目录

[**wxPython**](#wxPython)

[**QtPy5**](#QtPy5)

[tkinter](#tkinter)

1. wxPython模块：
2. wx.Frame框架：
   1. frame = wx.Frame(parent 框架的父窗口。若为顶级窗口，则为None。
      1. , id = -1 新窗口ID号，-1表示自动生成。
      2. , title = “” 标题
      3. , pos = wx.DefaultPosition 新窗口左上角的坐标，默认值为（-1，-1）表示系统决定。
      4. , size = wx.DefaultSize 新窗口初始尺寸，默认值为（-1，-1）表示系统决定。
      5. , style = wx.DEFAULT\_FRAME\_STYLE 指定窗口类型的常量，可使用、运算来组合它们。
      6. , name = “frame”) 框架内在名字。可用来寻找窗口。
3. 创建wx.Frame子类：
   1. class MyFrame(wx.Frame):
      1. def \_\_init\_\_(self, …):
         1. wx.Frame.\_\_init\_\_(…)
   2. app = wx.App()
   3. frame = MyFrame(…)
   4. frame.Show()
   5. app.MainLoop()
4. 使用wx.Frame前需要创建wx.APP类：
   1. 只需一个窗口：
      1. app = wx.App()
      2. frame = wx.Frame(…)
      3. frame.show()
      4. app.MainLoop()
   2. 需多个窗口（创建app子类）：
      1. class App(wx.app):
      2. def OnInit(self):
      3. frame = wx.Frame(…)
      4. frame.show()
      5. return True
      6. app = App()
5. StaticText(静态文本部件)：
   1. panel = wx.Panel(self) 创建画板
   2. title = wx.StaticText(...) 创建标题
   3. font = wx.Font(pointSize 字体整数尺寸，磅
      1. , family 指定字体
      2. , style 是否倾斜
      3. , weight 醒目程度
      4. , underline = False 仅Windows，是否下划线
      5. , faceName = “” 字体名
      6. , encoding = wx.FONTENCODING\_DEFAULT) 选择编码
      7. wx.Font(16, wx.DEFAULT, wx.FONTSTYLE\_NORMAL, wx.NORMAL) 创建字体
   4. title.SetFont(font) 设置字体
   5. wx.StaticText(parent, id, label, pos = …, size = …, style = 0, name = “staticText”) 把画板设为parent
      1. 在指定位置创造内容为label的静态文字。
6. TextCtrl（输入文本类）：
   1. wx.TextCtrl(parent, id, value = “”, pos = wx.DefaultPosition, size = wx.DefaultSize, style = 0, validator =
   2. ----------wx.DefaultValidator, name = wx.TextCtrlNameStr)
      1. style:
         1. wx.TE\_Center 文本居中
         2. wx.TE\_LEFT 文本左对齐
         3. wx.TE\_NOHIDESEL 文本高亮
         4. wx.TE\_PASSWORD \*号代替文本
         5. wx.TE\_PROCESS\_ENTER 按下Enter键时触发一个文本输入事件
         6. wx.TE\_PROCESS\_TAB 若指定，按下Tab时创建一个通常的字符事件
         7. wx.TE\_READONLY 只读，用户无法修改
         8. wx.TE\_RIGHT 右对齐
      2. value：初始文本
      3. validator：过滤数据以确保只键入要接收的数据
7. Button（按钮类）：
   1. xxx = wx.Button(parent, id, label, pos, size = wx.DefaultSize, style = 0, validator, name = “button”)
      1. 参数参考wx.TextCtrl
8. BoxSizer布局：
   1. vsizer = wx.BoxSizer(wx.VERTICAL) 创建BoxSizer实例
      1. wx.VERTICAL 垂直 wx.HORIZONTAL 水平
   2. vsizer.Add(control, proportion, flag, border)
      1. control：要添加的控件
      2. proportion：与窗口大小的关系。0 : 固定大小，1 : 等比大小，2 : 比窗口放大更多。
      3. flag：指定控件所需边距和对齐方式。
         1. wx.LEFT 左边距 wx.RIGHT 右边距 wx.BOTTOM 底边距 wx.TOP 上边距 wx.ALL 以上四个
         2. wx.ALIGN\_LEFT 左对齐 wx.ALIFN\_RIGHT 右对齐 …TOP …BOTTOM
         3. …\_CENTER\_VERTICAL 垂直对齐 …\_CENTER\_HORIZONTAL 水平对齐 …\_CENTER 居中对齐
         4. wx.EXPAND 占有sizer定位方向上所有可用空间
         5. 标记之间用 | 连接
         6. wx.LEFT | wx.RIGHT | wx.EXPAND 指定border后会自动拉伸至左右间隔为指定大小
      4. border：所指定边距的大小。
   3. panel.SetSizer(Sizer) 应用Sizer
9. 事件：
   1. button.Bind(wx.EVT\_BUTTON, 方法名) 为按钮绑定事件，事件发生时执行方法。
      1. wx.EVT\_MOTION wx.ENTER\_WINDOW wx.LEAVE\_WINDOW wx.EVT\_MOUSEWHEEL
   2. def 方法名(self, event):
      1. pass
   3. wx.MessageBox(message) 弹出信息为message的提示框
10. **QtPy5：**
    1. 控件类：
       1. QWidget：窗体容器
       2. QLabel：标签，用于设置文字、图片。
       3. QTableView：显示信息表格
       4. QPushButton：按钮
       5. QTextEdit：文本输入框
       6. QListWidget：列表
       7. QTableWidget：表格
       8. QCheckBox：勾选框
       9. QMessageBox：消息框
       10. QComboBox：下拉框
    2. 代码调试：
       1. from PyQt5 import QtCore, QtGui, QtWidgets
       2. setupUi()方法：
          1. MainWindow.resize(790, 786) 设置窗体大小
             1. .setMinimumSize(QtCore.QSize(960, 786)) 设置窗体最小值
             2. .setMaximumSize(QtCore.QSize(960, 786)) 设置窗体最大值
       3. QPushButton：
          1. 按钮名 .事件【pressed】.connect(方法名)
       4. QComboBox：
          1. 下拉框名 .事件【activated[str]】.connect(方法名) 当选择某选项时触发，将该选项的内容传入方法
          2. def 方法名(self, text):
       5. QTextEdit：
          1. text = 输入框名 .toPlainText() 返回输入框的内容，文件路径会自动转义
          2. 输入框名 .setText(字符串) 设置输入框内容为字符串
       6. QMessageBox：
          1. QtWidgets.QMessageBox.warning(父窗体，标题，内容，按钮们，关闭时返回值)（按钮之间用 | 隔开）
          2. QtWidgets.QMessageBox.about(父窗体，标题，内容，按钮们，关闭时返回值) 关于对话框
          3. QtWidgets.QMessageBox.question(父窗体，标题，内容，按钮们，关闭时返回值) 询问对话框
          4. QtWidgets.QMessageBox.information(父窗体，标题，内容，按钮们，关闭时返回值) 消息对话框
          5. QtWidgets.QMessageBox.critical (父窗体，标题，内容，按钮们，关闭时返回值) 错误对话框
          6. 按钮与返回值：返回值为按钮值
             1. QtWidgets.QMessageBox.Ok QtWidgets.QMessageBox.Yes QtWidgets.QMessageBox.No
       7. QListWidget：
          1. 列表名 .addItem(字符串) 添加字符串进列表
          2. 列表名 .clear() 清空列表
       8. QListWidgetItem：
          1. item = QtWidgets.QListWidgetItem(QListWidget对象) 创建列表项
          2. item.setText(‘xxx’) 设置项文本
          3. item.setToolTip(‘xxx’) 设置鼠标在项上悬浮时的提示
       9. QTableWidget：
          1. table.insertRow(num) 添加序号为num的新行
          2. table.setItem(num, 0, xxx) 对num行的第0项进行设置
          3. table.horizontalHeader().setStretchLastSection(True) 使表格的最后一列适应容器大小
       10. QFileDialog：
           1. fname = QFileDialog.getOpenFileName(None, '选择文件', 'D:\\', ‘图片文件(\*.jpeg;\*)’) 选择文件
              1. 第三个参数：起始路径
              2. 选择文件类型（第四个参数）：括号外随意命名，\*.xxx为指定后缀，\*表示所有文件
              3. 返回值：(绝对路径, 选择文件类型)
       11. QPixmap:
           1. QPixmap(‘image.png’) 打开图片，返回Pixmap对象
           2. .scaled(QSize对象，QtCore.Qt.IgnoreAspectRatio, QtCore.Qt.SmoothTransformation)
       12. QMainWindow：
           1. MainWindow.setFixedSize(MainWindow.width(), MainWindow.height()) 固定窗体大小并禁止最大化
       13. QMovie：
           1. Gif图片：
              1. gif = QtGui.Qmovie(‘xx.gif’) 生成Qmovie对象
              2. label.setMovie(gif) 设置到标签
              3. gif.start() \ gif.stop() \ label.clear()
       14. 窗体显示：
           1. import sys
           2. def show\_MainWindow():
              1. app = QtWidgets.QApplication(sys.argv) 实例化QApplication类，作为主程序入口
              2. MainWindow = QtWidgets.QMainWindow() 创建QMainWindow
              3. ui = Ui\_MainWindow() 实例化UI类
              4. ui.setupUi(MainWindow) 设置窗体UI
              5. MainWindow.show() 显示窗体
              6. sys.exit(app.exec\_()) 窗口创建完成后，结束主循环
       15. 图片标签设置：
           1. from PyQt5.QtGui import QPalette,QPixmap,QColor 导入QtGui模块
              1. ->class Ui\_MainWindow
              2. ->def setupUI(self, MainWindow)

title\_img = QPixmap(‘image\\image1.png’) 打开位图image1.png

self.lable\_title\_img.setPixmap(title\_img) 设置调色板

* + 1. widget背景图片设置：
       1. from PyQt5.QtGui import QPalette,QPixmap,QColor 导入QtGui模块
          1. ->class Ui\_MainWindow
          2. ->def setupUI(self, MainWindow)
          3. ->对应widget代码的下方

self.widget\_query.setAutoFillBackground(True) 开启自动填充背景

palette = QPalette() 实例化调色板类

palette.setBrush(QPalatte.Background或MainWindow.backgroundRole(), QtGui.QBrush(QtGui.QPixmap(‘image\\image2.png’)))

适应窗口大小设置：

palette.setBrush(MainWindow.backgroundRole(), QBrush(QPixmap(‘image.png’).scaled(MainWindow.size(), QtCore.Qt.IgnoreAspectRatio, QtCore.Qt.SmoothTransformation)))

self.widget\_query.setPalette(pallete)

1. 内置tkinter模块：
   1. import tkinter.messagebox:
      1. tkinker.messagebox.showinfo(‘提示’，’xx’) 弹出提示消息框
      2. tkinker.messagebox.showwarning(‘警告’，’xx’) 警告框
      3. tkinker.messagebox.showerror(‘错误’，’xx’) 错误框
      4. result = tkinker.messagebox.askokcancel(‘询问’，’xx’) 确定和取消的询问框，返回True,False
      5. result = tkinker.messagebox.askquestion(‘询问’，’xx’) 是和否两个按钮的询问框，返回’yes’,’no’
      6. result = tkinker.messagebox.askretrycancel(‘询问’，’xx’) 重试和取消的询问框，返回True,False