**PyQt5整体介绍**

PyQt5是基于图形程序框架Qt5的Python语言实现，由一组Python模块构成。  
PyQt5的官方网站是：www.riverbankcomputing.co.uk。  
PyQt5模块介绍的官网地址是：<http://pyqt.sourceforge.net/Docs/PyQt5/introduction>.   
  
PyQt5包括的主要模块如下。

* QtCore模块——涵盖了包的核心的非GUI功能，此模块被用于处理程序中涉及的时间、文件、目录、数据类型、文本流、链接、QMimeData、线程或进程等对象。
* QtGui模块——涵盖了多种基本图形功能的类，包括但不限于：窗口集、事件处理、2D图形、基本的图像和界面、字体和文本类。
* QtWidgets模块——包含了一整套UI元素控件，用于建立符合系统风格的Classic界面，非常方便，可以在安装时选择是否使用此功能。
* QtMultimedia模块——包含了一套类库，用于处理多媒体事件，通过调用API接口访问摄像头、语音设备、收发消息（Radio Functionality）等。
* QtBluetooth模块——包含了处理蓝牙活动的类库，其功能包括：扫描设备、连接、交互等行为。
* QtNetwork模块——包含了用于进行网络编程的类库，通过提供便捷的TCP/IP及UDP的C/S代码集合，使得基于Qt的网络编程更容易。
* QtPositioning模块——用于获取位置信息，此模块允许使用多种方式实现定位，包括但不限于：卫星、无线网、文字信息。此模块一般用在网络地图定位系统中。
* Enginio模块——用于构建客户端的应用程序库，在运行时访问Qt Cloud 服务器托管的应用程序。  
  QtWebSockets模块——包含了一组类程序，用于实现WebSocket协议。  
  QtWebKit模块——包含了用于实现基于WebKit2的网络浏览器的类库。
* QtWebKitWidgets模块——提供了一组类库，用于实现一种由Widgets包构建的、基于WebKit1的网络浏览器。
* QtXml模块——包含了用于处理XML的类库，此模块为SAX和DOM API 的实现提供了函数。
* QtSvg模块——通过一组类库，为显示矢量图形文件的内容提供了函数。
* QtSql模块——提供了数据库对象的接口以供使用。
* QtTest模块——包含了通过单元测试，调试PyQt5应用程序的功能。
* QtHelp模块——包含了用于创建和查看可查找的文档的类。
* QtOpenGL模块——使用OpenGL库来渲染3D和2D图形。该模块使得Qt GUI库和OpenGL库无缝集成。
* QtXmlPatterns模块——所包含的类实现了对XML和自定义数据模型的Xquery与XPath的支持。
* QtDesigner模块——所包含的类允许使用PyQt扩展Qt Designer。
* Qt模块——将上面模块中的类综合到一个单一的模块中。这样做的好处是你不用担心哪个模块包含了哪个特定的类；坏处是加载到整个Qt框架中，从而增加了应用程序的内存占用。
* uic模块——所包含的类用来处理.ui文件，该文件由Qt Designer创建，用于描述整个或者部分用户界面。它可以将.ui文件编译为.py文件，以便其他Python程序调用。  
  PyQt5增加了很多模块，可以去官方网站查看，基本上看模块名字就知道大概用处了。PyQt5已经没有phonon模块了，使用QtMultimedia来处理媒体。  
  另外，PyQt5新增的QtWebEngineWidgets模块替代了过时的QtWebKit，但是QtWebKit还在，而新模块更耗内存，具体使用哪个由读者自己决定。

**.2 PyQt 5主要类介绍**

PyQt5 API拥有620多个类和6000个函数。它是一个跨平台的工具包，可以运行在所有主流的操作系统上，包括Windows、Linux和Mac OS。

* QObject类：在类层次结构中是顶部类（Top Class），它是所有PyQt对象的基类。
* QPaintDevice类：所有可绘制的对象的基类。
* QApplication类：用于管理图形用户界面应用程序的控制流和主要设置。它包含主事件循环，对来自窗口系统和其他资源的所有事件进行处理和调度；它也对应用程序的初始化和结束进行处理，并且提供对话管理；还对绝大多数系统范围和应用程序范围的设置进行处理。
* QWidget类：所有用户界面对象的基类。QDialog类和QFrame类继承自QWidget类，这两个类有自己的子类系统（Sub-Class System）。
* QFrame类：有框架的窗口控件的基类。它也被用来直接创建没有任何内容的简单框架，但是通常要用到QHBox或QVBox，因为它们可以自动布置放到框架中的窗口控件。
* QMainWindow类：提供一个有菜单栏、锚接窗口（如工具栏）和状态栏的主应用程序窗口。
* QDialog类：最普通的顶级窗口。如果一个窗口控件没有被嵌入到父窗口控件中，那么该窗口控件就被称为顶级窗口控件。在通常情况下，顶级窗口控件是有框架和标题栏的窗口。在Qt中，QMainWindow和不同的QDialog的子类是最普通的顶级窗口。  
  图A-2至图A-6展示了PyQt 5中重要的类及其继承关系。  
    
    
    
  上面的类图是使用UML建模工具Edraw UML Diagram绘制的。比如在图A-2所示的类图中，每一个方框代表一个PyQt 5的类，在方框的第一行显示的是类名，方框之间通过符号表示类与类之间的继承关系。继承指的是一个类（称为子类）继承另一个类（称为父类）的功能，通过继承可以增加子类的新功能。  
  下面是常用的控件。
* QLabel控件：用来显示文本或图像。
* QLineEdit窗口控件：提供了一个单页面的单行文本编辑器。
* QTextEdit窗口控件：提供了一个单页面的多行文本编辑器。
* QPushButton窗口控件：提供了一个命令按钮。
* QRadioButton控件：提供了一个单选钮和一个文本或像素映射标签。
* QCheckBox窗口控件：提供了一个带文本标签的复选框。
* QspinBox控件：允许用户选择一个值，要么通过按向上/向下键增加/减少当前显示值，要么直接将值输入到输入框中。
* QScrollBar窗口控件：提供了一个水平的或垂直的滚动条。
* QSlider控件：提供了一个垂直的或水平的滑动条。
* QComboBox控件：一个组合按钮，用于弹出列表。
* QMenuBar控件：提供了一个横向菜单栏。
* QStatusBar控件：提供了一个适合呈现状态信息的水平条，通常放在QMainWindow的底部。
* QToolBar控件：提供了一个工具栏，可以包含多个命令按钮，通常放在QMainWindow的顶部。
* QListView控件：可以显示和控制可选的多选列表，可以设置ListMode或IconMode。
* QPixmap控件：可以在绘图设备上显示图像，通常放在QLabel或QPushButton类中。
* Qdialog控件：对话框窗口的基类。  
  QWidget是所有用户界面类的基类，它能接收所有的鼠标、键盘和其他系统窗口事件。没有被嵌入到父窗口中的Widget会被当作一个窗口来调用，当然，它也可以使用setWindowFlags(Qt.WindowFlags)函数来设置窗口的显示效果。QWidget的构造函数可以接收两个参数，其中第一个参数是该窗口的父窗口；第二个参数是该窗口的Flag，也就是Qt.WindowFlags。根据父窗口来决定Widget是嵌入到父窗口中还是被当作一个独立的窗口来调用，根据Flag来设置Widget窗口的一些属性。  
  QMainWindow（主窗口）一般是应用程序的框架，在主窗口中可以添加所需要的Widget，比如添加菜单栏、工具栏、状态栏等。主窗口通常用于提供一个大的中央窗口控件（如文本编辑或者绘制画布）以及周围的菜单栏、工具栏和状态栏。QMainWindow常常被继承，这使得封装中央控件、菜单栏，工具栏以及窗口状态变得更容易，也可以使用Qt Designer来创建主窗口。

**3 QApplication类**

QApplication类用于管理图形用户界面应用程序的控制流和主要设置，可以说QApplication是PyQt的整个后台管理的命脉。任何一个使用PyQt开发的图形用户界面应用程序，都存在一个QApplication对象。  
在PyQt中，可以通过如下代码载入必需的模块，获得QApplication类。  
from PyQt5.QtWidgets import QApplication  
在PyQt的应用程序实例中包含了QApplication类的初始化，通常放在Python脚本的if **name** == "**main**": 语句后面，类似于放在C的main函数里，作为主程序的入口。因为QApplication对象做了很多初始化，所以它必须在创建窗口之前被创建。  
QApplication类还可以处理命令行参数，在QApplication类初始化时，需要引入参数sys.argv。sys.argv是来自命令行的参数列表，Python脚本可以从shell运行，比如用鼠标双击qtSample.py，就启动了一个PyQt应用程序。引入sys.argv后就能让程序从命令行启动，比如在命令行中输入 python qtSample.py，也可以达到同样的效果。  
QApplication类的初始化可以参考以下脚本引用。应用程序整体框架为：  
if **name** == "**main**":  
app = QApplication(sys.argv)  
# 界面生成代码 ...

sys.exit(app.exec\_())

sys.exit()函数可以结束一个应用程序，使应用程序在主循环中退出。  
QApplication采用事件循环机制，当QApplication初始化后，就进入应用程序的主循环（Main Loop），开始进行事件处理，主循环从窗口系统接收事件，并将这些事件分配到应用程序的控件中。当调用sys.exit()函数时，主循环就会结束。  
PyQt 5的应用程序是事件驱动的，比如键盘事件、鼠标事件等。在没有任何事件的情况下，应用程序处于睡眠状态。主循环控制应用程序什么时候进入睡眠状态，什么时候被唤醒。