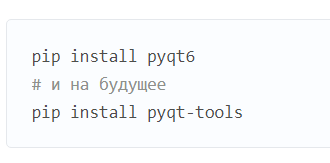
**Тема занятия №31: Программы с интерфейсом (GUI) - tkinter и pyqt6**

**1. Разработка интерфейса на Pyqt6**

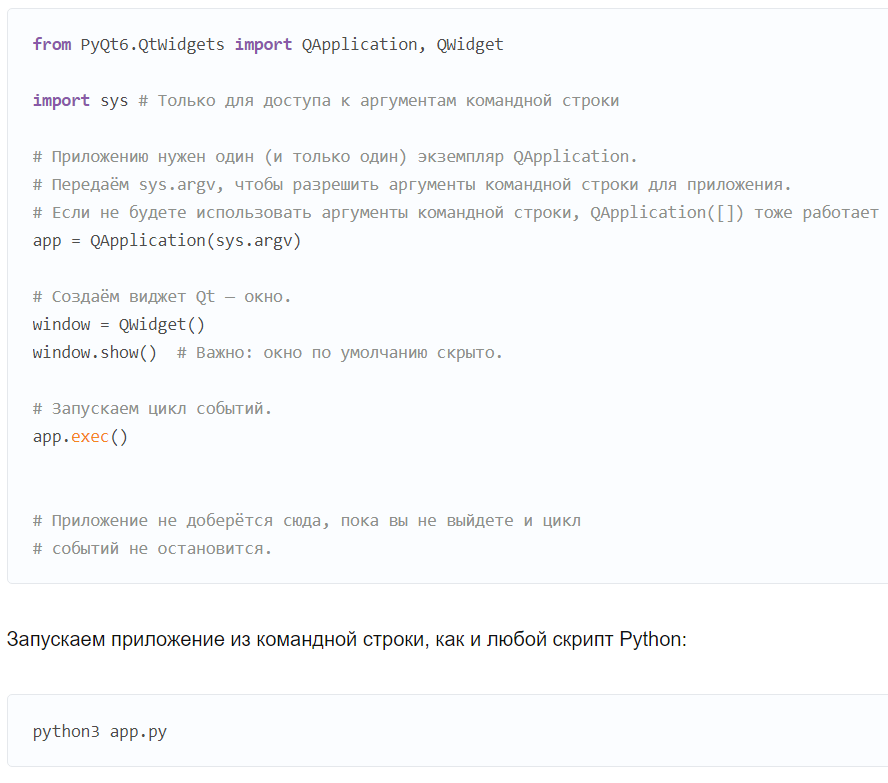
Сначала создадим несколько простых окон на рабочем столе, чтобы убедиться, что PyQt работает, и разберём базовые понятия. Затем кратко изучим цикл событий и то, как он связан с программированием графического интерфейса на Python. В заключение поговорим о QMainWindow с полезными элементами интерфейса, такими как панели инструментов и меню. Подробно я расскажу о них в следующих руководствах.

Создание приложения

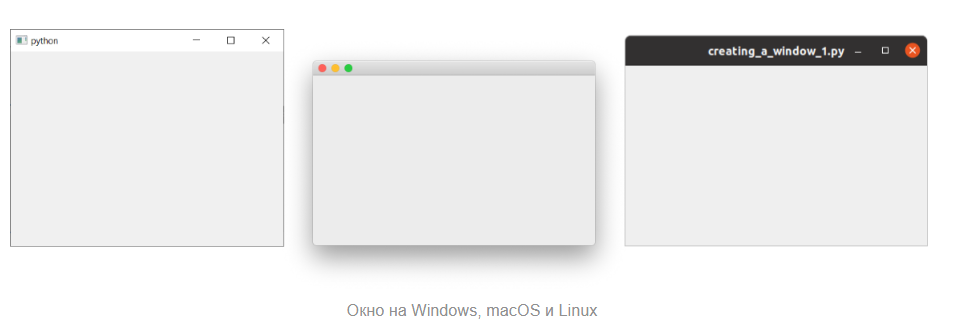
Установка PyQt:



Сначала создадим новый файл Python с любым названием (например app.py) и сохраним его. Исходный код приложения показан ниже. Введите его полностью и постарайтесь не ошибиться. Если что-то напутаете, Python укажет, что именно:



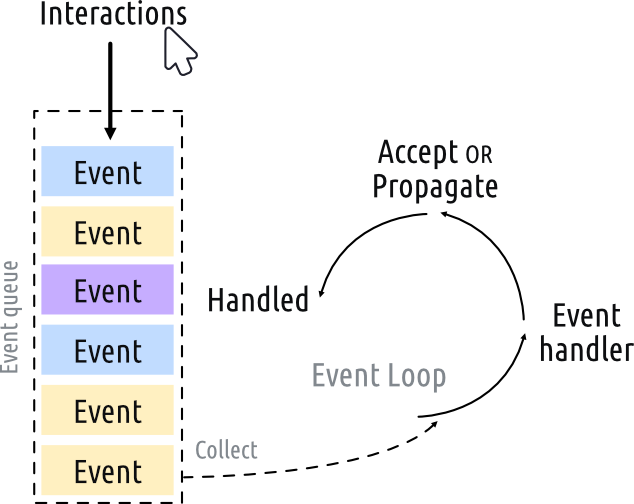
Выполнив его, мы увидим окно. В Qt автоматически создаётся окно с обычным оформлением, возможностью его перетаскивать и менять размер. То, что вы увидите, зависит от платформы, где этот пример выполняется. Вот как отображается это окно на Windows, macOS и Linux (Ubuntu):



Что такое «цикл событий»?

Прежде чем вывести окно на экран, разберём ключевые понятия, касающиеся организации приложений в мире Qt. Если вам уже знакомы циклы событий, можете пропустить эту часть статьи.

Основной элемент всех приложений в Qt — класс QApplication. Для работы каждому приложению нужен один — и только один — объект QApplication, который содержит цикл событий приложения. Это основной цикл, управляющий всем взаимодействием пользователя с графическим интерфейсом:



При каждом взаимодействии с приложением — будь то нажатие клавиши, щелчок или движение мыши — генерируется событие, которое помещается в очередь событий. В цикле событий очередь проверяется на каждой итерации: если найдено ожидающее событие, оно вместе с управлением передаётся определённому обработчику этого события. Последний обрабатывает его, затем возвращает управление в цикл событий и ждёт новых событий. Для каждого приложения выполняется только один цикл событий.

Класс QApplication содержит цикл событий Qt (нужен один экземпляр QApplication). Приложение ждёт в цикле событий новое событие, которое будет сгенерировано при выполнении действия. Всегда выполняется только один цикл событий.

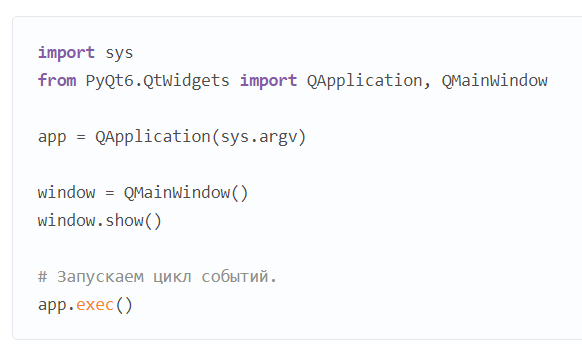
QMainWindow

Итак, в Qt любые виджеты могут быть окнами. Например, если заменить QtWidget на QPushButton. В этом примере получается окно с одной нажимаемой кнопкой:



Классно, но не очень полезно на самом деле: редко когда нужен пользовательский интерфейс, состоящий только из одного элемента управления. Зато возможность с помощью макетов вкладывать одни виджеты в другие позволяет создавать сложные пользовательские интерфейсы внутри пустого QWidget.

В Qt уже есть решение для окна — виджет QMainWindow, имеющий стандартные функции окна для использования в приложениях, который содержит панели инструментов, меню, строку состояния, закрепляемые виджеты и многое другое. Рассмотрим эти расширенные функции позже, а пока добавим в приложение простой, пустой QMainWindow:



Запускаем и видим главное окно. Точно такое же, как и раньше.

QMainWindow пока не очень интересный. Добавим контент. Чтобы сделать настраиваемое окно, лучше создать подкласс QMainWindow, а затем настроить окно в блоке \_\_init\_\_. Так окно станет независимым в плане поведения. Итак, добавляем подкласс QMainWindow — MainWindow:

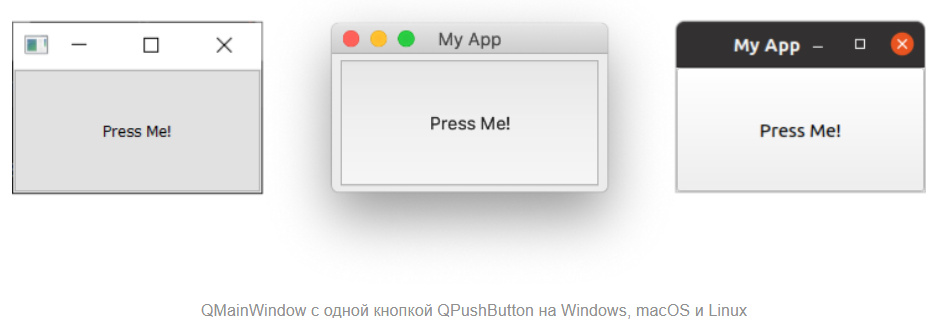


Для этого демо используем QPushButton. Основные виджеты Qt всегда импортируются из пространства имён QtWidgets, как и классы QMainWindow и QApplication. При использовании QMainWindow задействуем .setCentralWidget для размещения виджета (здесь виджет — QPushButton) в QMainWindow, по умолчанию он занимает всё окно. Как добавлять в окна несколько виджетов? Об этом поговорим рассмотрим в руководстве по макетам.

При создании подкласса из класса Qt, чтобы разрешить Qt настраивать объект, всегда нужно вызывать функцию super \_\_init\_\_.

В блоке \_\_init\_\_ сначала используем .setWindowTitle(), чтобы поменять заголовок главного окна. Затем добавляем первый виджет — QPushButton — в середину окна. Это один из основных виджетов Qt. При создании кнопки можно ввести текст, который будет на ней отображаться. Вызываем .setCentralWidget() в окне. Это специальная функция QMainWindow, которая позволяет установить виджет на середину окна.

Запускаем и снова видим окно, но на этот раз с виджетом QPushButton в центре. Нажатие кнопки ничего не даст — с этим мы разберёмся после:



Сигналы QPushButton

Сейчас у нас есть QMainWindow с центральным виджетом QPushButton. Подключим эту кнопку к пользовательскому методу Python. Создадим простой настраиваемый слот the\_button\_was\_clicked, принимающий сигнал clicked от QPushButton:



Запускаем. Если нажать на кнопку, в консоли появится текст Clicked! («Нажата!»)