PyQt5

# 模块介绍:

PyQt中有非常多的功能模块,开放中最常用的的功能模块有三个:

* **QtCore:** 包含了核心的非GUI的功能,主要和时间,文件与文件夹,各种数据,流,URLs,mime类文件,进程与线程一起使用.
* **QtGui:** 包含了窗口系统,事件处理,2D图像,基本绘画,字体和文字类.
* **QWidgets:** 包含了一些列创建桌面应用的UI元素.

可以参考PyQt官网的所有模块.地址: [Modules — PyQt Documentation v5.15.4 (riverbankcomputing.com)](https://www.riverbankcomputing.com/static/Docs/PyQt5/module_index.html#ref-module-index)

C++具体实现的API文档,地址: [Qt Widgets C++ Classes | Qt Widgets 5.15.13](https://doc.qt.io/qt-5/qtwidgets-module.html)

# PyCharm External Tools

## QtDesigner

## PyUIC

## Pyrcc

# 基本窗口配置

窗口内的所有控件,若想在窗口中显示,都需要标识的父亲是谁,而不能直接show函数显示.

## 设置窗口icon

可以下载icon图标网站:

w.setWindowIcon(QIcon(r"C:\Sensirion\icons\Temperature.svg"))

## 生成ui文件,并在代码里调用

在Qt Designer里保存后会生成 .ui的文件, 然后在代码里可以用如下方式调用:

app = QApplication(sys.argv)  
ui = uic.loadUi(r".\firstQt.ui")  
ui.show()  
app.exec()

## 使得控件随窗口缩放而缩放的方法

点击窗口类，再选择标题栏上的布局

Graphical user interface, text

Description automatically generated

## ui文件转换成py文件

**pyuic5 -o demo.py demo.ui**  该命令就可以转换

可以在这个地址查看有没有安装pyuic5: C:\work\virtenvs\3p8\_64bit\_SensiPython\_4p0\Scripts

# Qt Designer快捷键:

Ctrl + r: 预览

Ctrl + 上下键: 微调

Alt + Enter: 代码里有一个函数或类,但是没有事前导入库, 可以用这个自动导入.

# 基本逻辑:

1. 窗口: QWidget

两种创建方式:

* + 1. 在自己写类的时候,可以继承QWidget, class MyWindown(QWidget)
    2. 自己创建窗口: w = QWidget()

Graphical user interface

Description automatically generated

1. 在窗口上面添加布局

layout = QVBoxlayout()

self.setLayout(layout)

1. 在布局上添加控件:

button = QPushButton(“Button”)

layout.addWidget(button)

# Overview

基本控件:

|  |  |  |
| --- | --- | --- |
| **No.** | **Widget** | **Name** |
| 1 | QPushButton | 按钮 |
| 2 | QLabel | 标签 |
| 3 | QTextEdit | 可编辑框 |
| 4 | QToolTip | 文本提示 |
| 5 | QLineEdit | 单行输入框 |
| 6 | QMessageBox | 消息框 |
| 7 | QRadioButton | 单选按钮 |
| 8 | QComboBox | 下拉列表 |
| 9 | QPixmap | 图片显示 |
| 10 | QGroupBox | 分组框 |
| 11 | QProgressBar | 进度条 |
| 12 | QDialog | 对话框 |
| 13 | QProgressDialog | 进度条对话框 |
| 14 | QCheckBox | 复选框 |
| 15 | QSlider | 滑块 |
| 16 | QStatusBar | 状态栏 |
| 17 | QFileDialog | 文件对话框 |
| 18 | QToolBar | 工具栏 |
| 19 | QSplitter | 分割条 |
| 20 | QMenuBar | 菜单栏 |
| 21 | QScrollBar | 滚动条 |
| 22 | QTimeEdit | 时间编辑框 |
| 23 | QDateEdit | 日期编辑框 |
| 24 | QDateTimeEdit | 时间日期编辑框 |

高级控件:

|  |  |  |
| --- | --- | --- |
| **No.** | **Widget** | **Name** |
| 1 | QListWidget | 列表控件 |
| 2 | QTableWidget | 表格控件 |
| 3 | QTreeWidget | 树形控件 |
| 4 | QTabWidget | 选项卡 |
| 5 | QDockWidget | 浮动窗口 |
| 6 | QStackedWidget | 堆叠窗口 |
| 7 | QStackedLayout | 堆叠布局 |
| 8 | QGridLayout | 网格布局 |
| 9 | QFormLayout | 表单布局 |
| 10 | QHBoxLayout | 水平布局 |
| 11 | QVBoxLayout | 垂直布局 |
| 12 | QCalendarWidget | 日历 |
| 13 | QThread | 线程 |
| 14 | QTimer | 定时器 |

[CSND\_Blog](https://blog.csdn.net/yurensan/article/details/121056155?ops_request_misc=&request_id=&biz_id=102&utm_term=pyqt5%20dateedit&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-1-121056155.142%5ev86%5einsert_down28v1,239%5ev2%5einsert_chatgpt&spm=1018.2226.3001.4187)

# 布局

在Qt里面布局分为四大类:

* QBoxLayout
* QGridLayout
* QFormLayout
* QStackedLayout

## QBoxLayout

盒子布局,一般用它的两个子类QHBoxLayout和QVBoxLayout, 水平和垂直布局.

## QGridLayout

QGridLayout简介

网格布局QGridLayout 将窗口分割成行和列来进行排列，通常使用方法addWidget()将部件添加到网格布局中，或者使用方法addLayout()将一个布局添加到网格布局中，还可以通过对所添加的控件的行数与列数的跨越，来实现一个部件占据多个单元格。一个常规的QGridLayout布局如下图：

Shape

Description automatically generated

QGridLayout布局示意图

**QGridLayout常用方法1：**

**addWidget(widet , row, col, alignment)**

该方法给给网格布局添加部件，设置指定的行和列，起始位置的默认值为（0,0），其参数含义为：

widget：所添加的控件；

row：控件的行数，默认从0开始；

column：控件的列数，默认从0开始；

alignment：对齐方式.。

**QGridLayout常用方法2：**

**addWidget(widget, fromRow, fromColulmn, rowSpan, columnSpan, alignment)**

当所添加的的控件跨越很多行或者列的时候，使用这个函数，其参数含义为：

widget：所添加的控件;

fromRow：控件的起始行数;

fromColumn：控件的起始列数;

rowSpan：控件跨越的行数;

column：控件跨越的列数;

alignment：对齐方式.

QGridLayout类继承关系：

## QFormLayout

## QStackedLayout

# Spacers

## Horizontal Spacer

## Vertical Spacer

# Buttons

## Push Button

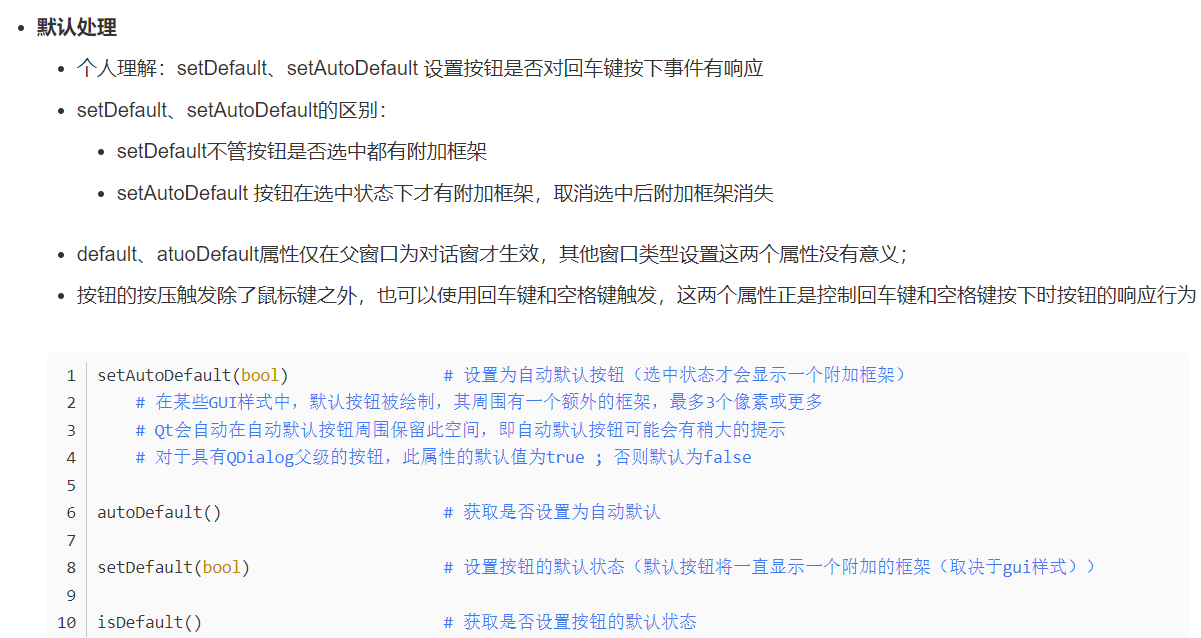


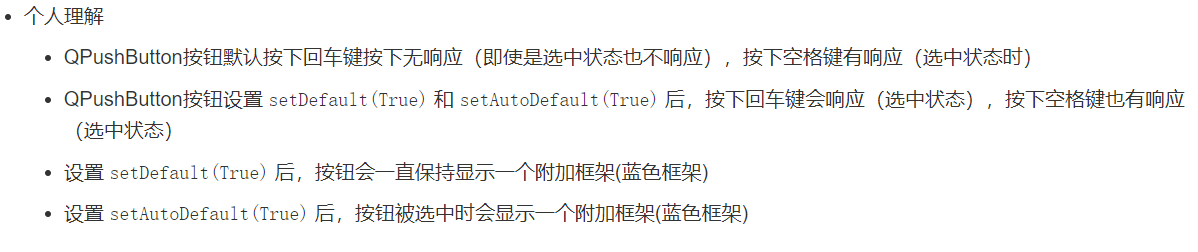
**快捷键设置:**

**当是英文文本时, 字母前面加 “&”, 他的快捷键就是 “Alt+&后面的那个字母”, 比如:**

**“A&BC”,它的快捷键就是 “Alt+B”.**









[(88条消息) PyQt QPushButton应用实例（大小，字体，隐藏边框，添加图片）\_pyqt调整pushbutton大小\_前所未有i的博客-CSDN博客](https://blog.csdn.net/weixin_44919631/article/details/104345019?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522168527347916800192231511%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=168527347916800192231511&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-25-104345019-null-null.142%5ev88%5econtrol_2,239%5ev2%5einsert_chatgpt&utm_term=pyqt5%20%E6%8C%89%E9%94%AE%20%E5%AD%97%E4%BD%93%E8%BE%B9%E6%A1%86%E8%AE%BE%E7%BD%AE&spm=1018.2226.3001.4187)

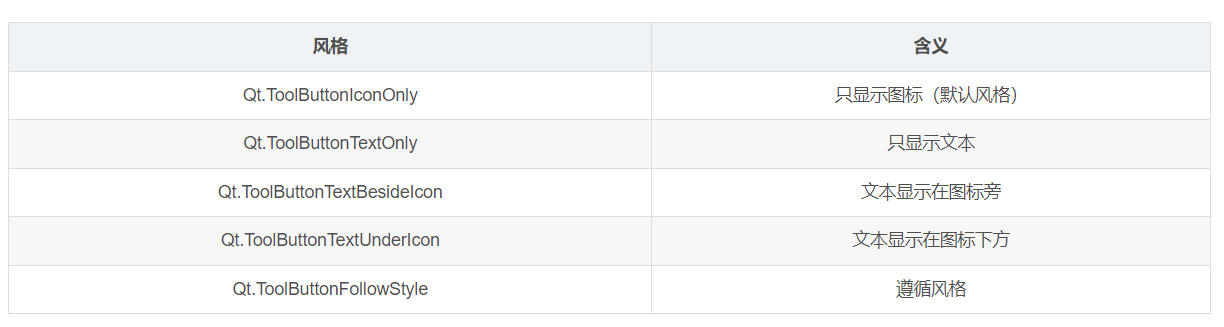
## Tool Button

PyQt5中的ToolButton是一种常用的工具按钮控件，用于在工具栏或菜单栏中添加按钮，方便用户进行操作。使用ToolButton可以实现以下功能：

1. 添加图标和文本：可以在按钮上添加图标和文本，方便用户识别和操作。
2. 设置快捷键：可以为按钮设置快捷键，方便用户使用键盘进行操作。
3. 添加菜单：可以为按钮添加下拉菜单，方便用户进行更多操作。
4. 设置状态：可以设置按钮的状态，如选中状态、禁用状态等。

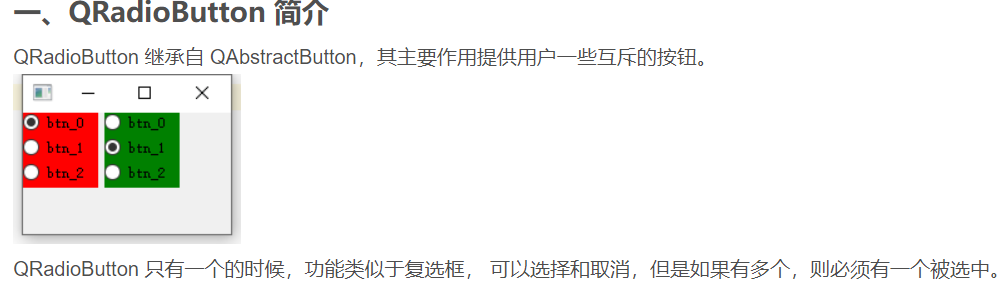
使用ToolButton的步骤如下：

1. 导入PyQt5模块：需要导入PyQt5.QtWidgets模块。
2. 创建ToolButton对象：使用QToolButton()函数创建ToolButton对象。
3. 设置属性：可以设置ToolButton的图标、文本、快捷键、菜单等属性。
4. 添加到工具栏或菜单栏：使用addWidget()函数将ToolButton添加到工具栏或菜单栏中。





## Radio Button



可以通过按键分组的方式实现不同类型的按键不互斥.

QButtonGroup

## Check Box

CheckBox是复选框控件, 其对应与PyQt5中的QCheckBox类, 是一个带有文本标签的开关按钮, 实现通过复选框的选中状态设置用户权限的功能. 复选框通常成组的显示, 但是每个复选框的开关状态相互独立的. 当有一个或多个需要明确的开关状态时, 可以使用复选框.



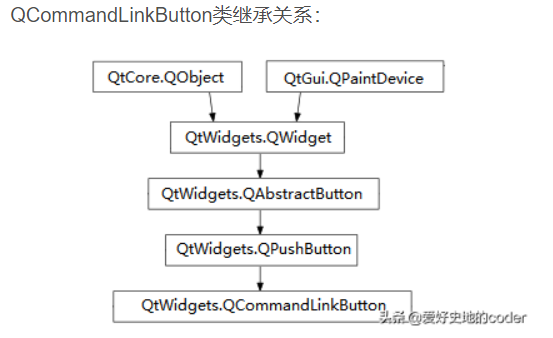
## Command Link Button

QCommandLinkButton部件是一个继承于QPushButton的控件，拥有QPushButton所有的方法和信号。它的用途类似于单选按钮， 常应用于一组互斥选项间进行选择，其外观类似通常的平面按钮，它除了有标题文本外，还允许设置一个描述文本，在默认情况下，它还会带有一个向右的箭头图标，表示按下该控件将打开另一个窗口或者页面。

除了继承自QPushButton的方法和信号外，它还有常用函数：

setDescription()： 设置描述信息

description()： 获得按钮的描述信息





## Dialog Button Box

**QDialogButtonBox介绍**

dialogButtonBox按钮盒是Qt提供的可以快速地布置一组按钮的组件，Qt定义了一系列标准按钮，可以通过在Designer中简单勾选就可以在界面上部署一组按钮。

dialogButtonBox的对应类是QDialogButtonBox，该类是从QWidget类继承的，也是Buttons部件中唯一不从QAbstractButton继承的组件，因为dialogButtonBox是一组按钮的容器，本身并不是按钮。



A picture containing text, font, line, algebra

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**ButtonRole**

在Designer中创建的QDialogButtonBox对应的Button，都有指定的ButtonRole，而我们创建自定义的Button加入到QDialogButtonBox中去时，也需要设定该按钮的ButtonRole。

系统根据你设置的ButtonRole来安排按键的位置。

**ButtonRole含义**

ButtonRole主要有如下三个作用：

同一种ButtonRole的角色，界面风格相同，同一种ButtonRole的按钮在排列时应该依次连续排列

同一种ButtonRole的角色所能发送的信号相同

给自定义Button加入QDialogButtonBox时进行初始化以简单快捷实现。



# Item Views(Item-Based)

## List View

**MVC编程思想**

源数据由模型 (Model) 读取，然后在视图 (View) 组件上显示和编辑，在界面上编辑修改的数据又通过模型保存到源数据。

Model/View 结构将数据模型和用户界面分离开来，分别用不同的实现，是一种显示和编辑数据的有效结构，在处理大型数据时尤其明显。

二者之间数据的交互则是通过委托(delegate)来实现。

**常用Models**

QT提供了一些现成的models用于处理数据项。

QStringListModel 用于存储简单的QString列表。

StandardItemModel 管理复杂的树型结构数据项，每项都可以包含任意数据。

QDirModel 提供本地文件系统中的文件与目录信息。

QSqlQueryModel, QSqlTableModel,QSqlRelationTableModel用来访问数据库。

QListView常用方法:



QListView常用信号:

clicked： 单击某项时，发射此信号；

doubleClicked： 双击击某项时，发射此信号；

activated： 当用户激活index指定的项目时，发射此信号；

entered： 当鼠标光标进入index指定的项目时，发射此信号；

iconSizeChanged： 在视图可见时设置此图标大小时，发射此信号；

indexesMoved： 在视图中移动索引时，发射此信号；

pressed： 按下鼠标按钮时，发射此信号；

viewportEntered： 当鼠标光标进入视图时，发射此信号。

**步骤**

QListView对象

定义model，使用model的setStringList方法或者appendRow(item)给model填数。

QListView.setModel()方法关联二者。

槽函数传递的参数是选中的index。

**Listview和Listwidget的区别:**

ListView 里没有自己的"模型"model，要自己建模来保存数据,这可以很大程度上降低数据冗余,提高程序的效率.但是要求我们对view/model框架比较了解,不适合新手使用.

ListWidget是ListView的子类,在ListWidget中已经帮我们定义好了一个模型.这个模型非常方便,十分全面.这样就可以直接在ListWidget里面添加数据,而不用在从新制定一个模型了.十分方便.

在很多情况下，只需要把一小部分的项呈现给用户，这就没必要使用model这样重量级的组件。PYQt为了方便起见也提供了一些方便的项视图类，分别是QListWidget，QTableWidget和QTreeWidget，使用这些类可以直接对item进行操作。这种实现很像Qt早期版本，组件中包含了相应的item，例如QTableWidget中包含有QTableWidgetItem等。但是对于很大的数据，我们则需要使用PYQt的view类，比如QListView，QTabelView和QTreeView，同时需要提供一个model，可以是自定义model，也可以是PYQt预置的model。例如，如果数据来自数据库，那么你可以使用QTabelView和QSqlTableModel这两个类

原文链接：https://blog.csdn.net/seicany/article/details/9705985

## Tree View

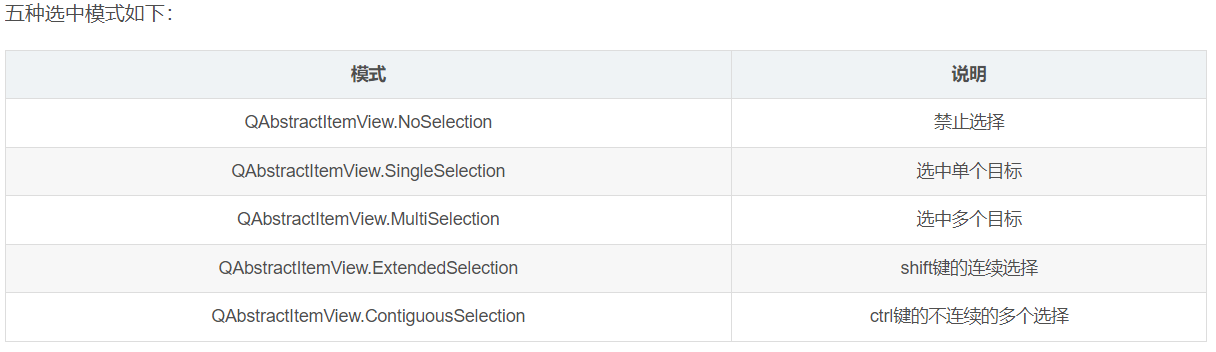
QTreeView实现对模型（Model）中的条目(ITEM)的树形显示。它是模型-视图(Model-View)类之一，是Qt模型-视图框架的组成部分。它实现了由QAbstractItemView类定义的接口，因此能够显示从QAbstractItemModel类派生出的模型所提供的数据。模型-视图体系结构确保树形视图的内容随着模型的更改而更新。其表头由QHeaderView类来创建。

## Table View

QTableView类用于以表格形式输出信息，可通过自定义的数据模型来显示数据，通过setModel来绑定数据源。

设置单元格的选中模式(QAbstractItemView共有五种选中模式)

self.table\_view.setSelectionMode(QAbstractItemView.SingleSelection) # 选中一个单元格



禁止编辑(QAbstractItemView有七种编辑模式)

self.table\_view.setEditTriggers(QAbstractItemView.NoEditTriggers)

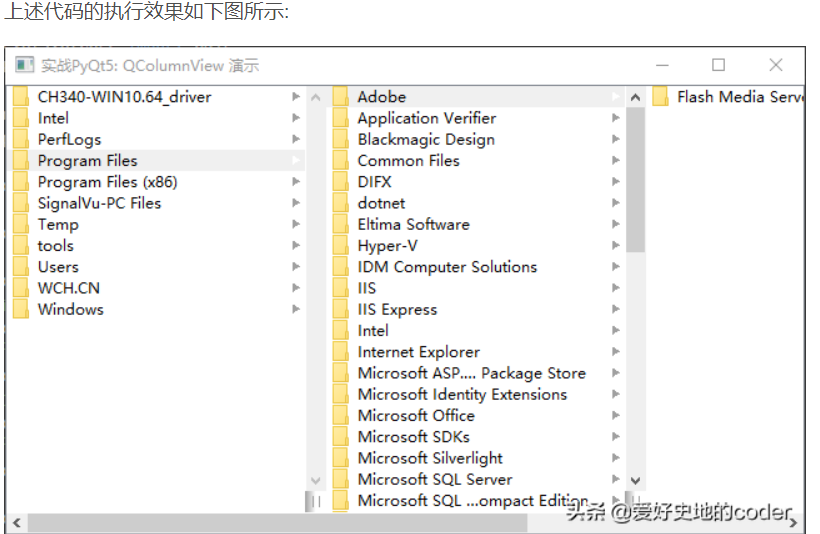
A screenshot of a computer

Description automatically generated with low confidence

## Column View

QColumnView类提供了Model-View框架中的列视图实现。 它在许多QListView中显示一个模型，可以看做是一个级联的列表视图。它是Qt模型-视图框架的组成部分。它实现了由QAbstractItemView类定义的接口，因此能够显示从QAbstractItemModel类派生出的模型所提供的数据。

QColumnView部件是一种类似树形视图的视图，树形视图的层次结构是纵向排列的，父节点在上，子节点在下，同时兄弟节点也是上下排列，而在QColumnView中这种层级关系是通过视图中的列来体现的，兄弟节点是上下排列，但父子节点是横向排列，父节点在左列，子节点在父节点的右边，并通过多列来支持多级父子关系，从展示效果来看，QColumnView部件的展示数据在视图中是一列一列的展示，每列数据就是一列表视图中的数据。



QColumnView常用方法：

* setModel()： 设置视图所关联的模型(Model)；
* setRootIndex()： 设置根索引；
* setColumnWidths()： 设置列宽；
* setSelectionModel()： 设置选择模式。

QColumnView常用信号：

* updatePreviewWidget： 当指定索引的预览部件发生改变时，发射该信号

A picture containing text, screenshot, font, diagram

Description automatically generated

## Undo View

# Item Widgets (Item-Based)

## List Widget

QListWidget类是一个基于条目的接口，用于从列表中添加或删除条目，列表中的每个条目都是一个QListWidgetItem对象，QListWidget可以设置为多重选择

QListWidget类中常用的方法:

| **方法** | **描述** |
| --- | --- |
| addItem() | 在列表中添加QListWidgetItem对象或字符串 |
| addItems() | 添加列表中的每个条目 |
| insertItem() | 在指定地索引处插入条目 |
| clear() | 删除列表的内容 |
| setCurrentItem() | 设置当前所选的条目 |
| sortItems() | 按升序重新排列条目 |

QLIstWidget类中常用的信号:

| **信号** | **含义** |
| --- | --- |
| currentItemChanged | 当列表中的条目发生改变时发射此信号 |
| itemClicked | 当点击列表中的条目时发射此信号 |

## Tree Widget

**TreeWidget树控件**对应 [PyQt5](https://so.csdn.net/so/search?q=PyQt5&spm=1001.2101.3001.7020) 中的**QTreeWidget类**，即，在Qt中，我们可以通过[QTreeWidget](https://so.csdn.net/so/search?q=QTreeWidget&spm=1001.2101.3001.7020)类构造树形结构。QTreeWidget以树的形式显示各个项，每个**树节点（项）使用QTreeWidgetItem**来表示，是界面编程中的一个重要控件。

类的常用方法及说明

QTreeWidget类

| **方法** | **说明** |
| --- | --- |
| addTopLevelItem（） | 添加顶级节点 |
| insertTopLevelItems（） | 在树的顶层索引中插入节点 |
| setColumnCount（） | 设置树结构中的列数 |
| setColumnWidth（） | 设置列的宽度 |

QTreeWidgetItem类

| **方法** | **说明** |
| --- | --- |
| setText（） | 设置节点文本 |
| text（） | 获取节点文本 |
| setIcon（） | 设置节点图标 |
| setHeaderLabels（） | 设置列标题名 |
|  |  |

创建TreeWidget控件

树形结构实现

通过向QTreeWidget添加顶层节点，顶层节点添加子节点，依次这样添加节点的方式构造树形结构。

## Table Widget

QTableWidget是Qt程序中常用的显示数据表格的控件，类似于c#中的DataGrid。QTableWidget是QTableView的子类，它使用标准的数据模型，并且其单元数据是通过QTableWidgetItem对象来实现的，使用QTableWidget时就需要QTableWidgetItem。用来表示表格中的一个单元格，整个表格就是用各个单元格构建起来的.

QTableWidget类中的常用方法



A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing text, screenshot, font, line

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

# Containers

## Group Box

## Scroll Area

问题的提出

在GUI应用中，常常会遇到这样的问题，比如显示一幅大的图像，但显示区域有限，不能完整地显示整个图像，或者能展示界面的区域有限，但是界面中需要展现的控件、内容却很多，在限定的区域内无法全部展示。这个时候我们就需要区域是可以水平和(或)垂直方向可以滚动，以实现对整个图像或者所有控件的浏览。在PyQt中，提供滚动区控件QScrollArea来实现这个功能。

QScrollArea简介

QScrollArea滚动区域控件类提供了关于另一个窗口的滚动的视图，一个滚动区域通常用来显示一个子窗口。如果子窗口中的内容超过了显示窗口的大小，这时候QSrollArea就会自动提供滚动条，通过操作滚动条，用户就可以浏览整个图像或者是操控窗口中的所有控件。

QSrollArea可以给任何QWidget控件添加滚动条。

QScrollArea常用方法有：

setWidget()： 设置控件为QScrollArea的子控件；

takeWidget()：删除QScrollArea的子控件；

widget()： 返回QScrollArea的子控件；

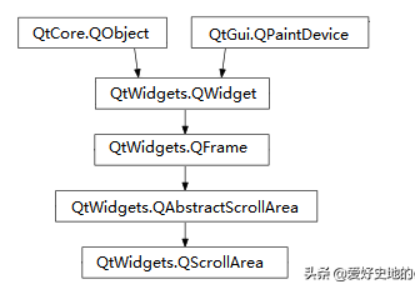
setWidgetResizable()： 设置为true，则滚动区域部件将自动调整，以避免可以不显示的滚动条，或者利用额外的空间；

widgetResizable()： 获得区域里的控件是否自动可调的设置；

ensureVisible()： 确保一定区域可见，必要时滚动；

ensureWidgetVisible()： 确保指定的控件widget可见，必要时滚动；

QScrollArea类继承关系:



## Tool Box

QToolBox控件提供了一个可进行一列选项卡操作方式，在每个选项卡内可包含不同的控件布局。QToolBox将选项卡一个一个地显示，当前项目显示在当前选项卡下方。每个选项卡在选项卡列中都有一个索引位置。一个选项卡的项目是一个QWidget。

QToolBox的每个选项都有一个itemText()，一个可选的itemIcon()，一个可选的itemToolTip()和一个widget()。选项的属性可以通过setItemText()，setItemIcon()和setItemToolTip()来改变。每个项目都可以使用setItemEnabled()单独启用或禁用。

QToolBox常用方法有：

addItem()： 添加一个项目(Item)；

insertItem()： 在指定位置插入一个项目；

removeItem()： 移除一个项目；

count()： 获得总共的项目数；

currentIndex()： 当前操作的项目；

setCurrentIndex()： 设置当前操作的项目；

setItemText()： 设置项目要显示的文本；

setItemIcon()： 设置项目图标；

setItemTooltip()： 设置项目提示信息；

setItemEnabled()： 启用或者禁用项目。Tab Widget

QToolBox常用信号：

* currentChanged： 当前项目发生改变时发出该信号。

QToolBox类继承关系：

A picture containing text, font, screenshot, receipt

Description automatically generated

QToolBox类提供了一列选项卡的小部件（选项卡内含项目）。

工具箱是一个小部件，它将选项卡一个一个的显示，当前项目显示在当前选项卡下方。每个选项卡在选项卡列中都有一个索引位置。一个选项卡的项目是一个QWidget。

每个项目都有一个itemText()，一个可选的itemIcon()，一个可选的itemToolTip()和一个widget()。项目的属性可以通过setItemText()，setItemIcon()和setItemToolTip()来改变。每个项目都可以使用setItemEnabled()单独启用或禁用。

项目使用addItem()添加，或使用insertItem()插入特定位置。项目总数由count()给出。可以删除项目，或使用removeItem()从工具箱中删除项目。通过组合removeItem()和insertItem()，您可以将项目移动到不同的位置。

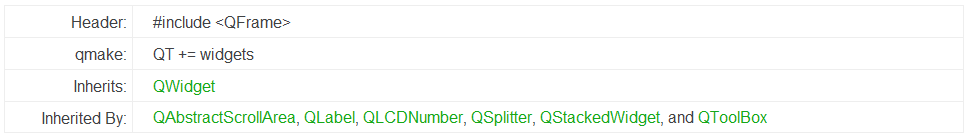
当前项目小部件的索引由currentIndex()返回，并用setCurrentIndex()设置。可以使用indexOf()来找到特定项目的索引，并且item()返回给定索引处的项目。

currentChanged()信号在当前项目改变时发出。

## Stacked Widget

PyQt5 StackedWidget'是一个基于Python编程语言和PyQt5框架的小部件，它可以容纳多个子部件，并且只能显示当前活动的子部件，所有其他子部件都被隐藏。通过单击工具栏、菜单栏等控件上的按钮，用户可以轻松地在当前部件和其他部件之间进行切换。它可以用于构建复杂的用户界面和交互式应用程序，是PyQt5框架的重要部分。

## Frame

继承关系  


A screenshot of a computer

Description automatically generated with medium confidence

**0 控件边框样式（框架+阴影+线宽 组合）**

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## MDI Area

一种同时显示多个窗口的方法是，创建多个独立的窗口，这些独立的窗口被称为SDI（Single Document Interface 单文档界面），每个窗口都有自己的菜单系统，工具栏等，这需要占用很多资源

MDI（Multiple Document Interface 多文档界面）应用程序占用较少的内存资源，子窗口可以放在主窗口的容器中，这个容器控件被称为QMidArea

QMidArea控件通常占据在QMainWindow对象的中央位置，子窗口在这个区域是QMdiSubWindow类的实例，可以设置任何QWidget作为子窗口对象的内部控件，子窗口在MDI区域进行级联排列布局

QMidArea类和QMdiSubWindow类中的常用方法:

| **方法** | **描述** |
| --- | --- |
| addSubWindow() | 将一个小控件添加在MDI区域作为一个新的子窗口 |
| removeSubWindow() | 删除一个子窗口的小控件 |
| setActiveSubWindow() | 激活一个子窗口 |
| cascadeSubWindows() | 安排子窗口在MDI区域级联显示 |
| tileSubWindows() | 安装子窗口在MDI区域平铺显示 |
| closeActiveSubWindow() | 关闭活动的子窗口 |
| subWindowList() | 返回MDI区域的子窗口列表 |
| setWidget() | 设置一个小控件作为QMdiSubWindow实例对象的内部控件 |

## Dock Widget

在 Qt 中，Dock Widget 是一个可停靠的窗口，可以在主窗口的四个边缘停靠，用户可以通过拖动和放置 Dock Widget 到主窗口的边缘来创建和定位布局。Dock Widget 常用于显示工具栏、属性编辑器、文件浏览器等工具。

使用 Dock Widget，我们可以将一些常用的功能视图整合到主窗口中，并且这些视图可以根据需要随时显示或隐藏，方便用户使用。当用户不再需要这些视图时，可以将它们隐藏到窗口的边缘，不会占用主窗口的空间。

Dock Widget 提供了一些有用的功能，如可以指定停靠区域，可以设置 Dock Widget 是否可以关闭、浮动和悬停等，以及在浮动和停靠时可以显示不同的标题。Dock Widget 也可以添加各种 Qt 组件和自定义窗口小部件，使其具有更强的灵活性和定制性。

总之，Dock Widget 是一种非常方便的界面设计工具，可以在主窗口中添加各种视图和工具，提高用户的工作效率。

函数 用法

QDockWidget() 创建一个可悬浮的控件

setWidget(self, QWidget) 向悬浮框中添加控件

setFloating（） 设置默认是否为悬浮状态

addDockWidget(self, Qt\_DockWidgetArea, QDockWidget） 第一个参数为默认停靠的位置

第二个参数为添加的可悬浮的控件

————————————————

A screenshot of a computer

Description automatically generated with medium confidence

## QAxWidget

QAxWidget类是一个包装ActiveX控件的QWidget，它可以组织在一个小部件层次结构和布局中，或者作为一个事件过滤器。

ActiveX控件

ActiveX插件，也叫做OLE控件或OCX控件，它是一些软件组件或对象，可以将其插入到WEB网页或其他应用程序中。

# Input Widget

## Combo Box

## Font Combo Box

QFontComboBox小部件继承自QComboBox，是一个用来选择字体的控件。在该下拉框里填充了按字母顺序排序的字体系列名称表，选中字体时，会发射信号currentFontChanged(font), 根据其输入参数font，就可在槽函数中修改目标对象的字体。

QFontComboBox常用方法：

setCurrentFont()： 设置当前选择的字体；

currentFont()： 获得当前所选择的字体；

setFontFilters()： 设置字体过滤器，参数见QFontComboBox.FontFilter取值列表；

fontFilters()： 获得当前的字体过滤器。

QFontComboBox.FontFilter取值：

QFontComboBox.AllFonts： 显示所有字体

QFontComboBox.ScalableFonts： 显示可缩放字体

QFontComboBox.NonScalableFonts： 显示不可缩放字体

QFontComboBox.MonospacedFonts： 显示等宽字体

QFontComboBox.ProportionalFonts： 显示等比例字体

QFontComboBox常用信号:

currentIndexChanged； 当前索引发生了改变

currentFontChanged； 当前字体发生了改变

QFontComboBox类继承关系：

Diagram

Description automatically generated

## Line Edit

LineEdit是单行文本框，可以从字面意思理解，即：它只能输入单行字符串。

|  |  |
| --- | --- |
| 信号 | 说明 |
| textChanged() | 当更改文本框中的内容时发射该信号 |
| editingFinished() | 当文本框中的内容编辑结束时发射该信号，以按下<Enter>键为编辑结束标志 |

### 1. 简述

QLineEdit即单行文本编辑器，是Qt下常用的人机交互控件，用来接收用户输入的信息。  
QLineEdit并不像表面那么简单，下面介绍如何在QlineEdit下实现密码自动隐藏、背景提示文字，以及控制输入字符的范围。

2. 代码之路

**2.1 密码隐藏**  
密码隐藏就是用户在输入密码等信息时，输入控件的信息自动隐藏，而显示特殊字符来标识显示输入文字的个数。  
密码隐藏有2种实现方式：  
1） 在输入每个字符期间都进行隐藏  
使用QLineEdit的setEchoMode函数，举例如下：

ui->lineEdit\_passwd->setEchoMode(QLineEdit::Password);

2）输入时显示字符，失去焦点时隐藏  
依然使用QLineEdit的setEchoMode函数，参数改变，举例如下：

ui->lineEdit\_passwd->setEchoMode(QLineEdit::PasswordEchoOnEdit);

**2.2 背景文字**  
即用户在输入前显示的提示性文字，实际控件的内容为空，输入字符后提示的文字自动消失，变成用户输入的字符。  
调用QLineEdit的setPlaceholderText函数即可，举例如下:

1. ui->lineEdit\_name->setPlaceholderText(QStringLiteral("用户名为数字、字母或特殊字符"));
3. PyQt5使用：
5. self.lineEdit\_name.setPlaceholderText("请输入以','分隔的变量！例如：a,b")

**2.3 控制输入范围**  
使用正则表达式+过滤器的方法可以控制任意的输入范围。（QRegExp 类+QRegExpValidator类）  
例如，限制用户名为数字、英文和特殊字符组成的字符串，可以用如下代码实现：

1. QRegExp rx("[a-zA-Z0-9!@#%^&\*()\_]+$"); //创建一个正则表达式对象
2. rxQRegExpValidator \*pReg = new QRegExpValidator(rx, this); //创建一个过滤器对象
3. ui->lineEdit\_name->setValidator(pReg); //限制用户名范围
4. ui->lineEdit\_passwd->setValidator(pReg); //限制密码范围

**2.4 其它提高用户体验方法**  
可以用setFocus()函数设置焦点，从而使用户无须选择控件的步骤；  
可以用setStyleSheet()方法改变字体、背景颜色等。

使用QLineEdit一直都是用text()和setText()函数来获得、显示文字。

QLineEdit类中常用的方法如下表

方法 描述

setAlignment() 按固定值方式对齐文本

Qt.AlignLeft：水平方向靠左对齐

Qt.AlignRight:水平方向靠右对齐

Qt.AlignCenter：水平方向居中对齐

Qt.AlignJustify：水平方向调整间距两端对齐

Qt.AlignTop：垂直方向靠上对齐

Qt.AlignBottom：垂直方向靠下对齐

Qt.AlignVCenter：垂直方向居中对齐

setEchoMode() 设置文本框的显示格式，允许输入的文本显示格式的值可以是：

QLineEdit.Normal：正常显示所输入的字符，此为默认选项

QLineEdit.NoEcho：不显示任何输入的字符，常用于密码类型的输入，且长度保密

QLineEdit.Password：显示与平台相关的密码掩饰字符，而不是实际输入的字符

QLineEdit.PasswordEchoOnEdit：在编辑时显示字符，负责显示密码类型的输入

setPlaceholderText() 设置文本框显示文字

setMaxLength() 设置文本框所允许输入的最大字符数

setReadOnly() 设置文本为只读

setText() 设置文本框的内容

text() 返回文本框的内容

setDragEnable() 设置文本框是否接受拖动

selectAll() 全选

setFocus() 得到焦点

setInputMask() 设置掩码

setValidator() 设置文本框的验证器（验证规则），将限制任意可能输入的文本，可用的校验器为

QIntValidator:限制输入整数

QDoubleValidator:限制输入浮点数

QRegexpValidator:检查输入是否符合正则表达式

QLineEdit类中常用信号如下

信号 描述

selectionChanged 只要选择改变了，这个信号就会发射

textChanged 当修改文本内容时，这个信号就会发射

editingFinished 当编辑文本结束时，这个信号就会发射

定义输入掩码的字符

下表列出了输入掩码的占位符和字面字符，并说明其如何控制数据输入

字符 含义

A ASCII字母字符是必须输入的（A-Z，a-z）

a ASCII字母字符是允许输入的，但不是必须输入的

N ASCII字母字符是必须输入的（A-Z，a-z，0-9）

n ASCII字母字符是允许输入的，但不是必须输入的

X 任何字符都是必须输入

x 任何字符都是允许输入的，但不是必须输入的

9 ASCII数字字符是必须输入的（0-9）

0 ASCII数字字符是允许输入的，但不是必须输入的

D ASCII数字字符是必须输入的（1-9）

d ASCII数字字符是允许输入的，但不是必须的（1-9）

# ASCII数字字符与加减字符是允许输入的，但不是必须的

H 十六进制格式字符是必须输入的（A-F，a-f，0-9）

h 十六进制格式字符允许输入，但不是必须的

B 二进制格式字符是必须输入的（0,1）

b 二进制格式字符是允许输入的，但不是必须的

> 所有字母字符都大写

< 所有字母字符都小写

！ 关闭大小写转换

\ 使用‘\’转义上面列出的字符

掩码由掩码字符与分隔符字符串组成，后面可以跟一个分号和空白字符，空白字符在编辑后会从文本删除的

掩码示例如下：

掩码 注意事项

000.000.000.000;\_ ip地址，空白字符是‘\_’

HH:HH:HH:HH:HH:HH; MAC地址

0000-00-00 日期，空白字符是空格

>AAAAA-AAAAA-AAAAA-AAAAA-AAAAA;# 许可证号，空白字符是‘\_’，所有字母都转换为大写

## Text Edit

QTextEdit类是一个多行文本框控件，可以显示多行文本内容，当文本内容超出控件显示范围时，可以显示水平个垂直滚动条，Qtextedit不仅可以用来显示文本还可以用来显示HTML文档

QTextEdit类中常用的方法  
方法    描述  
setPlainText()    设置多行文本框的内容  
toPlainText()    返回多行文本框的文本内容  
setHtml()    设置多行文本框的文本内容为HTML文档，HTML文档是描述网页的  
toHtml()    返回多行文本框的HTML内容  
clear()    清除多行文本框的内容

1. 添加文本,添加HTML格式

# 显示文本

def showText(self):

self.textedit.setPlainText("hello world")

# 显示HTML

def showHTML(self):

self.textedit.setHtml('<font color="blue" size="5">Hello World</font>')Plain Text Edit

注意：这里的添加文本的方式会先将文本框清空再进行添加，若想要追加，则可以使用append方法

self.textedit.append(要追加的字符串格式的内容)

## PlainTextEdit

QPlainTextEdit是一个纯文本编辑器，和QTextEdit基本相似，但是不能编辑富文本也不能编辑表格；

它和QTextEdit 差不多（但是它不是继承QTextEdit），但是它更适合大的文本！ 具体差异看上面！

QTextEdit 是按照像素来滚动的，而QPlainTextEdit 是按照行来滚的，

## Spin Box

### 一、基础知识

1.QSpinBox简介

QSpinBox是计数器控件，允许用户输入整数，或者通过上下按键递增或者递减，默认调整范围是0-99，每次变化步数1，用户可以自行修改范围和步数；

QSpinBox常用方法如下：  


QSpinBox信号：  


## Double Spin Box

可以显示小数

## Time Edit

Text

Description automatically generated

## Date Edit

## Date/Time Edit

## Dial

QDial是一个圆表盘控件, 各种仪表仪盘都可以抽象成一个圆表盘控件，例如汽车仪表盘上的速度计，就是一个圆表盘。

QDial继承自QAbstractSlider,因此也继承了相应的方法和信号，除此以外，它还有一些自身的常用方法。

QDial常用方法：

setNotchesVisible()： 设置是否显示刻度

notchesVisile()： 刻度可见返回True,不可见返回False

setWrapping()： 设置是否回绕

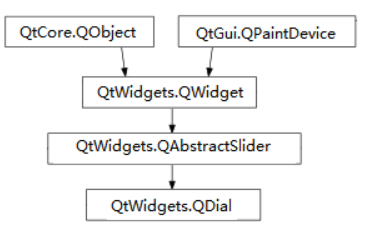
wrapping()： 如果回绕则返回True, 否则返回False

setNotchTarget()： 设置刻度之间的目标刻度

notchTarget()： 返回刻度间的目标宽度

notchSize()： 当前刻度尺寸

QDial类继承关系：



[CSDN\_Blog](https://blog.csdn.net/qq_43950000/article/details/121285775?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522168350338316800184128835%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=168350338316800184128835&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduend~default-1-121285775-null-null.142%5ev86%5einsert_down28v1,239%5ev2%5einsert_chatgpt&utm_term=pyqt5%E4%BB%AA%E8%A1%A8%E7%9B%98&spm=1018.2226.3001.4187)

## Horizontal/Vertical Scroll Bar

QScrollbar滚动条控件和QSlider的行为相似，它们都继承自QAbstractSlider, 都是提供在一定范围内数值变化的交互控件，但它们的外观表现不一样，QSlider用于控制时间变化，比如播放器，有一定时长，用QSlider就非常贴切。QScrollbar常用于空间位置的变化，比如，一幅大的图像，在屏幕上只能显示其中一部分，就可以使用滚动条来控制。

和QSlider类似，QScrollbar提供垂直方向和水平方向的滚动条。

QScrollbar常用方法：

setMinimum()： 设置滚动条的最小值

setMaximum()： 设置滚动条的最大值

setSingleStep()： 设置滚动条的步长

setValue()： 设置滚动条的值

value()： 获得滚动条控件的值

QScrollbar中的常用信号：

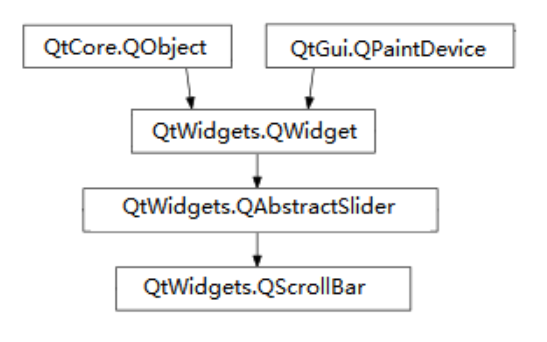
valueChanged： 当滑块的值发生改变时发射此信号，此信号是最常用的

sliderPressed： 当用户按下滑块时发射此信号

sliderMoved： 当用户拖动滑块时发射此信号

sliderReleased： 当用户释放滑块时发射此信号

QScrollbar类继承关系：



## Slider

**QSlider简介**

QSlider小部件提供了一个垂直或水平滑块。

滑块是控制有界值的经典控件。它允许用户沿水平或垂直凹槽移动滑块手柄，并将手柄的位置转换为合法范围内的整数值。

**QSlider常用属性和方法**

slider 提供了两个自然的步长；较大的步长用 pageStep 表示，相当于按下键盘上的 PageDown 和 PageUp；较小的步长用 singleStep 表示，相当于按下键盘上的上、下键。

Left/Down,使滑动条减少一个singleStep

Right/Up,使滑动条增加一个singleStep

PageUp/PageDown,使滑动条增加/减少一个pageStep

Home/End,使滑动条移动到min/max(注意不要搞反)



hasTracking()|获取是否启用了追踪

**QSlider常用信号**

sliderMoved和valueChanged的区别在于,sliderMoved只会在滑块被拖动的情况下激活,在直接点击滑动条时它并不会被激活

A screenshot of a computer

Description automatically generated with low confidence

**QSlider刻度线**

QSlider允许在滑动条旁边画一条刻度线,并设置刻度线的位置和刻度线之间的间距。

**关于刻度线的方法**

间距值的单位不是毫米厘米这些,而是根据min,max得到的单位。

A screenshot of a computer

Description automatically generated with low confidence

**刻度线的位置TickPosition**

A screenshot of a computer

Description automatically generated with low confidence

## QSplitter

QSplitter是一个带切分条（splitter handle）的布局管理器，可以通过setHandleWidth()函数来设置切分条的宽带。QSplitter可以通过setOrientation()函数来指定分割方向，子组件按加载顺序进行指定方向排列。

## StackedWidget

## Key Sequence Edit

一、功能简介

Key Sequence Edit输入部件是用于输入快捷键序列的一个部件，输入快捷键完成后会发射keySequenceChanged（ QKeySequence）和editingFinished（）信号，前者的信号签名就是新输入的快捷键序列，后者则不能提供快捷键，但可以通过部件的keySequence属性获取。

# Display Widgets

## Label:

标签控件主要用于显示用户不能编辑的文本，标识窗体上的对象（例如：给文本框、列表框添加，描述信息等）

## Text Browser:

QTextBrowser类为富文本浏览器提供了超文本导航，通常用作程序日志的输出窗口，默认只读。但只是默认，后期可以自行修改

## Graphics View

用来显示图像

## Calendar Widget

## LCD Number

PyQt5的LCDNumber是一种用于显示数字的小部件。它是一个数字显示器，可用于显示整数或小数，还可以设置数字的精度和显示模式。该小部件在GUI应用程序中常常用于显示实时数据，例如计时器、传感器数据等。

## Progress Bar

进度条

## Horizontal Line

画一条水平线

## Vertical Line

画一条垂直线

## OpenGL Widget

内嵌窗口, 可以画图

## QQuickWidget

# FAQ

## Q: 清空布局:

关键语句:

layout.removeItem(item) 删除layout

layout.widget().deleteLater() #删除widget

示例:

item\_list = list(range(self.displaylayout.count()))

item\_list.reverse() # 倒序删除，避免影响布局顺序

for i in item\_list:

item = self.displaylayout.itemAt(i)

self.displaylayout.removeItem(item) #删除layout

if item.widget():

item.widget().deleteLater() #删除widget

在网上查资料1看到用layout.itemAt(i).widget().deleteLater()来删除layout中的控件，但是这里需要注意几个问题：

1. QLayout的操作都是通过QLayoutItem来实现的，QLayoutItem可能包含widget（如QWidgetItem），也可能不包含widget（如QLayout、QSpacerItem）2，所以在使用itemAt(i).widget().deleteLater()删除前需要进行判断。否则，可能会报错：AttributeError: 'NoneType' object has no attribute 'deleteLater'。

2. 对于所有的QLayoutItem，可以采用layout.removeItem(item)进行删除。

3. 对于包含widget的QLayoutItem，如果只用layout.removeItem(item)，而不用 itemAt(i).widget().deleteLater() 删除对应的控件对象，可能会导致之前创建的widget残留在UI界面上。

4. 遍历layout时，需要按照倒序进行依次删除，否则按正序会改变layout中item的对应序号。