xxx.txt', 'w', encoding='utf-8')
help(QWidget)
sys.stdout.close()
sys.stdout = out
```

1.3 Eric6的安装与使用

概念 Eric是一个全功能的Python编辑器和IDE，与PyQt5的结合，非常方便实现界面与业务逻辑的分，快速开发GUI

Eric6的优点：

1. 跨多平台良好
2. 调试器支持设置断点、单步调试、查看变量值等
3. 支持工程
4. 支持自动补全
5. 支持智能感知
6. 支持自动语法检查
7. 自持缩进。会自动判断流程控制语句
8. 支持代码折叠
9. 支持很多小工具，比如正则表达式、测试器
10. 与Qt Designer结合很好，方便GUI的开发
11. 支持代码的版本管理 如SVN
12. 使用PyQt5作为图形库，界面美观、简洁
13. 支持在线自动更新

1.3.1 Eric的安装以及汉化

1. 下载Eric

下载链接

汉化包

2. 解压两个压缩包

- 将汉化包中的文件全部拷贝到Eric中进行覆盖操作

3. 在正式安装之前 先安装Qsci模块

```
pip install -i https://pypi.douban.com/simple --trusted-host pypi.douban.com QScintilla
```

4. 在指定环境安装

```
### 切换到eric的目录下安装
python install.py
### 1. 没有报错就表示安装成功
### 2. 安装完成之后，如果在eric6文件夹下没有生成eric.bat文件，就需要进入eric6\eric文件夹中击eric6.pyw文件
```

小技巧 可以直接安装Qsci模块。它会连带着将依赖模块一起安装

小坑爹 最好不要使用虚拟环境去安装Eric，要是使用推荐使用Adaconda

```
1. 安装成功Eric之后，桌面上的快捷方式会在打不开
2. 因为pythonw.exe在真实环境与虚拟环境中的位置不一样，此时我们可以使用notepad++等第三编辑器打开那个cmd文件
```

3. 修改pythonw.exe文件的正确位置即可解决打不开报错等坑爹问题
4. 或者直接将真实环境下的pythonw3.exe pythonw.exe pythonw.pdb(pythonw相关的文件复制到拟环境指定位置)

无敌踩坑

1. 无法定位程序输入点 OPENSSL_sk_new_reserve 于动态链接库C:\Windows\System32\libssl-1_1x64.dll上

- 将虚拟环境下的基础环境中D:\Program Files\Python37\DLLs中的libssl-1_1-x64.dll复制到上报错目录(记得先备份，以免后续出问题)

2. pythonw.exe虚拟环境下不存在的问题

- 直接将真实环境下的pythonw3.exe pythonw.exe pythonw.pdb(pythonw相关的文件复制到拟环境指定位置)

3. Eric-Code Info Provider:No source code documentation provider has been registered

- issue方案

1.3.2 Eric的相关配置

1.3.3 安装自动补全插件jedi

1. jedi是一个超级棒的Python自动补全库，可以在很多编辑器以及IDE中使用

```
pip install -i https://pypi.douban.com/simple --trusted-host pypi.douban.com jedi
```

2. 在菜单中选择插件进行Eric的插件安装

1.3.4 测试Eric6

```
import sys
from PyQt5.QtWidgets import QWidget, QApplication
if __name__ == '__main__':
    app = QApplication(sys.argv)
    q = QWidget()
    q.resize(360, 360)
    q.show()
    sys.exit(app.exec_())
```

1.3.5 Eric的基本使用

第二章 Python的基础知识

第三章 Qt Designer的使用

3.1 Qt Designer快速入门

1. Qt Designer是一款强大灵活并且可视化的GUI设计工具
2. .ui文件有两种方式转换为py文件，并被其他python文件引用
   - 命令的方式
   - 手工的方式(Eric手工转换)
3. Qt Designer优点：
   - 使用简单，可以通过拖拽点击的方式完成UI设计
   - 转换python文件简单

路径一般是在安装的PyQt5-tools中

3.1.1 新建主窗口

1. 新建一个Main窗口

3.1.2 窗口主要区域介绍

3.1.3 查看UI文件

3.1.4 将.ui文件转换为.py文件

Qt Designer设计的界面保存为ui文件，内容结构类似XML，我们需要的是py文件，就需要将ui文件换为py文件

1. ui-->py文件的转换方法
   - 通过Eric6将ui文件转换为py文件
     1. 打开一个Eric项目，然后切换到窗体
     2. 右键添加一个窗体
     3. 右键编译窗体
     4. 返回到源代码目录会发现多了一个Ui_xxx.py文件
   - 通过命令行把ui文件转换为py文件
     1. PyQt5安装成功后，pyuic5命令默认安装在对应的python的Scripts目录下
     2. pyuic5 -o xxxx.py xxxx.ui 就可以将ui文件转换为py文件
   - 通过python脚本把ui文件转换为py文件

```
import os
CurrDir = './'
def listUiFile():
    FileList = []
    files = os.listdir(CurrDir)
    for fileName in files:
        if os.path.splitext(fileName)[1] == '.ui':
```

```
            FileList.append(fileName)
    return FileList

def ui2py(UiFileName):
    return os.path.splitext(UiFileName)[0] + '.py'

def runMain():
    FileList = listUiFile()
    for uifile in FileList:
        pyfile = ui2py(uifile)
        cmd = 'pyuic5 -o {} {} '.format(pyfile, uifile)
        os.system(cmd)

if __name__ == '__main__':
    runMain()
```

注意后两种方式转换出来py文件后，需要加上如下代码才能真正运行

```
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

3.1.5 界面与逻辑分离

我们要做的其实很简单只需要继承ui转换的那个py文件中的窗口类

实现如下代码就可以运行

```
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

小技巧可以将MainWindow = QtWidgets.QMainWindow()、ui = Ui_MainWindow()、ui.setupUi MainWindow)这三行代码的实现放进我们派生类的构造函数中即 **init**()方法中

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

class MyMainWindow(QMainWindow,Ui_MainWindow):
    def __init__(self):
        super().__init__()
```

```
        self.setupUi(self)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    myWin = MyMainWindow()
    myWin.show()
    sys.exit(app.exec_())
```

解析个人见解

1. 将这行代码MainWindow = QtWidgets.QMainWindow()放进QMainWindow这个父类中的__init_()中实现

2. ui = Ui_MainWindow()、ui.setupUi(MainWindow)将这两行代码放进派生类的__init__()中实现

3. 因为派生类继承Ui那个类，所以self就是ui对象，而派生类也继承自QMainWindow，所以self同也是QMainWindow对象，所以有self.setupUi(self)

3.2 布局管理

小知识普通小控件的基类是QtWidgets.QFrame

四大布局

1. 垂直布局

概念控件默认按照从上至下的顺序进行纵向添加

2. 水平布局

概念控件默认按照从左到右的顺序进行纵向添加

3. 栅格布局

概念将窗口控件放入一个网格之中，然后将它们合理地划分成若干行row和列column，并把其中的个窗口控件放置在合适的单元cell中，这里的单元即是由行列交叉划分出来的空间

4. 表单布局

概念控件以两列的形式布局在表单中，左侧是标签，右侧是输入控件

提示一般布局有两种方式

5. 通过布局管理器显式指明使用什么布局(右键选择使用怎样的布局)

6. 通过容器控件进行布局(如拖拽左侧的容器控件)

3.2.1 使用布局管理器

3.2.2 使用容器进行布局

本质使用容器进行控件布局其实还是要调用布局管理器进行统筹布局

3.3 Qt Designer 实战应用

了解属性编辑器的四大属性：

1. geometry

2. sizePolicy

3. minimumSize

4. maximumSize

3.3.1 绝对布局

geometry----[(215,140),75*23] 距离窗口左上角(215,140) 大小(75x23)

```
self.pushButton = QtWidgets.QpushButton(self.centralwidget)
self.pushButton.setGeometry(QtCore.QRect(215,140,75,23)
self.pushButton.setObjectName("pushButton")
```

3.3.2 使用布局管理器布局

1. 垂直布局

将几个控件使用垂直布局管理器布局后，其geometry属性都是灰色的，表示它们的位置和大小已经垂直布局管理器接管了

2. 栅格布局

grid.Layout.addWidget(窗口控件，行位置，列位置，要合并的行数，要合并的列数)后两个参数是选的 默认就是1，1表示占据当前的行列位置，不进行合并操作

3. 水平布局

1. Vertical Spacer 表示两个水平方向上布局不要彼此挨着
2. Horizontal Spacer 表示垂直方向上的布局不要紧挨
3. Horizontal Line 表示某个控件跟左侧的的布局不是一个类别的，所以用一条线将其分开来

一个基本原则PyQt有一个基本原则

主窗口中的所有窗口控件都有自己的父类

4. minimumSize和maximumSize属性

小提示每个控件都有自己的两个尺寸

1. sizeHint　　尺寸提示　窗口控件的期望尺寸
2. minimumSize　　最小尺寸　窗口控件的压缩最小尺寸

5. sizePolicy 属性

作用

如果窗口控件在布局管理器中的布局不能满足我们的需求，那么就可以设置sizePolicy来实现布局的调

3.3.3 其他需要注意的内容

重点使用Qt Designer 开发一个完整的GUI程序流程如下

1. Qt Designer 布局的顺序

- 将一个窗口控件拖入窗口中并放置在大致的正确位置上，除了容器container窗口，一般不需要

整窗口的尺寸大小

- 对于要用代码引用的窗口控件，应指定一个名字，对应需要微调的窗口控件，可以设置其对应属性
- 重复以上步骤，直到所需要的全部窗口控件都放到了窗口中
- 如有需要，在窗口控件之间可以用分割线以及距离控件
- 选择需要布局的窗口控件，使用布局管理器或者分窗口对它们进行布局
- 重复上一步，直到所有窗口控件和分隔符都布局好为止
- 单击窗口，并使用布局管理器对齐进行布局
- 为窗口的标签设置伙伴关系
- 如果按键次序有问题，则需要设置窗口的Tab键次序
- 在适当的地方为内置的信号和槽建立信号与槽的链接
- 预览窗口，并检查所有内容是否按照预想进行工作
- 设置窗口的对象名。窗口的标题并进行保存
- 使用Eric等类似工具编译窗口，然后根据需要生成对应的对话框代码
- 进行正常的代码编写工作，即编写业务逻辑文件

2. 使用布局管理器对窗体进行布局

针对窗体的布局，那是会将布局管理器布满整个窗体屏幕的，一般用不到

小技巧当我们布局时出现很多步错误时，撤销比较麻烦时，可以使用单击窗体空白处，选择布局--->破布局(还原到初始布局状态)

3. 设置伙伴关系

- 在display widgets名字前加上&
- 名字需要是英文的
- Alt+S快捷键可以快速定位伙伴关系

4. 设置Tab键顺序

- 菜单---Edit----设置Tab键次序
- 鼠标右键----制表符顺序列表--上下拖动控件进行排序

3.3.4 测试程序

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(800, 600)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.layoutWidget = QtWidgets.QWidget(self.centralwidget)
        self.layoutWidget.setGeometry(QtCore.QRect(0, 130, 821, 171))
        self.layoutWidget.setObjectName("layoutWidget")
        self.horizontalLayout = QtWidgets.QHBoxLayout(self.layoutWidget)
```

```
        self.horizontalLayout.setContentsMargins(0, 0, 0, 0)
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout")
        self.label_6 = QtWidgets.QLabel(self.layoutWidget)
        self.label_6.setText("")
        self.label_6.setObjectName("label_6")
        self.verticalLayout.addWidget(self.label_6)
        self.label = QtWidgets.QLabel(self.layoutWidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePo
icy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(1)
        sizePolicy.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth())
        self.label.setSizePolicy(sizePolicy)
        self.label.setObjectName("label")
        self.verticalLayout.addWidget(self.label)
        self.label_2 = QtWidgets.QLabel(self.layoutWidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePo
icy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(3)
        sizePolicy.setHeightForWidth(self.label_2.sizePolicy().hasHeightForWidth())
        self.label_2.setSizePolicy(sizePolicy)
        self.label_2.setObjectName("label_2")
        self.verticalLayout.addWidget(self.label_2)
        self.label_3 = QtWidgets.QLabel(self.layoutWidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePo
icy.Preferred)
        sizePolicy.setHorizontalStretch(1)
        sizePolicy.setVerticalStretch(1)
        sizePolicy.setHeightForWidth(self.label_3.sizePolicy().hasHeightForWidth())
        self.label_3.setSizePolicy(sizePolicy)
        self.label_3.setObjectName("label_3")
        self.verticalLayout.addWidget(self.label_3)
        self.horizontalLayout.addLayout(self.verticalLayout)
        spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidge
s.QSizePolicy.Minimum)
        self.horizontalLayout.addItem(spacerItem)
        self.gridLayout = QtWidgets.QGridLayout()
        self.gridLayout.setObjectName("gridLayout")
        self.label_5 = QtWidgets.QLabel(self.layoutWidget)
        self.label_5.setObjectName("label_5")
        self.gridLayout.addWidget(self.label_5, 0, 0, 1, 1)
        self.label_4 = QtWidgets.QLabel(self.layoutWidget)
        self.label_4.setObjectName("label_4")
        self.gridLayout.addWidget(self.label_4, 0, 1, 1, 1)
        self.doubleSpinBox_returns_min = QtWidgets.QDoubleSpinBox(self.layoutWidget)
        self.doubleSpinBox_returns_min.setObjectName("doubleSpinBox_returns_min")
        self.gridLayout.addWidget(self.doubleSpinBox_returns_min, 1, 0, 1, 1)
        self.doubleSpinBox_returns_max = QtWidgets.QDoubleSpinBox(self.layoutWidget)
        self.doubleSpinBox_returns_max.setObjectName("doubleSpinBox_returns_max")
        self.gridLayout.addWidget(self.doubleSpinBox_returns_max, 1, 1, 1, 1)
        self.doubleSpinBox_maxdrawdown_min = QtWidgets.QDoubleSpinBox(self.layoutWidget
```

```
        self.doubleSpinBox_maxdrawdown_min.setObjectName("doubleSpinBox_maxdrawdown
min")
        self.gridLayout.addWidget(self.doubleSpinBox_maxdrawdown_min, 2, 0, 1, 1)
        self.doubleSpinBox_maxdrawdown_max = QtWidgets.QDoubleSpinBox(self.layoutWidget

        self.doubleSpinBox_maxdrawdown_max.setObjectName("doubleSpinBox_maxdrawdown
max")
        self.gridLayout.addWidget(self.doubleSpinBox_maxdrawdown_max, 2, 1, 1, 1)
        self.doubleSpinBox_sharp_min = QtWidgets.QDoubleSpinBox(self.layoutWidget)
        self.doubleSpinBox_sharp_min.setObjectName("doubleSpinBox_sharp_min")
        self.gridLayout.addWidget(self.doubleSpinBox_sharp_min, 3, 0, 1, 1)
        self.doubleSpinBox_sharp_max = QtWidgets.QDoubleSpinBox(self.layoutWidget)
        self.doubleSpinBox_sharp_max.setObjectName("doubleSpinBox_sharp_max")
        self.gridLayout.addWidget(self.doubleSpinBox_sharp_max, 3, 1, 1, 1)
        self.horizontalLayout.addLayout(self.gridLayout)
        self.line = QtWidgets.QFrame(self.layoutWidget)
        self.line.setFrameShape(QtWidgets.QFrame.VLine)
        self.line.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line.setObjectName("line")
        self.horizontalLayout.addWidget(self.line)
        spacerItem1 = QtWidgets.QSpacerItem(400, 20, QtWidgets.QSizePolicy.Preferred, QtWid
ets.QSizePolicy.Minimum)
        self.horizontalLayout.addItem(spacerItem1)
        self.pushButton = QtWidgets.QPushButton(self.layoutWidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Maximum, QtWidgets.QSizeP
licy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.pushButton.sizePolicy().hasHeightForWidth())
        self.pushButton.setSizePolicy(sizePolicy)
        self.pushButton.setObjectName("pushButton")
        self.horizontalLayout.addWidget(self.pushButton)
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.toolBar = QtWidgets.QToolBar(MainWindow)
        self.toolBar.setObjectName("toolBar")
        MainWindow.addToolBar(QtCore.Qt.TopToolBarArea, self.toolBar)
        self.toolBar_2 = QtWidgets.QToolBar(MainWindow)
        self.toolBar_2.setObjectName("toolBar_2")
        MainWindow.addToolBar(QtCore.Qt.TopToolBarArea, self.toolBar_2)
        self.toolBar_3 = QtWidgets.QToolBar(MainWindow)
        self.toolBar_3.setObjectName("toolBar_3")
        MainWindow.addToolBar(QtCore.Qt.TopToolBarArea, self.toolBar_3)
        self.label_3.setBuddy(self.doubleSpinBox_sharp_min)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
        MainWindow.setTabOrder(self.doubleSpinBox_returns_min, self.doubleSpinBox_maxdra
down_min)
        MainWindow.setTabOrder(self.doubleSpinBox_maxdrawdown_min, self.doubleSpinBox_s
```

```
arp_min)
        MainWindow.setTabOrder(self.doubleSpinBox_sharp_min, self.doubleSpinBox_returns_m
x)
        MainWindow.setTabOrder(self.doubleSpinBox_returns_max, self.doubleSpinBox_maxdra
down_max)
        MainWindow.setTabOrder(self.doubleSpinBox_maxdrawdown_max, self.doubleSpinBox_
harp_max)
        MainWindow.setTabOrder(self.doubleSpinBox_sharp_max, self.pushButton)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.label.setText(_translate("MainWindow", "&收益"))
        self.label_2.setText(_translate("MainWindow", "最大回数"))
        self.label_3.setText(_translate("MainWindow", "&sharp比"))
        self.label_5.setText(_translate("MainWindow", "最小值"))
        self.label_4.setText(_translate("MainWindow", "最大值"))
        self.pushButton.setText(_translate("MainWindow", "开始"))
        self.toolBar.setWindowTitle(_translate("MainWindow", "toolBar"))
        self.toolBar_2.setWindowTitle(_translate("MainWindow", "toolBar_2"))
        self.toolBar_3.setWindowTitle(_translate("MainWindow", "toolBar_3"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```