PyQt5: первые программы | Python 3 для начинающих и чайников

Я начинаю перевод одного замечательного туториала PyQt5 от zetcode.

PyQt5 — это набор Python библиотек для создания графического интерфейса на базе платформы Qt5 от компании Digia.

Он доступен для Python 2.x и 3.x. Этот учебник использует Python 3.

Библиотека Qt является одной из самых мощных библиотек GUI (графического интерфейса пользователя). Официальный сайт PyQt5 — http://www.riverbankcomputing.com/software/pyqt/download5 (где вы можете скачать установочный файл для Windows).

Для linux-систем:

```
sudo apt-get install python3-pyqt5 pyqt5-dev-tools
```

PyQt5 реализован в виде набора python-модулей. Эта библиотека имеет более 620 классов и 6000 функций и методов.

Это мультиплатформенная библиотека, которая работает на всех основных операционных системах, в том числе Unix, Windows и Mac OS.

Простой пример

Это простой пример, показывающий небольшое окно. Тем не менее, мы можем многое сделать с этим окном. Мы можем изменить его размер, развернуть его или свернуть. Это требует написания значительного объёма кода. Однако кто-то уже запрограммировал эту функциональность до нас. Поскольку эта функциональность повторяется в большинстве приложений, нет необходимости писать её снова.

PyQt5 является инструментом высокого уровня. Если бы мы писали это на более низком уровне, следующий пример кода легко мог бы растянуться на сотни строк:

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
```

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget

if __name__ == '__main__':

    app = QApplication(sys.argv)

    w = QWidget()
    w.resize(250, 150)
    w.move(300, 300)
    w.setWindowTitle('Simple')
    w.show()

    sys.exit(app.exec_())
```

Приведенный выше код показывает небольшое окно на экране.

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget
```

Здесь мы делаем необходимые импорты. Основные виджеты расположены в PyQt5.QtWidgets.

```
app = QApplication(sys.argv)
```

Каждое приложение PyQt5 должно создать объект приложения (экземпляр QApplication). Параметр sys.argv это список аргументов командной строки. Скрипты Python можно запускать из командной строки. Это способ, которым мы можем контролировать запуск наших сценариев.

```
w = QWidget()
```

Виджет QWidget это базовый класс для всех объектов интерфейса пользователя в PyQt5. Мы предоставляем конструктор по умолчанию для QWidget. Конструктор по умолчанию не имеет родителя. Виджет без родителей называется окно.

```
w.resize(250, 150)
```

Метод resize() изменяет размеры виджета. Он стал 250 пикселей в ширину и

150 в высоту.

```
w.move(300, 300)
```

Метод move() двигает виджет на экране на координату x=300, y=300.

```
w.setWindowTitle('Simple')
```

Здесь мы задаём заголовок нашего окна.

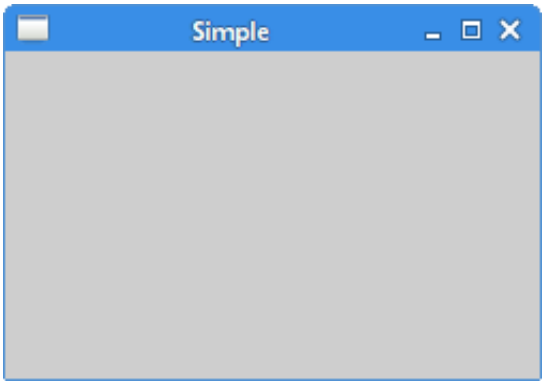
```
w.show()
```

Метод show() отображает виджет на экране. Виджет сначала создаётся в памяти, и только потом (с помощью метода show) показывается на экране.

```
sys.exit(app.exec_())
```

Наконец, мы попадаем в основной цикл приложения. Обработка событий начинается с этой точки. Основной цикл получает события от оконной системы и распределяет их по виджетам приложения. Основной цикл заканчивается, если мы вызываем метод exit() или главный виджет уничтожен. Метод sys.exit() гарантирует чистый выход. Вы будете проинформированы, как завершилось приложение.

Метод exec_() имеет подчеркивание. Это происходит потому, что exec является ключевым словом в python 2.



Значок приложения

Значок приложения — небольшое изображение, которое обычно отображается в верхнем левом углу заголовка. В следующем примере мы покажем, как сделать это в PyQt5. Мы также представим некоторые новые методы.

Не забудьте также скачать какую-нибудь иконку :)

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

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_())
```

Предыдущий пример был написан в процедурном стиле. Язык программирования Python поддерживает как процедурный, так и объектно-ориентированный стили программирования. Программирование в

PyQt5 означает программирование в ООП.

```
class Example(QWidget):

    def __init__(self):
        super().__init__()
        ...
```

Три важные вещи в объектно-ориентированном программировании это классы, данные и методы. Здесь мы создаем новый класс Example. Класс Example наследуется от класса **QWidget**. Это означает, что мы вызываем два конструктора: первый для класса Example и второй для родительского класса. Функция super() возвращает родительский объект Example с классом, и мы вызываем его конструктор.

```
self.initUI()
```

Создание GUI делегируется методу initUI().

```
self.setGeometry(300, 300, 300, 220)
self.setWindowTitle('Icon')
self.setWindowIcon(QIcon('web.png'))
```

Все три метода были унаследованы от класса QWidget.

Метод setGeometry() делает две вещи: помещает окно на экране и устанавливает его размер. Первые два параметра x и y — это позиция окна. Третий — ширина, и четвертый — высота окна. На самом деле, он сочетает в себе методы resize() и move() в одном методе.

Последний метод устанавливает иконку приложения. Чтобы сделать это, мы создали объект QIcon. QIcon получает путь к нашей иконке для отображения.

```
if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

Создаются объекты application и Example. Запускается основной цикл.

Подсказки

Мы можем предоставить всплывающую подсказку для любого из виджетов.

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import sys
from PyQt5.QtWidgets import (QWidget, QToolTip,
    QPushButton, QApplication)
from PyQt5.QtGui import QFont

class Example(QWidget):

    def __init__(self):
        super().__init__()

        self.initUI()

    def initUI(self):

        QToolTip.setFont(QFont('SansSerif', 10))

        self.setToolTip('This is a QWidget widget')

        btn = QPushButton('Button', self)
        btn.setToolTip('This is a QPushButton widget')
        btn.resize(btn.sizeHint())
        btn.move(50, 50)
```

```
        self.setGeometry(300, 300, 300, 200)
        self.setWindowTitle('Tooltips')
        self.show()

if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

В этом примере мы покажем подсказку для двух виджетов PyQt5.

```
QToolTip.setFont(QFont('SansSerif', 10))
```

Этот статический метод устанавливает шрифт, используемый для отображения подсказки. Мы используем шрифт 10px SansSerif.

```
self.setToolTip('This is a QWidget widget')
```

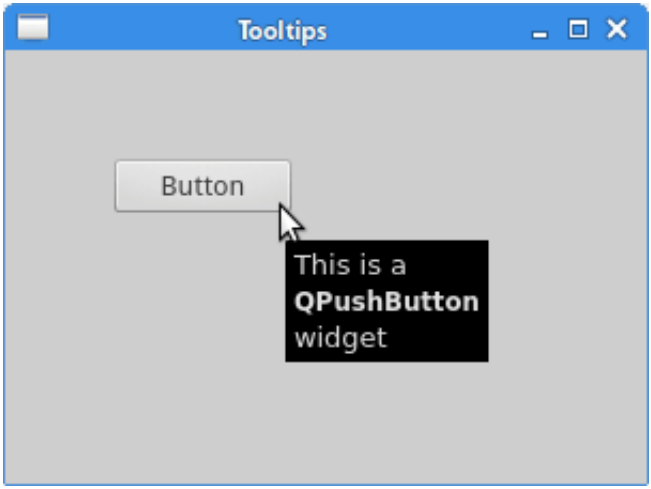
Чтобы создать всплывающую подсказку, мы вызываем метод setToolTip(). Мы можем использовать форматирование текста.

```
btn = QPushButton('Button', self)
btn.setToolTip('This is a QPushButton widget')
```

Мы создаем виджет кнопки и устанавливаем подсказку для него.

```
btn.resize(btn.sizeHint())
btn.move(50, 50)
```

Меняем размер кнопки и перемещаем относительно окна. Метод sizeHint() дает рекомендуемый размер для кнопки.



Закрытие окна

Очевидный способ закрыть окно, это нажать на крестик. В следующем примере мы покажем, как мы можем программно закрыть наше окно. Мы кратко рассмотрим сигналы и слоты.

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import sys
from PyQt5.QtWidgets import QWidget, QPushButton, QApplication
from PyQt5.QtCore import QCoreApplication

class Example(QWidget):

    def __init__(self):
        super().__init__()

        self.initUI()

    def initUI(self):

        qbtn = QPushButton('Quit', self)
        qbtn.clicked.connect(QCoreApplication.instance().quit)
```

```
        qbtn.resize(qbtn.sizeHint())
        qbtn.move(50, 50)

        self.setGeometry(300, 300, 250, 150)
        self.setWindowTitle('Quit button')
        self.show()

if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

В этом примере, мы создаем кнопку выхода. После нажатия на кнопку, приложение завершается.

```
qbtn = QPushButton('Quit', self)
```

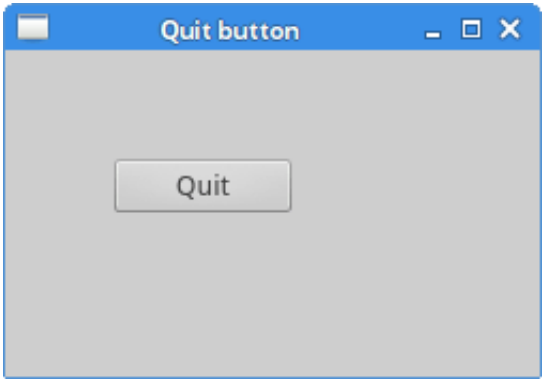
Мы создаем кнопку. Кнопка является экземпляром класса QPushButton. Первый параметр конструктора — название кнопки. Вторым параметром является родительский виджет. Родительский виджет является виджетом Example, который наследуется от QWidget.

```
qbtn.clicked.connect(QCoreApplication.instance().quit)
```

Система обработки событий в PyQt5 построена на механизме сигналов и слотов. Если мы нажмем на кнопку, вызовется сигнал “нажатие”. Слот может быть слот Qt или любая Python функция.

QCoreApplication содержит главный цикл обработки; он обрабатывает и диспетчеризирует все события. Метод instance() дает нам его текущий экземпляр.

Обратите внимание, что QCoreApplication создается с QApplication. Сигнал “нажатие” подключен к методу quit(), который завершает приложение. Коммуникация осуществляется между двумя объектами: отправителя и приемника. Отправитель кнопка, приемник — объект приложения.



Message Box

По умолчанию, если мы нажмем на крестик, QWidget закрывается. Иногда мы хотим изменить это поведение по умолчанию, например, если у нас есть открытый файл, в котором мы сделали некоторые изменения. Мы показываем окно с сообщением для подтверждения действия.

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import sys
from PyQt5.QtWidgets import QWidget, QMessageBox, QApplication

class Example(QWidget):

    def __init__(self):
        super().__init__()

        self.initUI()

    def initUI(self):

        self.setGeometry(300, 300, 250, 150)
        self.setWindowTitle('Message box')
        self.show()

    def closeEvent(self, event):
```

```
        reply = QMessageBox.question(self, 'Message',
            "Are you sure to quit?", QMessageBox.Yes |
            QMessageBox.No, QMessageBox.No)

        if reply == QMessageBox.Yes:
            event.accept()
        else:
            event.ignore()

if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

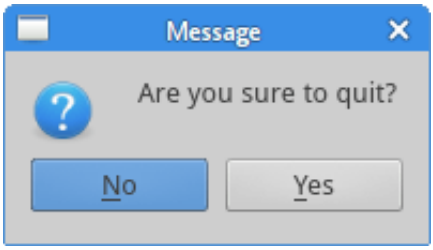
Если мы закрываем QWidget, генерируется QCloseEvent. Чтобы изменить поведение виджета, нам нужно переопределить обработчик события closeEvent().

```
reply = QMessageBox.question(self, 'Message',
    "Are you sure to quit?", QMessageBox.Yes |
    QMessageBox.No, QMessageBox.No)
```

Мы показываем окно с сообщением и с двумя кнопками: Yes и No. Первая строка отображается в заголовке окна. Вторая строка является текстовым сообщением и отображается в диалоговом окне. Третий аргумент определяет комбинацию кнопок, появляющихся в диалоге. Последний параметр — кнопка по умолчанию. Это кнопка, на которой изначально установлен фокус клавиатуры. Возвращаемое значение хранится в переменной reply.

```
if reply == QtGui.QMessageBox.Yes:
    event.accept()
else:
    event.ignore()
```

Здесь мы проверяем возвращаемое значение. Если мы нажмем на кнопку Yes, мы принимаем событие, которое приводит к закрытию виджета и приложения. В противном случае мы будем игнорировать событие закрытия.



Центрирование окна на экране

Следующий скрипт показывает, как мы можем центрировать окно на рабочем столе.

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-

import sys
from PyQt5.QtWidgets import QWidget, QDesktopWidget,
 QApplication

class Example(QWidget):

    def __init__(self):
        super().__init__()

        self.initUI()

    def initUI(self):

        self.resize(250, 150)
        self.center()

        self.setWindowTitle('Center')
        self.show()

    def center(self):

        qr = self.frameGeometry()
        cp = QDesktopWidget().availableGeometry().center()
```

```
        qr.moveCenter(cp)
        self.move(qr.topLeft())

if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

Класс QtWidgets.QDesktopWidget предоставляет информацию о компьютере пользователя, в том числе о размерах экрана.

```
self.center()
```

Код, который будет центрировать окно, находится в нами созданном методе center().

```
qr = self.frameGeometry()
```

Мы получаем прямоугольник, определяющий геометрию главного окна. Это включает в себя любые рамки окна.

```
cp = QDesktopWidget().availableGeometry().center()
```

Мы получаем разрешение экрана нашего монитора. И с этим разрешением, мы получаем центральную точку.

```
qr.moveCenter(cp)
```

Наш прямоугольник уже имеет ширину и высоту. Теперь мы установили центр прямоугольника в центре экрана. Размер прямоугольника не изменяется.

```
self.move(qr.topLeft())
```

Мы двигаем верхний левый угол окна приложения в верхний левый угол прямоугольника qr, таким образом, центрируя окно на нашем экране.

В этой части урока PyQt5 мы рассмотрели некоторые основы.
