第1章

PySide6 GUI编程基础

GUI 编程需要使用界面代码和可以使界面控件联动起来的代码，为提高编程效率可使这两部分代码分离，写到整个程序的不同部分。界面代码可以利用 Qt Designer 来设计窗口和对控件进行布局，然后将 Qt Designer 设计的 GUI 编译成 Python 的 py 文件，最后对界面上各控件之间的逻辑关系用窗口或控件提供的 API 方法及 PySide6 提供的信号、槽函数和事件处理函数来完成。窗口界面代码和对窗口逻辑编程的代码可以存储到不同的 py 文件中，以提高开发效率，实现窗口和逻辑的独立编程。

1.1 Python 开发环境搭建

Python 自带了编程环境，但是功能较弱，可以用第三方提供的开发环境进行 Python 程序和 PySide GUI 程序的开发，本书在 PyCharm 环境下编写 Python 程序和 PySide GUI 程序。

1.1.1 Python 和 PySide 简介

1. Python 简介

Python 是一种跨平台高级语言，可以用于 Windows、Linux 和 Mac 平台上。Python 语言简洁明了，即便是非软件专业的初学者也很容易上手。相对于其他编程语言来说，Python 有以下几个优点：

(1) Python 是开源免费的，用户使用 Python 进行开发或者发布自己的程序不需要支付任何费用，也不用担心版权问题，即使作为商业用途，Python 也是免费的。

(2) Python 的语法简单，和传统的 C/C++、Java、C＃等语言相比，Python 对代码格式的要求没有那么严格，这种宽松使得用户在编写代码时比较轻松，不用在细枝末节上花费太多

精力。

(3) Python是高级语言,封装较深,屏蔽了很多底层细节,比如Python会自动管理内存(需要时自动分配,不需要时自动释放)。

(4) Python是解释型语言,可应用于多个平台上,可移植性好。

(5) Python是面向对象的编程语言,可用于高效地开发GUI程序。

(6) Python有广泛的第三方应用程序包,用pip命令就可以安装,扩展性强,可以帮助用户完成各种各样的程序。它覆盖了文件I/O、数值计算、GUI、网络编程、数据库访问、文本操作等绝大部分应用场景。

2. PySide简介

PySide是Qt在Python的绑定,是将C++开发环境下的Qt移植到Python环境下。由于Python语句简单,用Python语言开发Qt应用程序就变得相对容易。下面内容是PySide几个主要模块的简介,其中QtWidgets、QtCore和QtGui是基本模块,开发GUI时都会用这三个模块,其他模块是扩展模块。本书用到的模块有QtWidgets、QtCore、QtGui、QtWebEngineWidgets、QtChart、QtMultimedia、QtSql和QtPrintSupport。

- QtWidgets是窗口模块,提供窗口类和窗口上的各种控件(按钮、菜单、输入框、列表框等)类。
- QtCore是核心模块,是其他模块的应用基础,包括五大模块:元对象系统、属性系统、对象模型、对象树、信号与槽。QtCore模块涵盖了PySide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、MIME、线程或进程等对象。
- QtGui模块涵盖多种基本图形功能的类,包括事件处理、2D图形、基本的图像和字体文本等。
- QtSql模块提供了常用关系型数据库的接口和数据库模型,方便读写数据库中的数据。
- QtMultimedia模块包含处理多媒体事件的类库,通过调用API接口访问摄像头、语音设备,播放音频和视频,录制音频和视频及拍照等。
- QtChart和QtDataVisualization模块用于数据可视化,可以绘制二维和三维数据图表。
- QtPrintSupport模块提供打印支持,能识别系统中安装的打印机并进行打印,可以对打印参数进行设置,提供打印对话框和打印预览对话框。
- QtBluetooth模块包含了处理蓝牙的类库,它的功能包括扫描设备、连接、交互等。
- QtNetwork模块包含用于网络编程的类库,这组类库通过提供便捷的TCP/IP及UDP的c/s程式码集合,使得网络编程更容易。
- QtWebEngine和QtWebEngineWidgets模块借助开源的Chromium浏览器,在应用程序中嵌入Web浏览功能。
- QtXml模块包含了用于处理XML的类库,提供实现SAX和DOM API的方法。
- QtOpenGL、QtOpenGLFunctions和QtOpenGLWidgets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。
- QtDesigner模块可以为Qt Designer创建自定义控件。

- QtSvg 模块为显示矢量图形文件的内容提供了函数。
- QtTest 模块包含了可以通过单元测试调试 PySide 应用程序的功能。
- QtStateMachine 模块可以创建和执行状态图。
- QtHelp 模块可以为应用程序集成在线帮助。
- QtConcurrent 模块支持多线程程序。
- Qt3DCore、Qt3DInput、Qt3DRender、Qt3DAnimation、Qt3DLogic、Qt3DExtras 等模块提供三维渲染、三维实时动画。

1.1.2 Python 开发环境的建立

编写 Python 程序，可以在 Python 自带的交互式界面开发环境中进行。由于其自带的开发环境的提示功能和操作功能不强大，因此可以在第三方提供的专业开发环境中编写 Python 程序，例如 PyCharm，然后调用 Python 的解释器运行程序。本书中 Python 程序的编写既可以在 Python 自带的开发环境中进行，也可以在第三方开发环境中进行，由读者根据自己的爱好自行决定。

1. 安装 Python

Python 是开源免费开发程序，用户可以到 Python 的官网上直接下载 Python 安装程序。登录 Python 的官方网站，可以直接下载不同平台上不同版本的安装程序。Python 的安装文件不大，最新 3.10.2 版只有 27.6MB。单击 Downloads，可以找到不同系统下的各个版本的 Python 安装程序。下载 Python 安装程序时，根据自己的计算机是 32 位还是 64 位选择相应的下载包，例如单击 Windows installer(64-bit)可以下载 64 位的可执行安装程序，一般选择该项即可；单击 Windows embeddable package(64-bit)表示使用 zip 格式的绿色免安装版本，可以直接嵌入(集成)到其他的应用程序中；单击 web-based installer 表示通过网络安装，也就是说下载的是一个空壳，安装过程中还需要联网下载真正的 Python 安装包。Python 安装程序也可以在国内的一些下载网站上找到，例如在搜索引擎中输入“Python 下载”，就可以找到下载链接。

以管理员身份运行 Python 的安装程序 python-3.10.2-amd64.exe，在第 1 步中，如图 1-1 所示，选中 Add Python 3.10 to PATH，单击 Customize installation 项；在第 2 步中，勾选所有项，其中 pip 项专门用于下载第三方 Python 包。单击 Next 按钮进入第 3 步，勾选 Install for all users 项，并设置安装路径，不建议安装到系统盘中，单击 Install 按钮开始安

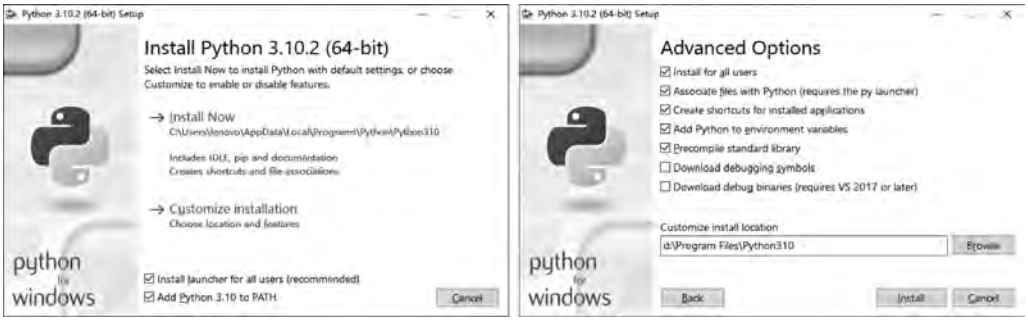

图 1-1 Python 的安装过程

装。安装路径会自动保存到 Windows 的环境变量 PATH 中，Python 可以多个版本共存在一台机器上。安装完成后，在 Python 的安装目录 Scripts 下出现 pip.exe 和 pip3.exe 文件，用于下载其他安装包。

安装完成后，需要测试一下 Python 是否能正常运行。从 Windows 的已安装程序中找到 Python 自己的开发环境 IDLE，如图 1-2 所示，在“>>>”提示下输入“1＋2”或者“print("hello")”并按 Enter 键，如果能返回 3 或者 hello，说明 Python 运行正常。

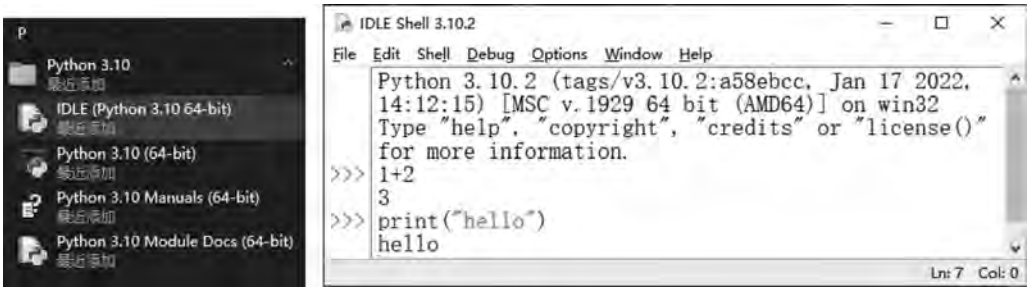

图 1-2　测试 Python

2. 安装 PySide6 及其他包

安装完 Python 后，接下来需要安装本书用到的包 PySide6、openpyxl、pyinstaller、qt-material 和 pymysql，每个包可以单独安装，也可以一次安装多个，下面介绍 Windows 系统中安装 PySide6 的步骤。以管理员身份运行 Windows 的 cmd 命令窗口，输入 pip install pyside6 后按 Enter 键就可以安装 PySide6 包，如图 1-3 所示。也可以用 pip install pyside6 openpyxl pyinstaller pymysql 命令一次安装多个包。如果要卸载包，可以使用 pip uninstall pyside6 命令。

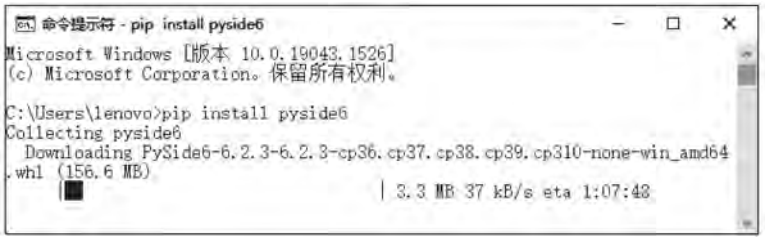

图 1-3　安装 PySide6 包

有些安装包比较大，例如 PySide6 有 156.6MB，如果直接从国外网站上下载 PySide6 可能比较慢，可以使用镜像网站下载，例如清华大学的镜像网站，格式如下所示。

```
pip install pyside6 -i https://pypi.tuna.tsinghua.edu.cn/simple
```

3. 安装 PyCharm

如果只是编写简单的程序，在 Python 自带的开发环境中写代码是可以的。但对于专业的程序员来说，其编写的程序比较复杂，在 Python 自带的开发环境中编写代码就有些捉襟见肘了，尤其是编写面向对象的程序，无论是代码提示功能还是出错信息的提示功能远没

有专业开发环境的功能强大。PyCharm 是一个专门为 Python 打造的集成开发环境(IDLE),带有一整套可以帮助用户在使用 Python 语言开发时提高其效率的工具,比如调试、语法高亮、项目管理、代码跳转、智能提示、自动完成、单元测试、版本控制等。PyCharm 可以直接调用 Python 的解释器运行 Python 程序,极大地提高了 Python 的开发效率。

PyCharm 由 Jetbrains 公司开发,可以在其官网上下载,如图 1-4 所示,PyCharm 有两个版本,分别是 Professional(专业版)和 Community(社区版)。专业版是收费的;社区版是完全免费的,单击 Community 下的 Download 按钮可以下载社区版 PyCharm。在搜索引擎中输入"PyCharm 下载",也可以在其他下载平台找到 PyCharm 下载链接。

图 1-4 PyCharm 下载页面

以管理员身份运行下载的安装程序 pycharm-community-2021.3.2.exe(读者下载的版本可能与此不同),在第 1 个安装对话框中单击 Next 按钮,在第 2 个安装对话框中设置安装路径,如图 1-5 所示。单击 Next 按钮,在第 3 个安装对话框中勾选.py 项,将 py 文件与 PyCharm 关联,单击 Next 按钮,在最后一个安装对话框中单击 Install 按钮开始安装,最后单击 Finish 按钮完成安装。

图 1-5 PyCharm 的安装过程

1.1.3 Python 开发环境使用基础

1. Python 自带集成开发环境

在安装 Python 时,同时也会安装一个集成开发环境 IDLE,它是一个 Python Shell (可

以在打开的 IDLE 窗口的标题栏上看到),在“>>>”提示下逐行输入 Python 程序,每输入一行后按 Enter 键,Python 就执行这一行的内容。前面我们已经应用 IDLE 输出了简单的语句,但在实际开发中,需要编写多行代码时,应在写完代码后一起执行所有的代码,以提高编程效率,为此可以单独创建一个文件保存这些代码,待全部编写完成后一起执行。

在 IDLE 主窗口的菜单栏上选择 File→New File 命令,将打开 Python 的文件窗口,在该窗口中直接编写 Python 代码。在输入一行代码后再按 Enter 键,将自动换到下一行,等待继续输入。单击菜单 File→Save 后,再单击菜单 Run→Run Module 或按 F5 键就可以执行,结果将在 Shell 中显示。文件窗口的 Edit 和 Format 菜单是常用的菜单,Edit 菜单用于编辑查找,Format 菜单用于格式程序,例如使用 Format→Indent Region 可以使选中的代码右缩进。单击菜单 Options→Configure IDLE 可以对 Python 进行设置,例如更改编程代码的字体样式、字体大小、字体颜色、标准缩进长度、快捷键等。

在文件窗口中输入下面一段代码,按 F5 键运行程序,在 Shell 窗口中可以输出一首诗,如图 1-6 所示。

```
#Demo 1_1.py
print(' ' * 20)
print(' ' * 10 + '春晓')
print(' ' * 15 + '----孟浩然')
print('春眠不觉晓,处处闻啼鸟.')
print('夜来风雨声,花落知多少.')
```

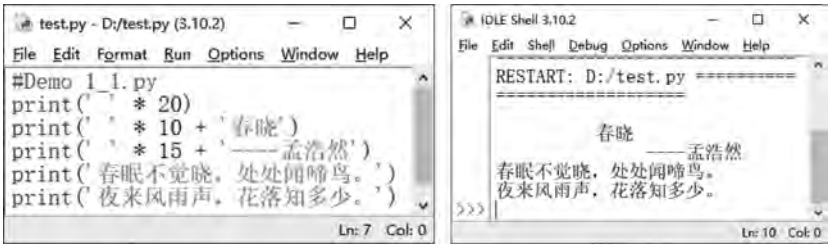

图 1-6 Python 文件窗口和 Shell 窗口

在文件窗口中打开本书实例 Demo1_2.py,见下面的代码,按 F5 键后运行程序,得到一个窗口。对该程序的解释见下一节的内容。

```
import sys #Demo1_2.py
from PySide6 import QtCore, QtGui, QtWidgets

app = QtWidgets.QApplication(sys.argv)
myWindow = QtWidgets.QWidget()
myWindow.setWindowTitle('Demo1_2')
myWindow.resize(500,400)

myButton = QtWidgets.QPushButton(myWindow)
myButton.setGeometry(150,300,150,50)
```

```
myButton.setText('关 闭')
str1_1 = ''*10 + '程序员之歌\n'
str1_2 = ''*15 + '---«江城子»改编\n'
str1_3 = '''
十年生死两茫茫,写程序,到天亮.\n\
千行代码,Bug何处藏.\n\
纵使上线又怎样,朝令改,夕断肠.\n\
领导每天新想法,天天改,日日忙.\n\
相顾无言,惟有泪千行.\n\
每晚灯火阑珊处,程序员,正加班.
'''
peo = str1_1 + str1_2 + str1_3
myLabel = QtWidgets.QLabel(myWindow)
myLabel.setText(peo)
myLabel.setGeometry(50,10,400,300)
font = QtGui.QFont()
font.setPointSize(15)
myLabel.setFont(font)
myButton.setFont(font)
myButton.clicked.connect(myWindow.close)
myWindow.show()
sys.exit(app.exec())
```

2. PyCharm 集成开发环境

要使 PyCharm 成为 Python 的集成开发环境,需要将 Python 设置成 PyCharm 的解释器。启动 PyCharm,如图 1-7 所示,在欢迎对话框中,选择 New Project 项,弹出 New Project 设置对话框,在 Location 中输入项目文件的保存路径,该路径应为空路径,选中 New environment using,并选择 Virtualenv,从 Base interpreter 中选择 Python 的解释器 python.exe,勾选 Inherit global site-packages 和 Make available to all projects,将已经安装的包集成到当前项目中,并将该配置应用于所有的项目。最后单击 Create 按钮,进入 PyCharm 开发环境。

PyCharm 正常启动后,也可以按照下面步骤添加新的 Python 解释器。单击菜单 File→Settings 打开设置对话框,单击左侧项目下的解释器 Python Interpreter,然后单击右边 Python Interpreter 后面的 ⚙ 按钮,选择 Add,弹出添加 Python 解释器的对话框,如图 1-8 所示,左侧选择 System Interpreter,单击右侧 Interpreter 后的 ··· 按钮,弹出选择 Python 解释器的对话框,找到 Python 安装目录下的 python.exe 文件,单击 OK 按钮,回到设置对话框,右边将显示已经安装的第三方程序包。最后单击 OK 按钮关闭所有对话框。

进入 PyCharm 后,单击 File→New 菜单,然后选择 Python File,输入文件名并按 Enter 键后,建立 Python 新文件,输入代码后,要运行程序,需要单击菜单 Run→Run 命令后选择对应的文件,即可调用 Python 解释器运行程序。

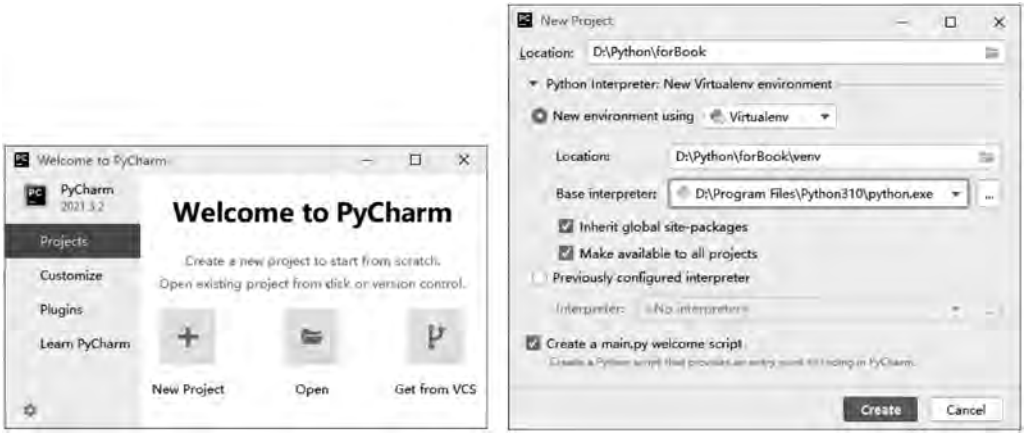

图 1-7　配置 Python 解释器

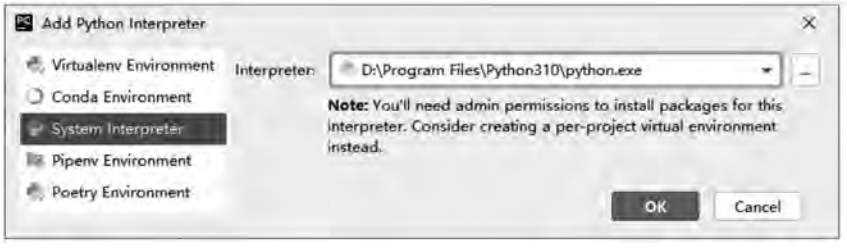

图 1-8　选择 Python 解释器对话框

1.2　PySide6 窗口的运行机理

窗口是图形用户界面(GUI)程序开发的基础,我们平常所见的各种图形界面都是在窗口中放置不同的控件、菜单和工具条,实现不同的动作和目的。图形界面程序开发就是在窗口上放置不同类型的控件、菜单和工具条按钮,并为各个控件、菜单和工具条按钮编写代码使其“活跃”起来。因此要进行图形界面开发,必须首先理解 PySide6 中窗口产生的机理和运行方法,之后再在窗口中添加各种控件。

1.2.1　关于 QWidget 窗口

PySide6 的 QtWidgets 模块集中了可视化编程的各种窗口和控件,这些窗口和控件一般都是直接或间接从 QWidget 类继承来的。继承自 QWidget 类的控件按照功能可以分成如图 1-9 所示的分类,进行可视化编程就要熟悉这些控件的方法、属性、信号及槽函数,以及控件的事件和事件的处理函数。QWidget 是从 QObject 和 QPaintDevice 类继承而来的,QObject 类主要实现信号和槽的功能,QPaintDevice 类主要实现控件绘制的功能。

QWidget 类通常用作独立显示的窗口,这时窗口上部有标题栏,QWidget 类也可以当作普通的容器控件使用,在一个窗口或其他容器中添加 QWidget,再在 QWidget 中添加其

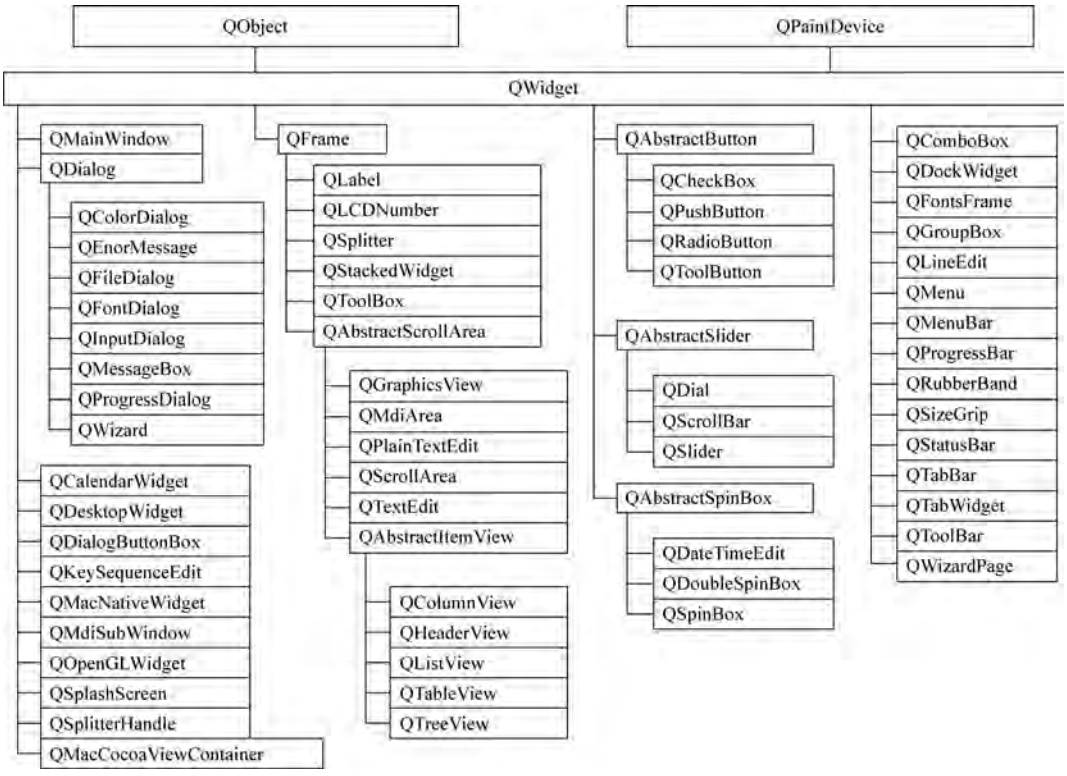

图 1-9 继承自 QWidget 的类

他控件。当一个控件有父窗口时，不显示该控件的标题栏；当控件没有父窗口时，会显示标题栏。常用于独立窗口的类还有 QMainWindow 和 QDialog，它们都是从 QWidget 类继承而来的，关于 QWidget、QMainWindow 和 QDialog 窗口的详细内容参见第 3 章。

1.2.2 QWidget 窗口的初始化类

在创建 QWidget 窗口对象之前，需要先介绍一个 QApplication 类。QApplication 类的继承关系如图 1-10 所示，QApplication 类管理可视化 QWidget 窗口，对 QWidget 窗口的运行进行初始化参数设置，并负责 QWidget 窗口的退出收尾工作，因此在创建 QWidget 窗口对象之前，必须先创建一个 QApplication 类的实例，为后续的窗口运行做好准备。如果不是基于 QWidget 窗口的程序，可以使用 QGuiApplication 类进行初始化，有些程序通过命令行参数执行任务而不是通过 GUI，这时可以使用 QCoreApplication 类进行初始化，以避免初始化占用不必要的资源。

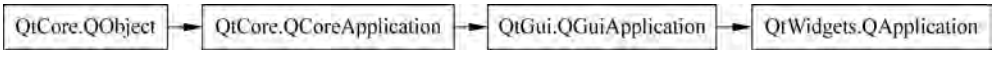

图 1-10 QApplication 类的继承关系

QApplication 类是从 QGuiApplication 类继承来的，QGuiApplication 类为 QWidget 窗口提供会话管理功能，用户退出时可以友好地终止程序，如果终止不了还可以取消对应的进

程,可以保存程序的所有状态用于将来的会话。QGuiApplication 类继承自 QCoreApplication 类,QCoreApplication 类的一个核心功能是提供事件循环(event loop)。这些事件可以来自操作系统,如鼠标、定时器(timer)、网络,以及其他原因产生的事件都可以被收发。通过调用 exec()函数进入事件循环,遇到 quit()函数退出事件循环,退出时发送 aboutToQuit()信号,类似于 Python 的 sys 模块的 exit()方法。当某个控件发出信号时,sendEvent()函数立即处理事件,postEvent()函数把事件放入事件队列以等待后续处理,处于队列中的事件可以通过 removePostedEvent()方法删除,也可通过 sendPostedEvent()方法立即处理事件。

由于 QApplication 类进行可视化界面的初始化工作,因此在任何可视化对象创建之前必须先创建 QApplication 对象,而且还可以通过命令行参数设置一些内部状态。QApplication 类的主要功能有处理命令行参数,设置程序的内部初始状态;处理事件,从窗口接收事件,并通过 sendEvent()和 postEvent()发送给需要的窗口;获取指定位置处的窗口(widgetAt())、顶层窗口列表(topLevelWidgets()),处理窗口关闭(closeAllWindows())等事件;使用桌面对象信息进行初始化,这些设置如调色板(palette)、字体(font)、双击间隔(doubleClickInterval),并跟踪这些对象的变化;定义整个可视化程序界面的外观,外观由 QStyle 对象包装,运行时通过 setStyle()函数进行设置;提供一些非常方便的类,例如屏幕信息类(desktop)和剪贴板类(clipboard);管理鼠标(setOverrideCursor())。

1.2.3 QWidget 窗口的创建

PySide6 的窗口类主要有三种,分别为 QWidget、QMainWindow 和 QDialog,其中 QMainWindow 和 QDialog 从 QWidget 类继承而来。要创建和显示窗口,需要用这 3 个类中的任意一个类实例化对象,并让窗口对象显示并运行起来。窗口类在 PySide6 的 QtWidgets 模块中,使用窗口类之前,需要用"from PySide6.QtWidgets import QWidget, QMainWindow, QDialog"语句把它们导入进来。

下面的代码创建一个空白的 QWidget 窗口,读者需要理解这段代码,这是整个 PySide6 可视化编程最基础的知识。

```
1  import sys                          #Demo1_3.py
2  from PySide6.QtWidgets import QApplication,QWidget
3
4  app = QApplication(sys.argv)#创建应用程序实例对象
5  myWindow = QWidget()                #创建窗口实例对象
6  myWindow.show()                     #显示窗口
7  n = app.exec()                  #执行 exec()方法,进入事件循环,若遇到窗口退出命令,返回整数 n
8  sys.exit(n)                         #通知 Python 系统,结束程序运行
```

- 第 1 行导入系统模块 sys,这个系统模块是指 Python 系统,而不是操作系统。
- 第 2 行导入 QApplication 类和 QWidget 类,PySide6 的类都是以大写字母"Q"开始。
- 第 4 行创建 QApplication 类的实例对象 app,为窗口的创建进行初始化,其中 sys.