# GUI &PyQt

# Работа с ОС через потоки

поток ввода-вывода stdin/stdout

# Библиотеки виджетов

**низкоуровневые**

- openGL
- Vulkan
- cocoa
- Windows API
- …

**высокоуровневые**

| Toolkit name | Windows | OS X | Unix-like | Programming language | License |
|---|---|---|---|---|---|
| AWT | cross-platform | | | Java | |
| CEGUI | Yes | Yes | Yes | C++ | MIT |
| Cocoa | No | Yes | No | Objective-C | Proprietary |
| Elementary | Yes | Yes | Yes | C | LGPL, BSD |
| FLTK | Yes | Yes | Yes | C++ | LGPL |
| Fox toolkit | Yes | No | Yes | C++ | LGPL |
| Fyne | cross-platform | | | Go | BSD |
| GNUstep | Yes | Yes | Yes | Objective-C | LGPL |
| GTK | Yes | Yes | Yes | C | LGPL |
| Kivy | cross-platform | | | Python | MIT |
| LCL | Yes | Yes | Yes | Object Pascal (Free Pascal) | LGPL |
| IUP | Yes | No | Yes | C | MIT |
| Juce | Yes | Yes | Yes | C++ | GPL, proprietary |
| LessTif | No | No | Yes | C | LGPL |
| MFC | Yes | No | No | C++ | Proprietary |
| Nana C++ | Yes | No | Yes | C++ | Boost license |
| OWL (superseded by VCL) | Yes | No | No | C++ (Borland C++) | Proprietary |
| Pivot (WTK) | cross-platform | | | Java | Apache License |
| Qt | Yes | Yes | Yes | C++ | LGPL, proprietary |
| Rogue Wave Views | Yes | No | Yes | C++ | proprietary |
| Shoes (GUI toolkit) | cross-platform | | | Ruby | MIT |
| Swing | cross-platform | | | Java | |
| Tk | Yes | Yes | Yes | C | BSD |
| TnFOX | Yes | Yes | Yes | C++ | LGPL |
| Ultimate++ | Yes | Yes | Yes | C++ | BSD |
| VCL (supersedes OWL) | Yes | No | No | Object Pascal (Delphi) | Proprietary |
| WTL | Yes | No | No | C++ | Microsoft Public License |
| wxWidgets | Yes | Yes | Yes | C++ | WxWindows license |

https://en.wikipedia.org/wiki/List_of_widget_toolkits?oldformat=true

# Эти же библиотеки использует ОС

- Windows GUI
- Apple Aqua
- KDE
- GNOME
- LXDE

*— Здравствуйте, это канал об аниме?*

*— Да.*

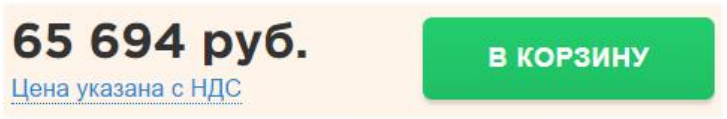
*— Как мне пропатчить KDE2 под FreeBSD?*

# Что такое Qt (cutie)?

- высокоуровневый фреймворк для разработки GUI на C++, работы с БД, интернетом, работы с медиа и тестирования
- разработан Nokia (Qt Software)
- кросс-платформенный настолько, что работал вот на этом
- имеет “обёртки” (как numpy) на Python, Java, Ruby, PHP и др.

# PyQT vs PySide

- PyQt (1998, Riverbank Computing Ltd) – старше, большое сообщество, регулярные обновления




- PySide (2008, Nokia) - наоборот

# Установка

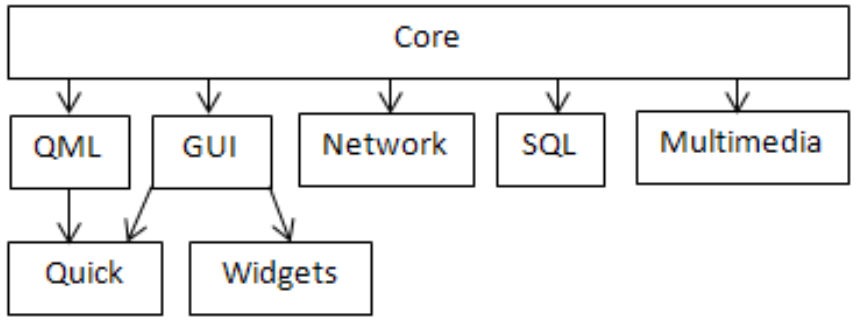
- Qt5 входит в Anaconda
- либо pip install PyQt5

Наиболее популярны Qt4 и Qt5.

Есть разница вплоть до названий и наследований классов и работы браузера.

В 2020 вышла Qt6 с модными функциями.

# Структура




у Qt отличный сайт https://doc.qt.io/qt-5/qtmodules.html

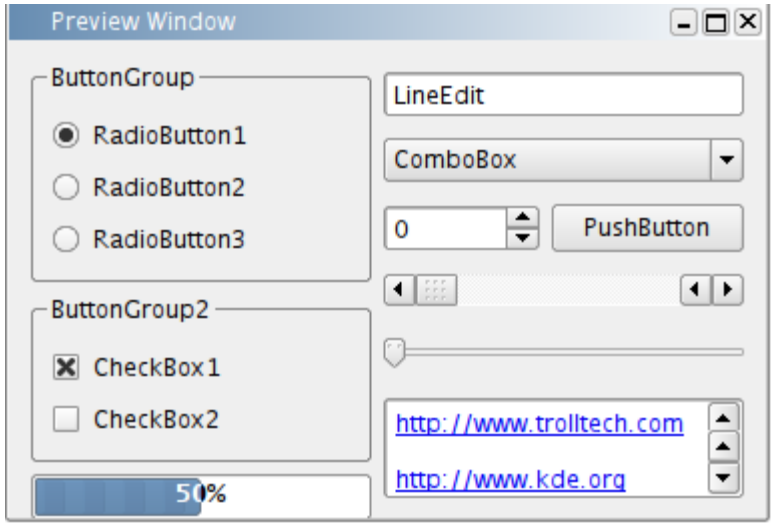
его документация существенно лучше PyQt'шной

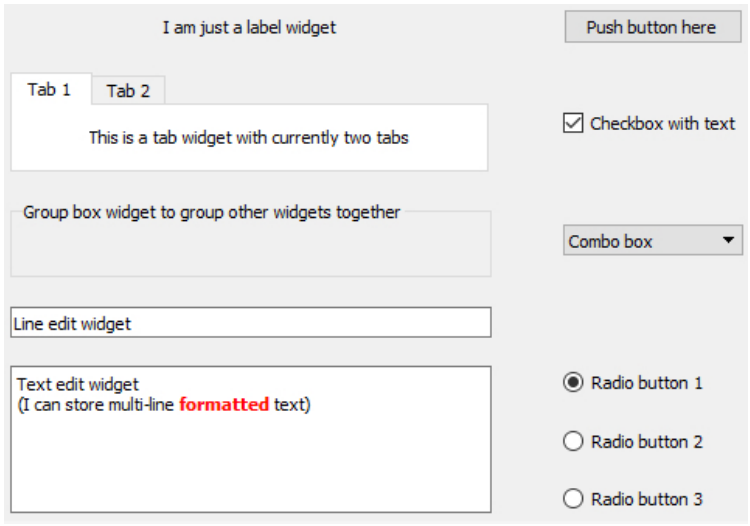
не стесняйтесь залезать и подсматривать там код на C++

Можно начать с этого мануала (оригинал на англ.)
или посмотреть курс от Penn State

# Виджеты (элементы интерфейса)

Qt практически полностью поддерживает общепринятые наименования виджетов




https://www.wikiwand.com/en/Graphical_widget

# Виджеты – что с ними делать?

- Создать виджет
- Прикрепить текущий виджет к другому виджету и наоборот
- Изменить атрибуты виджета (например, отображаемый текст)
- Считать атрибуты виджета (например, считать текст)
- Заполннить виджет прикреплёнными к нему виджетами
- Создать обработчик событий и обработчик сигналов (слоты) для каждого виджета (аналог, привет, асинхронное программирование и async-await)

# QApplication, QWidget

чаще всего будем импортировать:

**import sys**

**from PyQt5.QtWidgets import** QApplication, Qwidget

Все виджеты – это объекты и их наследники.

Основные виджеты расположены в PyQt5.QtWidgets. Его и будем создавать или от него будем наследоваться.

У объекта типа QtWidgets есть куча методов и полей (размер окна, название окна, иконка окна и т.д.)

# QApplication, QWidget

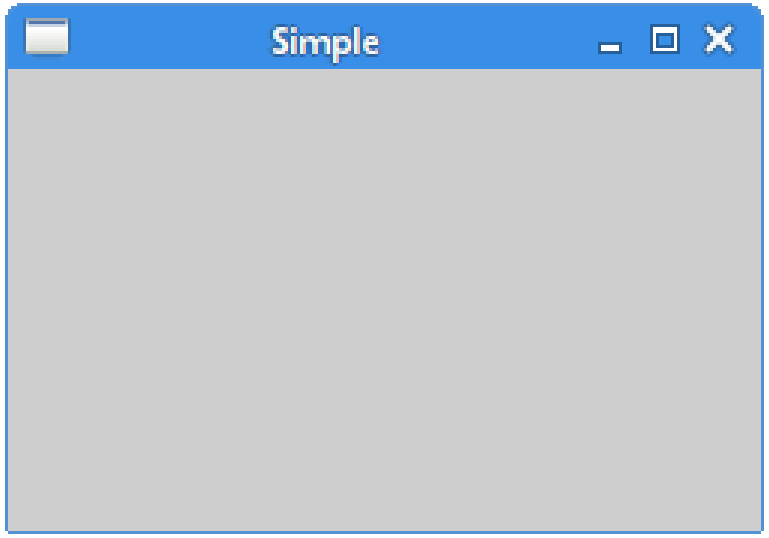
Как и в matplotlib, после после определения всех полей у виджета, нужно будет вывести его на экран с помощью метода show()

Перед тем, как начать выводить виджеты, каждое приложение PyQt5 должно создать объект приложения (объект QApplication).

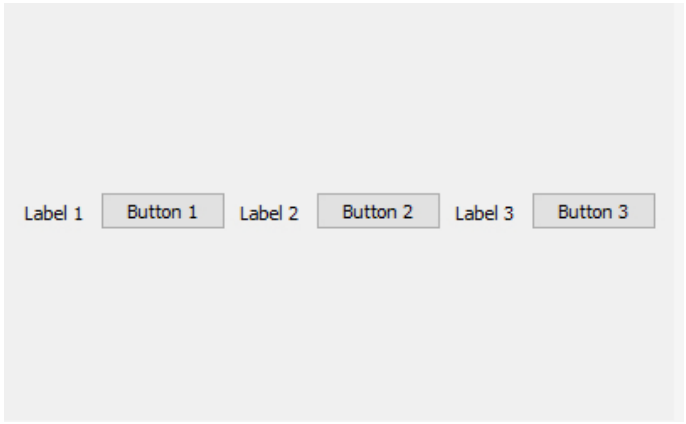
Завершить приложение можно строкой sys.exit(app.exec_()), которая вызовет закрытие приложения системой при нажатии крестика.

# QApplication, Qwidget - пример

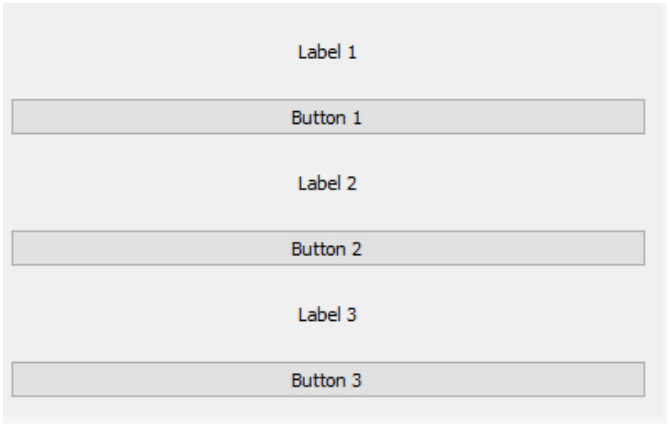
```
import sys
from PyQt5.QtWidgets import QApplication, Qwidget
from PyQt5.QtGui import Qicon
class Example(QWidget):
def __init__(self):
        super().__init__()
        self.initUI()

        def initUI(self):
                self.setGeometry(300, 300, 300, 220)
                self.setWindowTitle('Icon’)
                self.setWindowIcon(QIcon('web.png’))
                self.show()

if __name__ == '__main__’:
        app = QApplication(sys.argv)
        ex = Example()
        sys.exit(app.exec_())
```



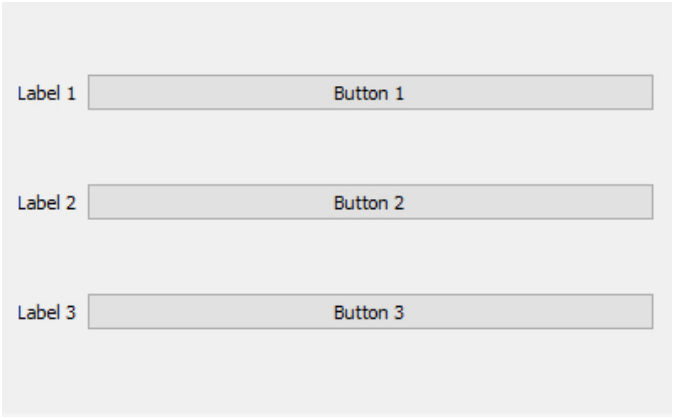
# Простейшее заполнение виджета




**горизонтальное**




**вертикальное**




**массивом**

# События

Все приложения с графическим интерфейсом являются событийно-ориентированными.

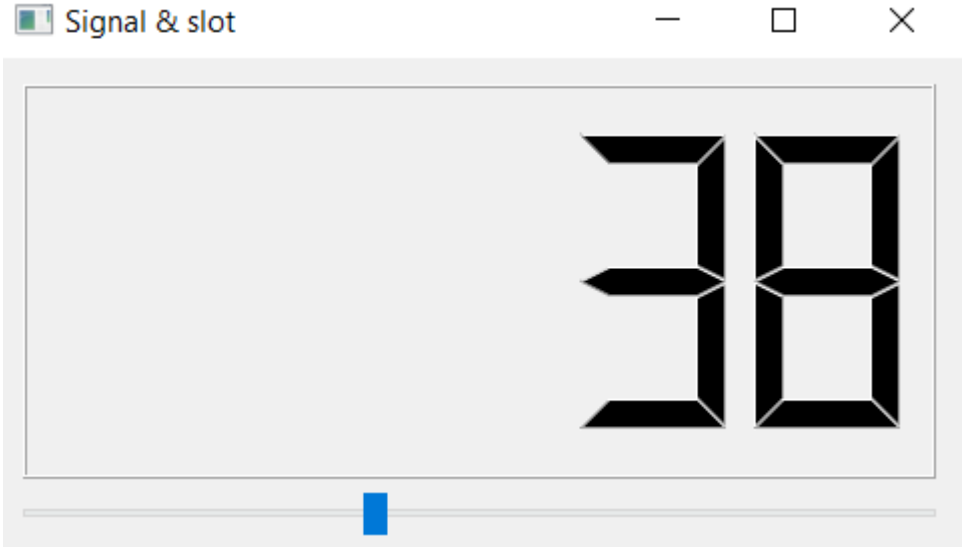
События вызываются главным образом пользователем приложения. Однако, они могут быть вызваны другими средствами, к примеру подключением к Интернету, диспетчером окон или таймером.

Когда мы вызываем метод exec_(), приложение входит в главный цикл. Главный цикл получает события и отправляет их объектам.

# События

- Объект – источник события (кнопка)
- Наблюдатель – обработчик событий, получающий уведомление от источника (кнопка нажата)

Википедия
как это работает в PyQT



# События

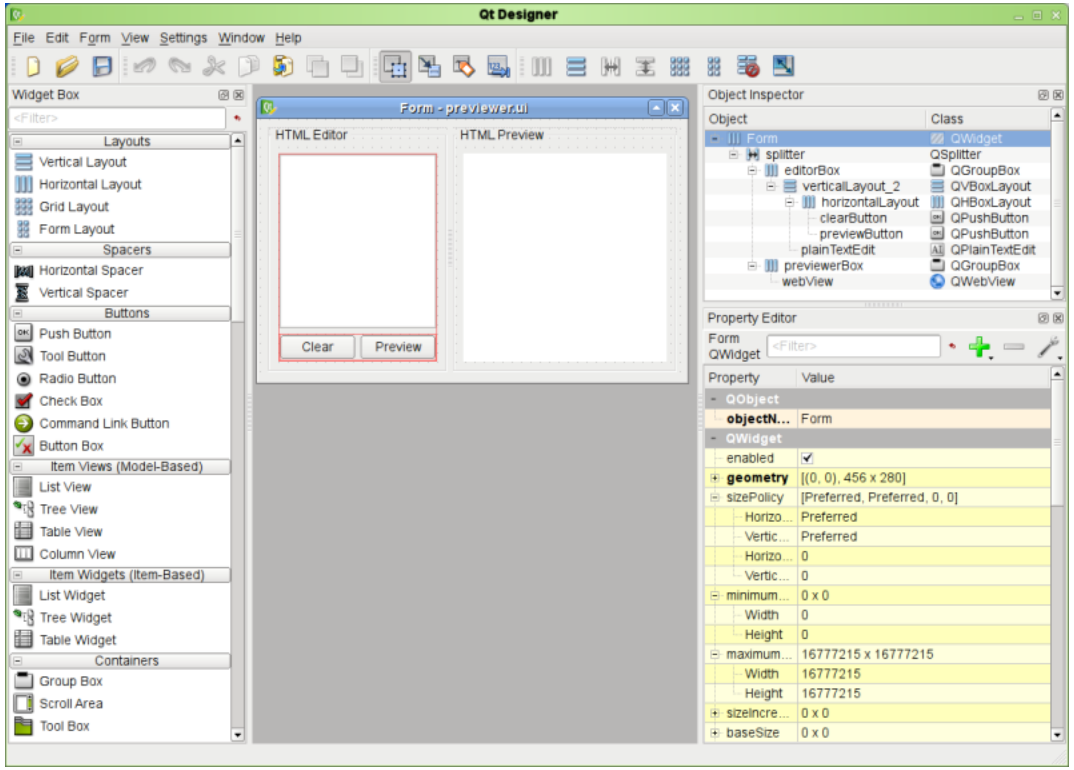
Обработчики событий системные и своим же названием описывают, к чему они относятся (например mousePressEvent).

Чаще всего, мы просто переопределяем обработчик события.

Сигналы – это события (в том числе собственноручно созданные), на которые можно подписывать свои обработчики – слоты.

Сигналы можно создавать для объектов-наследников QObject

# QtDesigner - вёрстка




https://www.cs.usfca.edu/~afedosov/qttut/