**PyQTGraph**

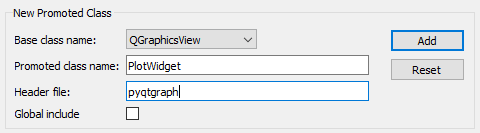
Несмотря на большое количество встроенных виджетов, их не всегда хватает для реализации всех задумок. Хорошая новость заключается в том, что PyQT позволяет создавать собственные виджеты со своими функциями. Некоторые разработчики выкладывают эти библиотеки в Интернет, чтобы их могли использовать другие люди. В этом уроке мы познакомимся с библиотекой [PyQTGraph](http://www.pyqtgraph.org/), которая включает в себя виджеты для работы с графиками.

Для установки библиотеки выполните стандартную команду в командной строке:

pip install pyqtgraph

Затем советуем вам настроить QtDesigner для работы с этой библиотекой. Это — необязательное условие, работать с ней можно и «вручную», как мы делали в самом первом уроке. Но это не столь удобно.

1. Открываем новый виджет в QtDesigner и добавляем **Graphics View**.
2. Нажимаем на него правой кнопкой мыши и выбираем вкладку **Преобразовать в...** (Promote to...).
3. Заполняем поля, как показано на картинке:

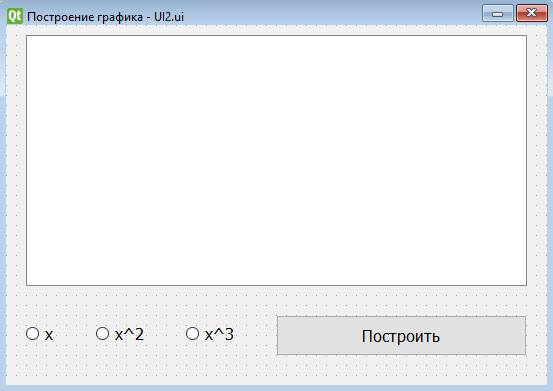


1. Нажимаем на кнопку «Добавить», а затем — на кнопку «Преобразовать».

Если нам в следующий раз понадобится добавить такой же виджет, то мы сможем выбрать его из выпадающего меню **Преобразовать в...**.

Напишем программу, которая будет строить графики одной из трех функций в зависимости от выбранного Radio Button.

Сначала создадим интерфейс в QtDesigner, не забыв при этом выполнить преобразование Graphics View в наш Plot Widget.



Преобразуем его с помощью **pyuic6** и начнем выполнять построение.

import sys

from PyQt6.QtWidgets import QApplication, QMainWindow

from ui import Ui\_MainWindow

class MyWidget(QMainWindow, Ui\_MainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.pushButton.clicked.connect(self.run)

def run(self):

if self.radioButton.isChecked():

self.graphicsView.clear()

self.graphicsView.plot([i for i in range(10)], [i for i in range(10)], pen='r')

elif self.radioButton\_2.isChecked():

self.graphicsView.clear()

self.graphicsView.plot([i for i in range(10)], [i \*\* 2 for i in range(10)], pen='g')

elif self.radioButton\_3.isChecked():

self.graphicsView.clear()

self.graphicsView.plot([i for i in range(10)], [i \*\* 3 for i in range(10)], pen='b')

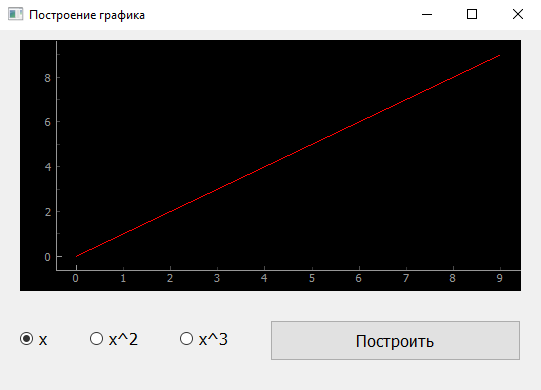
if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

ex = MyWidget()

ex.show()

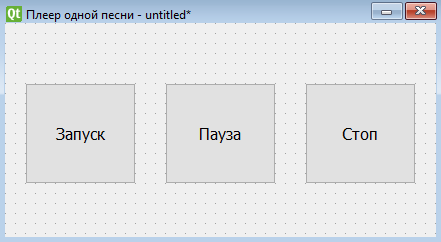
sys.exit(app.exec())



Подробнее про модуль pyqtgraph можно почитать в его официальной [документации](https://pyqtgraph.readthedocs.io/).

**Проигрывание музыки**

Часто в программах хочется добавить какую-либо аудио-информацию. Злоупотреблять этим не следует, но работать со звуками надо уметь. Напишем простейшее приложение-плеер, который будет проигрывать одну песню. Создадим интерфейс нашего приложения в QtDesigner.



Для работы с музыкальными файлами нам понадобится импортировать виджет **QtMultimedia**, а чтобы получить полный путь к файлу — **QtCore**.

from PyQt6 import QtCore, QtMultimedia

Посмотрим, как выглядит наше приложение:

import sys

from ui import Ui\_MainWindow

from PyQt6 import QtCore, QtMultimedia

from PyQt6.QtWidgets import QApplication, QMainWindow

class MyWidget(QMainWindow, Ui\_MainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.load\_mp3('1.mp3')

self.playBtn.clicked.connect(self.\_player.play)

self.pauseBtn.clicked.connect(self.\_player.pause)

self.stopBtn.clicked.connect(self.\_player.stop)

def load\_mp3(self,filename):

media = QtCore.QUrl.fromLocalFile(filename)

self.\_audio\_output = QtMultimedia.QAudioOutput()

self.\_player = QtMultimedia.QMediaPlayer()

self.\_player.setAudioOutput(self.\_audio\_output)

self.\_audio\_output.setVolume(50)

self.\_player.setSource(media)

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

ex = MyWidget()

ex.show()

sys.exit(app.exec())

В нем всего две функции. Начнем с конца, с функции load\_mp3. Она отвечает за загрузку музыкального файла и создание объекта **плеера**, в котором эта музыка будет содержаться.

Важно, что сначала мы передаем в эту функцию не название файла, а путь (если файл в рабочем каталоге приложения) или полный путь, после чего происходит «оборачивание» в формат QUrl. Затем создается медиа-объект, который помещается в плеер.

Во время инициализации, после того, как мы подключили наш интерфейс, мы вызываем эту функцию, и в нашем классе появится поле player. Затем необходимо лишь настроить кнопки, привязав к каждой из них соответствующую функцию нашего объекта. Функции объекта player достаточно очевидны по их названиям.