Съдържание

[1 Въведение 4](#_Toc133492090)

[1.1 Компоненти на PyQt5 4](#_Toc133492091)

[2 Структура на хранилището 5](#_Toc133492092)

[2.1 Файлове с описание 6](#_Toc133492093)

[3 Поддръжка за стари версии на Python 7](#_Toc133492094)

[4 Инсталиране на PyQt5 7](#_Toc133492095)

[4.1 Разбиране на правилната версия за инсталиране 7](#_Toc133492096)

[4.2 Инсталиране от Wheels 8](#_Toc133492097)

[4.2.1 Инсталиране на GPL версия 9](#_Toc133492098)

[4.2.2 Инсталиране на търговската версия 9](#_Toc133492099)

[4.2.3 Изграждане и инсталиране от изходния код 10](#_Toc133492100)

[4.2.4 Изграждане на модула sip 11](#_Toc133492101)

[4.2.5 Изграждане на PyQt5 11](#_Toc133492102)

[5 Поддръжка за сигнали и слотове 11](#_Toc133492103)

[5.1 Необвързани и обвързани сигнали 12](#_Toc133492104)

[5.2 Дефиниране на нови сигнали с pyqtSignal 13](#_Toc133492105)

[5.3 Свързване, прекъсване и излъчване на сигнали 13](#_Toc133492106)

[5.4 Типът PyQt\_PyObject аргумент на сигнала 14](#_Toc133492107)

[5.5 Свързване на слотове по име 15](#_Toc133492108)

[6 Поддръжка за свойства на Qt 15](#_Toc133492109)

[6.1 Дефиниране на нови свойства на Qt 15](#_Toc133492110)

[7 Поддръжка за OpenGL 16](#_Toc133492111)

[8 Поддръжка за Qt интерфейси 17](#_Toc133492112)

[9 Поддръжка за QSettings 17](#_Toc133492113)

[10 Интегриране на Python и QML 17](#_Toc133492114)

[10.1 Регистриране на типове Python 18](#_Toc133492115)

[10.2 Използване на QQmlListProperty 18](#_Toc133492116)

[10.3 Използване на прикачени свойства 19](#_Toc133492117)

[10.4 Използване на източници на стойност на собственост 19](#_Toc133492118)

[10.5 Писане на добавки на Python за qmlscene 19](#_Toc133492119)

[11 Поддръжка за кооперативно мулти-наследяване 20](#_Toc133492120)

[12 Неща, които трябва да знаете 20](#_Toc133492121)

[12.1 TLS поддръжка 20](#_Toc133492122)

[12.2 Аргументи на ключови думи 20](#_Toc133492123)

[12.3 Множествено наследяване 21](#_Toc133492124)

[12.4 Достъп до защитени членски функции 21](#_Toc133492125)

[12.5 None и NULL 21](#_Toc133492126)

[13 Използване на Qt Designer 22](#_Toc133492127)

[13.1 Използване на генерирания код 22](#_Toc133492128)

[13.2 Писане на разширения за Qt Designer. 23](#_Toc133492129)

[14 Системата за ресурси PyQt5 24](#_Toc133492130)

[14.1 pyrcc5 24](#_Toc133492131)

[14.2 QByteArray 24](#_Toc133492132)

[14.3 QColor 25](#_Toc133492133)

[14.4 Предварително дефинирани цветове 26](#_Toc133492134)

[14.5 QDate 27](#_Toc133492135)

[14.6 QDateTime 28](#_Toc133492136)

[14.7 QKeySequence 30](#_Toc133492137)

[14.8 QLine 30](#_Toc133492138)

[14.9 QPoint 31](#_Toc133492139)

[14.10 QPolygon 31](#_Toc133492140)

[14.11 QSize 32](#_Toc133492141)

[14.12 QSizeF 33](#_Toc133492142)

[14.13 QTime 33](#_Toc133492143)

[15 Използване на PyQt5 от Python Shell 34](#_Toc133492144)

[15.1 pylupdate5 35](#_Toc133492145)

[15.2 Разлики между PyQt5 и Qt 35](#_Toc133492146)

[16 API за разширение на PyQt5 37](#_Toc133492147)

# Въведение

“Qt” е набор от C++ библиотеки и инструменти за разработка, който включва независими от платформата абстракции за графични потребителски интерфейси, мрежи, нишки, регулярни изрази, SQL бази данни, SVG, OpenGL, XML, потребителски и приложни настройки, услуги за позициониране и местоположение, комуникации на малък обхват ( NFC и Bluetooth), уеб сърфиране, 3D анимация, диаграми, 3D визуализация на данни и взаимодействие с магазини за приложения. PyQt5 имплементира над 1000 от тези класове като набор от Python модули.

PyQt5 се състои от самия PyQt5 и редица добавки, които съответстват на допълнителните библиотеки на Qt. Всеки се предоставя като дистрибуция на източник и двоични колела за Windows, Linux и macOS.

PyQt5 поддържа платформите Windows, Linux, UNIX, Android, macOS и iOS и изисква Python v3.5 или по-нова версия.

## Компоненти на PyQt5

PyQt5 се състои от редица различни компоненти. На първо място има редица модули за разширение на Python. Всички те са инсталирани в PyQt5 пакета Python и са описани в [списъка с модули](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/module_index.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#ref-module-index) .

PyQt5 също така съдържа редица помощни програми.

* **pyuic5** съответства на помощната програма Qt **uic** . Той преобразува базирани на QtWidgets GUI, създадени с помощта на Qt Designer, в код на Python.
* **pyrcc5** съответства на помощната програма Qt **rcc** . Той вгражда произволни ресурси (напр. икони, изображения, файлове за превод), описани от файл за събиране на ресурси в модул на Python.
* **pylupdate5** съответства на помощната програма **lupdate** на Qt . Той извлича всички преводими низове от кода на Python и създава или актуализира .ts файлове за превод. След това те се използват от Qt Linguist за управление на превода на тези низове.

PyQt5 съдържа спецификационните файлове, които позволяват обвързвания за други библиотеки с класове, базирани на Qt, които допълнително разширяват PyQt5 за разработване и инсталиране.

# Структура на хранилището

Директорията docs/api съдържа структурирания скелет на документацията на API. Той се генерира автоматично от .sip файловете на PyQt5 и се актуализира с всяка нова версия на PyQt5. Те включват информация за всички елементи на API, включително аргументи и типове на метода, но не съдържат описания на тези елементи. Те не трябва да се променят на ръка.

Директорията descriptions съдържа файл за всеки отделен елемент от API на PyQt5 - дори до отделни членове на enum. Освен всичко друго, файлът съдържа reStructuredText, описващ API елемента, и :status:поле, описващо състоянието на описанието. Именно това :status:поле гарантира, че всички направени от потребителя модификации не могат да бъдат презаписани впоследствие. Файловете с описание се създават първоначално, когато нова версия на PyQt5 въвежда нови елементи в API. Тези описателни файлове, които не са били модифицирани, ще бъдат презаписани с всяко ново издание на Qt.

Директорията images съдържа изображенията, които са посочени във файловете с описание. Първоначално те бяха копирани от документацията на Qt и може да бъдат заменени с по-ориентирани към Python алтернативи.

Директорията snippets съдържа кодовите фрагменти, които са посочени във файловете с описание. Първоначално те бяха копирани от документацията на Qt, но с всеки ред от C++ код, превърнат в коментар на Python.

Директорията sphinx съдържа разширение Sphinx и тема, която имплементира системата за документация.

Скриптът sip2rst.py се изпълнява всеки път, когато се прави нова версия на PyQt5. Той актуализира директориите docs/api и descriptions.

Скриптът webxml2rst.py се изпълнява всеки път, когато се прави нова версия на Qt. Той актуализира директориите descriptions, imagesи snippets.

## Файлове с описание

Повечето приноси към документацията ще бъдат корекции към файлове с описание. Файловете с описание за всеки модул се поставят в специфична за модула поддиректория на “описания”. Името на файл с описание се извлича от пълното име на API елемента, който се описва, етикет тип, незадължителен уникален идентификатор и разширение .rst.

Една функция, метод или сигнал може да има няколко версии. Всяка версия се описва в отделен файл. В тези случаи името на всеки файл включва уникален числов идентификатор. Трябва да разгледате полето :realsig: в описанието на файла, за да определите коя от версиите описва файла.

Освен самото описание в reST формат, единствената част от файла на описанието, която трябва да бъде променяна, е полето :status:. Възможните стойности на това поле са описани по-долу.

**Todo**Описанието е извлечено от последното издание на Qt (или е заместващо, ако нищо не е извлечено) и не е било по-късно модифицирано. То ще бъде заменено при следващото издание на Qt.

**Done**

Описанието е променено и няма да бъде презаписано от следващото издание на Qt.

**Review**

Описанието е променено. Самото оригинално описание в Qt документа обаче е актуализирано, след като са направени модификациите. Следователно промените в документацията на Qt трябва да бъдат прегледани, за да се види дали трябва да се направят съответните промени в описанието.

Всички други полета във файла с описание не трябва да се променят.

# Поддръжка за стари версии на Python

Когато дадена версия на Python достигне края на живота си, тя няма да се поддържа в следващото издание на PyQt5. Например, ако текущата версия на PyQt5 е v5.15.x, тогава тя ще стане неподдържана във v5.15.x+1.

*Неподдържан* означава, че тези версии на Python все още могат да работят, но не са тествани.

# Инсталиране на PyQt5

Както GPL, така и търговските версии на PyQt5 могат да бъдат изградени от sdists или инсталирани от двоични колела. Въпреки че този раздел се концентрира върху самия PyQt5, той се отнася еднакво и за проектите за добавки (т.е. PyQtNetworkAuth, PyQtWebEngine, PyQt3D, PyQtChart, PyQtDataVisualization и PyQtPurchasing).

## Разбиране на правилната версия за инсталиране

Хората понякога погрешно вярват, че поддръжката на определена версия на Qt изисква съответстваща версия на PyQt.

Qt използва [семантично управление на версиите](https://translate.google.com/website?sl=en&tl=bg&hl=bg&client=webapp&u=https://semver.org/spec/v2.0.0.html) , когато взема решение за номера на версията на изданието. В обобщение основната версия се увеличава, когато изданието включва несъвместими промени, второстепенната версия се увеличава, когато изданието включва съвместими промени, а версията на корекцията се увеличава, когато изданието не включва видими от потребителя промени.

С PyQt5 номерът на версията на PyQt5 е свързан до известна степен с версията на Qt v5, така че:

* Основната версия винаги ще бъде **5** .
* За конкретна второстепенна версия n ще се изгражда срещу всяка версия на Qt v5, но няма да поддържа никакви нови функции, въведени в Qt v5. n+1 или по-късно.
* Той ще поддържа всички функции на поддържаните модули на Qt v5. n или по-рано.
* Поддръжката за нови модули може да бъде добавена към PyQt5 по всяко време. Това ще доведе до промяна само на версията на корекцията.
* Основната и второстепенната версия на последната версия на PyQt5 ще бъдат същите като последната версия на Qt v5.
* Версиите на корекцията на PyQt5 и Qt v5 са напълно несвързани една с друга.

Така, например, PyQt5 v5.1 ще се изгради срещу Qt v5.2, но няма да поддържа никакви нови функции, въведени в Qt v5.2. PyQt5 v5.1 ще поддържа всички функции на поддържаните модули на Qt v5.0 и тези нови функции, въведени в Qt v5.1.

В обобщение, винаги трябва да се опитвате да използвате най-новата версия на PyQt5, без значение каква версия на Qt v5 използвате

## Инсталиране от Wheels

Wheels са стандартният формат за опаковане на Python за чист Python или модули за двоично разширение като PyQt5. Wheels са предоставени за 32- и 64-битов Windows, 64-битов macOS и 64-битов Linux. Те съответстват на платформите, за които Qt Company предоставя двоични инсталатори.

Wheels се инсталират с помощта на програмата **pip** .

### Инсталиране на GPL версия

Това ще инсталира Wheels за вашата платформа и вашата версия на Python (при положение, че и двете се поддържат). Wheels ще бъде изтеглено автоматично от [PyPI](https://translate.google.com/website?sl=en&tl=bg&hl=bg&client=webapp&u=https://pypi.org/project/PyQt5/) .

Може да откриете, че **pip** не изтегля wheels, а вместо това изтегля sdist и се опитва да изгради PyQt5 от източника. Ако това стане, компилацията вероятно ще се провали със загадъчно съобщение за грешка. Има няколко причини за това:

* вашата версия на Python не се поддържа
* вашата версия на **pip** е твърде стара и не поддържа настоящите стандарти за именуване на Wheels
* Wheels не са осигурени за вашата платформа
* за да може **pip** да изгражда от изходния код, bin директорията на инсталацията на Qt трябва да е включена **PATH**.

**pip** също автоматично ще инсталира всички зависимости, които са необходими. В случая на самия PyQt5 това ще бъдат проектите PyQt5-Qt и PyQt5-sip. Проектът PyQt5-Qt съдържа частите от стандартна инсталация на LGPL Qt, изисквана от PyQt5. Проектът PyQt5-sip съдържа модула [sip](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/sip/sip-module.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp).

Модулът sip в PyQt5 е инструмент за генериране на Python bindings за C++ библиотеки, включително и за Qt. Той предоставя необходимите средства за свързване на C++ и Python и генерира кода за Python bindings, който се използва в PyQt5 за комуникация с Qt.

За да деинсталирате GPL версията, изпълнете:

pip uninstall PyQt5

### Инсталиране на търговската версия

Wheels също са предоставени за търговската версия на PyQt5. Преди да инсталирате изтегленият Wheels с помощта на **pip** , трябва да се уверите, че имате подходящ лиценз за Qt и сте решили как искате да разпространявате своите PyQt5 Wheels на вашите разработчици.

По подразбиране инсталирането на търговското PyQt5 Wheels ще направи същото като инсталирането на GPL Wheels, т.е. автоматично ще изтегли и инсталира необходимите части от стандартна LGPL Qt инсталация от PyPI. Има няколко причини, поради които може да не искате да направите това:

* имате търговски лиценз за Qt и трябва да сте сигурни, че се използва с PyQt5
* не разрешавате на вашите разработчици достъп до PyPI
* искате да сведете до минимум броя Wheels, които трябва да раздадете на вашите разработчици.

Решението на всички тези проблеми е да използвате програмата **pyqt-bundle** , за да свържете копие на вашата собствена Qt инсталация с вашия търговски PyQt5. Това ще създаде ново wheels, което можете лесно да разпространявате на вашите разработчици.

### Изграждане и инсталиране от изходния код

PyQt5 е изграден с помощта на [PyQt-builder](https://translate.google.com/website?sl=en&tl=bg&hl=bg&client=webapp&u=https://pypi.org/project/PyQt-builder/) . За да го инсталирате, стартирайте:

pip install PyQt-builder

PyQt-builder е разширение към генератора [на SIP](https://translate.google.com/website?sl=en&tl=bg&hl=bg&client=webapp&u=https://pypi.org/project/sip/) свързвания, което ще се инсталира автоматично.

Останалата част от тези инструкции предполага, че сте изтеглили PyQt5 sdist от [PyPI и ще използвате инструмента за команден ред](https://translate.google.com/website?sl=en&tl=bg&hl=bg&client=webapp&u=https://pypi.org/project/PyQt5/)**sip-install** на SIP, за да направите изграждането и инсталирането.

PyQt-builder разширява системата за изграждане чрез добавяне на [опции към](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt-builder/command_line_tools.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp)[инструментите на командния ред](https://www-riverbankcomputing-com.translate.goog/static/Docs/sip/command_line_tools.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) на SIP .

### Изграждане на модула sip

Не е необходимо да инсталирате модула [PyQt5.sip](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/sip/sip-module.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) преди изграждането на PyQt5, но той трябва да бъде инсталиран, преди да може да се използва PyQt5.

Модулът е изграден с помощта **setuptools** и е достъпен от проекта [PyQt5-sip](https://translate.google.com/website?sl=en&tl=bg&hl=bg&client=webapp&u=https://pypi.org/project/PyQt5-sip/) в PyPI. Той използва **setuptools** като своя система за изграждане и може да бъде инсталиран чрез **pip** или можете също да разопаковате sdist и да го инсталирате, като изпълните неговия скрипт **setup.py.**

### Изграждане на PyQt5

След като сте изтеглили sdist от PyPI, разархивирайте я и променете директорията до нейната основна директория (т.е. тази, която съдържа pyproject.toml файл).

Ако създавате комерсиалната версия на PyQt5, копирайте лицензния файл pyqt-commercial.sip в директорията sip.

Изграждане на допълнителни проекти за PyQt5

Допълнителните проекти на PyQt5 (т.е. PyQtNetworkAuth, PyQtWebEngine, PyQt3D, PyQtChart, PyQtDataVisualization и PyQtPurchasing) са изградени и инсталирани по абсолютно същия начин като самия PyQt5. Първо трябва да се изгради и инсталира PyQt5.

# Поддръжка за сигнали и слотове

Една от ключовите характеристики на Qt е използването на сигнали и слотове за комуникация между обекти. Използването им насърчава разработването на компоненти за многократна употреба.

Един сигнал се излъчва, когато се случи нещо, което може да е от интерес. Слот е Python callable функция. Ако сигналът е свързан със слот, тогава слотът се извиква, когато сигналът се излъчва. Ако сигналът не е свързан, тогава нищо не се случва. Кодът (или компонентът), който излъчва сигнала, не знае и не му пука дали сигналът се използва.

Механизмът за сигнал/слот има следните характеристики.

* Един сигнал може да бъде свързан към много слотове.
* Един сигнал може също да бъде свързан с друг сигнал.
* Аргументите на сигнала могат да бъдат всеки тип Python.
* Един слот може да бъде свързан към много сигнали.
* Връзките могат да бъдат директни (т.е. синхронни) или на опашка (т.е. асинхронни).
* Връзки могат да се правят през нишки.
* Сигналите може да са прекъснати.

## Необвързани и обвързани сигнали

Сигнал (по-специално необвързан сигнал) е атрибут на класа. Когато даден сигнал е посочен като атрибут на екземпляр от класа, тогава PyQt5 автоматично свързва екземпляра със сигнала, за да създаде обвързан сигнал . Това е същият механизъм, който самият Python използва за създаване на обвързани методи от функции на класа.

Обвързаният сигнал има connect() и методи disconnect(), emit() които изпълняват свързаната функционалност. Освен това има signal атрибут, който е сигнатурата на сигнала, който ще бъде върнат от SIGNAL() макроса на Qt.

Един сигнал може да бъде претоварен, т.е. един сигнал с определено име може да поддържа повече от една сигнатура. Сигналът може да бъде индексиран със сигнатура, за да се избере желаната. Сигнатурата е последователност от типове. Типът е или обект от тип Python, или низ, който е името на тип C++. Името на тип C++ се нормализира автоматично, така че, например, да може да се използва QVariant вместо не нормализирания const QVariant &.

Ако даден сигнал е претоварен, той ще има стойност по подразбиране, която ще се

## Дефиниране на нови сигнали с pyqtSignal

PyQt5 автоматично дефинира сигнали за всички вградени сигнали на Qt. Новите сигнали могат да бъдат дефинирани като атрибути на клас с помощта на фабриката pyqtSignal.

Параметри

* **типове** – типовете, които определят C++ сигнатурата на сигнала. Всеки тип може да бъде обект тип Python или низ, който е името на тип C++. Като алтернатива всеки може да бъде последователност от аргументи на типа. В този случай всяка последователност определя сигнатурата на различно претоварване на сигнала. Първото претоварване ще бъде по подразбиране.
* **име** – името на сигнала. Ако е пропуснато, тогава се използва името на атрибута на класа. Това може да бъде дадено само като аргумент за ключова дума.
* **ревизия** – ревизията на сигнала, който се експортира в QML. Това може да бъде дадено само като аргумент за ключова дума.
* **аргументи** – поредицата от имена на аргументите на сигнала, които се експортират в QML. Това може да бъде дадено само като аргумент за ключова дума.

Тип връщане

необвързан сигнал

## Свързване, прекъсване и излъчване на сигнали

Сигналите се свързват със слотове по [**connect()**](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/signals_slots.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#connect)метода на обвързан сигнал.

Параметри

* **слот** – слотът за свързване, или Python извикващ се, или друг обвързан сигнал.
* **тип** – типът на връзката, която да направите.
* **no\_receiver\_check** – потискане на проверката дали основният екземпляр на приемник на C++ все още съществува и въпреки това доставя сигнала.

Връщане

обект [за връзка](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qmetaobject-connection.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) , който може да бъде предаден на [**disconnect()**](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/signals_slots.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#disconnect). Това е единственият начин да прекъснете връзката с ламбда функция.

Сигналите се изключват от слотовете по [**disconnect()**](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/signals_slots.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#disconnect)метода на обвързан сигнал.

прекъсване на връзката

Изключете един или повече слота от сигнал. Ще се създаде изключение, ако слотът не е свързан към сигнала или ако сигналът изобщо няма връзки.

Параметри

**слот** – незадължителният слот за прекъсване на връзката, или обект [за свързване](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qmetaobject-connection.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) , върнат от [**connect()**](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/signals_slots.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#connect), извикващ Python или друг обвързан сигнал. Ако е пропуснато, всички слотове, свързани към сигнала, са изключени.

Сигналите се излъчват чрез използване на [**emit()**](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/signals_slots.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#emit)метода на обвързан сигнал.

Свързване на сигнали с помощта на аргументи на ключови думи

Възможно е да се свързват сигнали чрез подаване на слот като аргумент на ключовата дума, съответстващ на името на сигнала, когато създавате обект или използвате метода **pyqtConfigure()**.

## Типът PyQt\_PyObject аргумент на сигнала

Възможно е да прехвърлите всеки обект на Python като аргумент на сигнала, като посочите PyQt\_PyObject като типа на аргумента в подписа.

Това обикновено се използва за предаване на обекти, където действителният тип Python не е известен. Може да се използва и за предаване на цяло число, например, така че да не се изискват нормалните преобразувания от обект на Python в цяло число и обратно.

## Свързване на слотове по име

PyQt5 поддържа **connectSlotsByName()** функцията, която най-често се използва от генерирания от **pyuic5** код на Python за автоматично свързване на сигнали към слотове, които отговарят на проста конвенция за именуване. Въпреки това, когато даден клас има претоварени Qt сигнали (т.е. със същото име, но с различни аргументи), PyQt5 се нуждае от допълнителна информация, за да свърже автоматично правилния сигнал.

Декораторът [pyqtSlot()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtcore-module.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#pyqtSlot) може да се използва за указване кой от сигналите да бъде свързан към слота.

# Поддръжка за свойства на Qt

PyQt5 не поддържа настройката и получаването на свойства на Qt, сякаш са нормални атрибути на екземпляр. Това е така, защото името на свойство често е в конфликт с името на метода за получаване на свойството.Въпреки това, PyQt5 поддържа първоначалната настройка на свойствата, използвайки аргументи на ключови думи, предавани при създаване на екземпляр

## Дефиниране на нови свойства на Qt

Ново свойство на Qt може да бъде дефинирано с помощта на функцията pyqtProperty. Използва се по същия начин като стандартната property()функция на Python. Всъщност свойствата на Qt, дефинирани по този начин, също се държат като свойства на Python.

Параметри

* **тип** – вида на имота. Това е или обект от тип Python.
* **fget** – незадължителното извикване, използвано за получаване на стойността на свойството.
* **fset** – незадължителното извикване, използвано за задаване на стойността на свойството.
* **freset** – незадължителното извикване, използвано за нулиране на стойността на свойството до стойността по подразбиране.
* **fdel** – незадължителното извикване, използвано за изтриване на свойството.
* **doc** – незадължителният документен низ на свойството.
* **designable** – по избор задава Qt DESIGNABLE флаг.
* **scriptable** – по избор задава Qt SCRIPTABLE флаг.
* **съхранено** – по избор задава Qt STORED флага.
* **потребител** – по желание задава Qt USER флаг.
* **константа** – по избор задава Qt CONSTANT флага.
* **final** – по желание задава Qt FINAL флага.
* **notify** – незадължителният необвързан сигнал за известяване.
* **ревизия** – ревизията, експортирана в QML.

Тип връщане

обекта на собствеността.

Също така е възможно да използвате pyqtProperty като декоратор по същия начин като стандартната property() функция на Python. Следващият пример показва как да дефинирате int свойство с getter и setter:

Ако предпочитате терминологията на Qt, можете също да използвате (write вместо setter и read вместо getter).

# Поддръжка за OpenGL

Когато се компилира срещу Qt v5.1 или по-нова версия, PyQt5 имплементира набор от QOpenGL обвързвания за работния плот или OpenGL ES v2 обвързвания в зависимост от това как Qt е конфигуриран. Това премахва зависимостта от всякакви OpenGL обвързвания на трети страни, като PyOpenGL.

Получаването на обект, който изпълнява обвързванията за определена OpenGL версия и профил, се извършва по същия начин, както се прави от C++, т.е. чрез извикване **versionFunctions()**. В допълнение, обектът на обвързване също съдържа атрибути, съответстващи на всички константи на OpenGL.

# Поддръжка за Qt интерфейси

PyQt5 обикновено не поддържа дефинирането на клас, който наследява от повече от един Qt клас. Изключение е, когато се наследява от класове, които Qt дефинира като интерфейси, например QTextObjectInterface.

Qt интерфейс е абстрактен клас, който съдържа само чисто виртуални методи и се използва като mixin с подклас на QObject. Често се използва за дефиниране на интерфейса, който плъгинът трябва да имплементира.

Забелязваме, че PyQt5 не изисква еквивалент на макроса Q\_INTERFACES на Qt, за да използва интерфейсен клас.

# Поддръжка за [QSettings](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsettings.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp)

Qt предоставя класа [QSettings](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsettings.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) като независим от платформата API за постоянно съхранение и извличане на настройките на приложението. Настройките се извличат с помощта на **value()** метода. Типът на върнатата стойност обаче може да не е това, което се очаква. Някои платформи съхраняват само низови стойности, което означава, че типът на оригиналната стойност се губи. Следователно настройка с целочислена стойност 42 може да бъде извлечена (на някои платформи) като стойност на низ от '42'.

Като решение на този проблем изпълнението на PyQt5 на **value()** приема незадължителен трети аргумент, наречен type. Това е или обект от тип Python, напр int., или низ, който е името на тип C++, напр. 'QStringList'. Върнатата стойност ще бъде обект от искания тип.

# Интегриране на Python и QML

Qt включва QML като средство за декларативно описание на потребителски интерфейс и използване на JavaScript като скриптов език в него. Възможно е да се пишат цели самостоятелни QML приложения или да се комбинират с C++. PyQt5 позволява QML да бъде интегриран с Python по абсолютно същия начин. В частност:

* Типовете Python, които са подкласове от [QObject,](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qobject.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) могат да бъдат регистрирани с QML.
* Могат да се създават екземпляри на регистрирани типове Python и да се предоставят на QML скриптове.
* Екземпляри на регистрирани типове Python могат да бъдат създадени от QML скриптове.
* Единични екземпляри на регистрирани типове Python могат да бъдат създадени автоматично от QML машина и предоставени на QML скриптове.
* QML скриптовете взаимодействат с обектите на Python чрез техните свойства, сигнали и слотове.
* Свойствата, сигналите и слотовете на Python могат да получат номера на ревизии, така че само тези, реализирани от конкретна версия, да бъдат достъпни за QML.

## Регистриране на типове Python

Регистрирането на Python типове с QML се извършва по същия начин, както се прави с C++ класове, т.е. с помощта на функциите [qmlRegisterType()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtqml/qtqml-module.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#qmlRegisterType) , [qmlRegisterSingletonType()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtqml/qtqml-module.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#qmlRegisterSingletonType) , [qmlRegisterUncreatableType()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtqml/qtqml-module.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#qmlRegisterUncreatableType) и [qmlRegisterRevision() .](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtqml/qtqml-module.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#qmlRegisterRevision)

В C++ това са базирани на шаблон функции, които приемат C++ класа, а понякога и ревизия, като аргументи на шаблона. В реализацията на Python те просто се предават като първи аргументи на съответните функции.

## Използване на QQmlListProperty

Дефинирането на базирани на списък свойства в Python, които могат да бъдат актуализирани от QML, се извършва с помощта на класа QQmlListProperty. Въпреки това начинът, по който се използва в Python, е малко по-различен от начина, по който се използва в C++.

## Използване на прикачени свойства

За да използвате прикачени свойства в PyQt5, трябва да се предприемат следните стъпки:

* Когато извиквате [qmlRegisterType()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtqml/qtqml-module.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#qmlRegisterType) за регистриране на тип, който има прикачени свойства, типът на обекта на свойствата се предава като attachedProperties аргумент. Този тип ще се използва като фабрика за създаване на екземпляр на обекта свойства.
* Екземплярът на обект с прикачени свойства се извлича с помощта на функцията [qmlAttachedPropertiesObject()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtqml/qtqml-module.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#qmlAttachedPropertiesObject) по същия начин, както бихте направили от C++. Точно като [qmlRegisterType()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtqml/qtqml-module.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#qmlRegisterType) , [qmlAttachedPropertiesObject()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtqml/qtqml-module.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#qmlAttachedPropertiesObject) приема допълнителен първи аргумент, който е типът, който в C++ би бил аргументът на шаблона.

## Използване на източници на стойност на собственост

Източниците на стойности на свойства са имплементирани в PyQt5 по същия начин, както са имплементирани в C++. Просто подкласирайте от [QObject](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qobject.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) и [QQmlPropertyValueSource](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtqml/qqmlpropertyvaluesource.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) и осигурете имплементация на метода [setTarget()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtqml/qqmlpropertyvaluesource.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#setTarget) .

## Писане на добавки на Python за **qmlscene**

Qt позволява да се пишат плъгини, които имплементират QML модули, които могат да се зареждат динамично от C++ приложение. Тези добавки са подкласове на [QQmlExtensionPlugin](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtqml/qqmlextensionplugin.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) . PyQt5 поддържа абсолютно същото и позволява тези плъгини да бъдат написани на Python. С други думи, възможно е да се предоставят QML разширения, написани на Python, на C++ приложение и да се предоставят QML разширения, написани на C++, на Python приложение.

Модулът за разширение QML е директория, съдържаща файл, наречен qmldir. Файлът съдържа името на модула и името на плъгина, който имплементира модула. Може също така да посочи директорията, съдържаща приставката. Обикновено това не е необходимо, защото плъгинът е инсталиран в същата директория.

# Поддръжка за кооперативно мулти-наследяване

Кооперативното мулти-наследяване е техника за имплементиране на класове, които наследяват от множество родителски класове - обикновено един основен родителски клас и един или повече "mixin" класове, които добавят допълнително поведение. Тази техника прави лесно добавянето на нови "mixin" класове в по-късен етап за допълнително разширяване на поведението, без да е нужно да се променя имплементацията на класа или някой от съществуващия код, който създава инстанция на класа.

Техниката изисква всички методи init на родителските класове да следват един и същи шаблон в начина, по който обработват неразпознати ключови аргументи и използват super() за да извикват init методите на своите родителски класове.

# Неща, които трябва да знаете

## TLS поддръжка

Поддръжката за защита на транспортния слой (TLS) е все по-важна, особено на мобилни платформи, където приложението обикновено е преден край на сървър, базиран на облак. Тъй като и Python, и Qt прилагат различни API, които поддържат TLS, приложението на PyQt има избор кое да използва. Това е особено важно при внедряването на приложение, тъй като поддръжката може да се наложи да бъде включена или вградена в самото приложение.

## Аргументи на ключови думи

Qt използва QString класа за представяне на Unicode низове и QByteArray за представяне на байтови масиви или низове. В Python съответните естествени типове обекти са str и bytes. В Python v2 съответните естествени типове обекти са unicode и str.

PyQt5 прави всичко възможно, за да конвертира автоматично между обекти от различни типове. Явни преобразувания могат лесно да бъдат направени, когато е необходимо.

В някои случаи PyQt5 няма да извършва автоматични преобразувания, когато е необходимо да се прави разлика между различни претоварени методи.

Исторически QString прави разлика между празни низове и нулеви низове. Текущите версии на Qt третират нулевите низове като празни низове, но може да има друг C++ код, който приложенията на PyQt5 извикват, който поддържа разграничението. Следователно PyQt5 ще се преобразува None в null QString. Обратното преобразуване не се извършва и както нула, така и празен QString ще бъдат преобразувани в празен (т.е. нулева дължина) низ на Python.

## Множествено наследяване

Не е възможно да се дефинира нов Python клас, който подкласове от повече от един Qt клас. Изключение са класове, специално предназначени да действат като mixin класове като тези (като [QQmlParserStatus](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtqml/qqmlparserstatus.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) ), които имплементират Qt интерфейси.

## Достъп до защитени членски функции

Когато екземпляр на C++ клас не е създаден от Python, не е възможен достъп до защитените членски функции на този екземпляр. Опитите да направите това ще предизвикат изключение на Python. Също така всички методи на Python, съответстващи на виртуалните членски функции на екземпляра, никога няма да бъдат извикани.

## None и NULL

В целия PyQt5 None стойността може да бъде посочена навсякъде, където NULL е приемливо за базовия C++ код.По същия начин NULL се преобразува във None всеки път, когато се връща от базовия C++ код

# Използване на Qt Designer

Qt Designer е инструментът на Qt за проектиране и изграждане на графични потребителски интерфейси. Той ви позволява да проектирате елементи, диалози или цели основни прозорци, като използвате екранирани форми и прост интерфейс с функцията "drag-and-drop". Qt Designer има възможността да предварително преглежда вашите проекти, за да се уверите, че те работят така, както сте искали, и да ви позволи да ги прототипирате с вашите потребители, преди да започнете да пишете код.

Qt Designer използва XML .ui файлове за съхранение на дизайните и не генерира код сам по себе си. Qt включва утилитата "uic", която генерира C++ кода, който създава потребителския интерфейс. Qt включва също класа "QUiLoader", който позволява на приложението да зарежда .ui файлове и да създава съответния потребителски интерфейс динамично.

PyQt5 не опакова класа "QUiLoader", а вместо това включва Python модула "uic". Този модул, като "QUiLoader", може да зарежда .ui файлове, за да създаде потребителски интерфейс динамично. Както утилитата "uic", той също може да генерира Python код, който ще създаде потребителския интерфейс. Утилитата "pyuic5" на PyQt5 е команден интерфейс за модула "uic". Двете са описани подробно в следващите раздели.

## Използване на генерирания код

Кодът, който се генерира, има идентична структура с тази, генерирана от Qt UI може да се използва по същия начин.

Кодът е структуриран като единичен клас, който е извлечен от object типа Python. Името на класа е името на набора от обекти от най-високо ниво в Designer с Ui\_ предварителен.

Класът съдържа метод, наречен setupUi(). Това отнема един аргумент, който е изпълнимият модул, в който е създаден потребителският интерфейс. Типът на този аргумент (обикновено [QDialog](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtwidgets/qdialog.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) , [QWidget](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtwidgets/qwidget.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) или [QMainWindow](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtwidgets/qmainwindow.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) ) се задава в Designer. Ние наричаме този тип Qt базов клас .

## Писане на разширения за Qt Designer.

Qt Designer може да бъде разширен чрез писане на разширения. Обикновено това се прави с помощта на C++, но PyQt5 също позволява писането на разширения на Python. Повечето време разширението се използва, за да се представи персонализиран елемент в Designer, така че да изглежда като всеки друг елемент в кутията с елементи на Designer. Възможно е да се променят свойствата на елемента и да се свързват сигналите и слотовете му.

Също така е възможно да се добавят нови функционалности към Designer. Вижте документацията на Qt за пълните подробности. Тук ще се концентрираме върху описанието на това как да се пишат персонализирани елементи на Python.

Процесът на интегриране на персонализирани елементи на Python с Designer е много подобен на този, използван за елементите, написани на C++. Въпреки това има конкретни въпроси, които трябва да бъдат решени.

Designer трябва да има C++ разширение, което отговаря на интерфейса, дефиниран от класа QDesignerCustomWidgetInterface. (Ако разширението показва повече от един персонализиран елемент, то трябва да отговаря на интерфейса, дефиниран от класа QDesignerCustomWidgetCollectionInterface.) Освен това класът на разширението трябва да бъде подклас на QObject, както и на интерфейсния клас. PyQt5 не позволява Python класове да бъдат подклас на повече от един Qt клас.

Designer може да свързва само Qt сигнали и слотове. Той няма разбиране за Python сигнали или callable функции.

Designer може да редактира само Qt свойства, които представят C++ типове. Той няма разбиране за Python атрибути или Python типове.

# Системата за ресурси PyQt5

PyQt5 поддържа ресурсната система на Qt. Това е възможност за вграждане на ресурси като икони и файлове за превод в приложение. Това прави опаковането и разпространението на тези ресурси много по-лесно.

Файлът .qrc за събиране на ресурси е XML файл, използван за указване кои файлове с ресурси да бъдат вградени. След това приложението препраща към файловете с ресурси с техните оригинални имена, но предшествани от двоеточие.

За пълно описание, включително формата на файловете .qrc, вижте Qt Resource System в документацията на Qt.

## **pyrcc5**

**pyrcc5 е еквивалентът на PyQt5 на помощната програма rcc** на Qt и се използва по абсолютно същия начин. **pyrcc5** чете .qrc файла и файловете с ресурси и генерира Python модул, който трябва само да бъде import редактиран от приложението, за да могат тези ресурси да бъдат предоставени точно както ако са оригиналните файлове.

## QByteArray

Класът [QByteArray](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qbytearray.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя масив от байтове.

[QByteArray](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qbytearray.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) може да се използва за съхраняване както на необработени байтове (включително '\0's), така и на традиционни 8-битови низове, завършващи с '\0'. Използването [на QByteArray](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qbytearray.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) е много по-удобно от използването на . Зад кулисите той винаги гарантира, че данните са последвани от терминатор „\0“ и използва [имплицитно споделяне](https://translate.google.com/website?sl=en&tl=bg&hl=bg&client=webapp&u=https://doc.qt.io/qt-5/implicit-sharing.html) (копиране при запис), за да намали използването на паметта и да избегне ненужното копиране на данни const char \\*

В допълнение към [QByteArray](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qbytearray.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) , Qt също предоставя клас QString за съхраняване на низови данни. За повечето цели QString е класът, който искате да използвате. Той съхранява 16-битови Unicode знаци, което улеснява съхраняването на не-ASCII/не-Latin-1 знаци във вашето приложение. Освен това QString се използва навсякъде в Qt API. Двата основни случая, в които [QByteArray](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qbytearray.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) е подходящ, са когато трябва да съхранявате необработени двоични данни и когато запазването на паметта е критично (напр. с Qt за вграден Linux).

Един от начините за инициализиране на [QByteArray](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qbytearray.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) е просто да се предаде a на неговия конструктор. Например, следният код създава масив от байтове с размер 5, съдържащ данните „Hello“: const char \\*

## QColor

Класът QColor предоставя цветове на базата на RGB, HSV или CMYK стойности.

Цветът се определя нормално чрез RGB компоненти (червено, зелено и синьо), но е възможно да се определи и чрез HSV компоненти (оттенък, наситеност и стойност) и CMYK компоненти (циан, магента, жълто и черно). Освен това, цветът може да се определи и с използване на име на цвят. Името на цвета може да бъде някои от цветовете на SVG 1.0.

Конструкторът на класа QColor създава цвета на базата на RGB стойности. За да се създаде QColor на базата на HSV или CMYK стойности, съответно се използват функциите toHsv() и toCmyk(). Тези функции връщат копие на цвета, използвайки желания формат. Освен това, статичните функции fromRgb(), fromHsv() и fromCmyk() създават цветове от определените стойности. Алтернативно, цветът може да бъде преобразуван в някой от трите формата, използвайки функцията convertTo() (връщаща копие на цвета в желания формат), или някоя от функциите setRgb(), setHsv() и setCmyk() за промяна на формата на този цвят. Функцията spec() указва как е зададен цветът.

Цвят може да бъде зададен като RGB низ (например "#112233"), или ARGB низ (например "#ff112233") или име на цвят (например "синьо"), използвайки функцията setNamedColor(). Имената на цветовете се взимат от цветовете на SVG 1.0. Функцията name() връща името на цвета във формат "#RRGGBB". Цветовете могат също така да бъдат зададени с използване на setRgb(), setHsv() и setCmyk(). За получаване на по-светъл или по-тъмен цвят, използвайте функциите lighter() и darker() съответно.

Функцията isValid() указва дали QColor е валиден цвят. Например, RGB цвят с извън обхвата RGB стойности е невалиден. За целите на ефективността, QColor по-голямо внимание отделя на валидните цветове, и поради тази причина, резултатът от използване на невалиден цвят е неопределен.

Компонентите на цвета могат да бъдат извлечени индивидуално, например с функциите red(), hue() и cyan(). Стойностите на компонентите на цвета могат също да бъдат извлечени едновременно, използвайки функциите getRgb(), getHsv() и getCmyk(). Използвайки RGB модела на цвета, компонентите на цвета могат да се достъпвани и чрез функцията rgb().

Има няколко свързани не-членове: QRgb е typedef за беззнаково цяло число, което представя RGB тройката от стойности (r, g, b). Обърнете внимание, че той може да съдържа стойност за алфа-канала (за повече информация вижте секцията "Alpha-Blended Drawing"). Функциите qRed(), qBlue() и qGreen() връщат съответната компонента на дадената стойност на QRgb, докато функциите qRgb() и qRgba() създават и връщат тройката QRgb, базирана на дадените стойности на компонентите. Накрая, функцията qAlpha() връща алфа компонентата на предоставеното QRgb, а функцията qGray() изчислява и връща стойност за сив цвят, базирана на дадената стойност.

QColor е независим от платформата и устройството. Класът QColormap картира цвета към хардуера.

## Предварително дефинирани цветове

Има 20 предварително дефинирани QColors, описани от [GlobalColor](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#GlobalColor) enum, включително черен, бял, първичен и вторичен цвят, по-тъмни версии на тези цветове и три нюанса на сивото. [QColor](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qcolor.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) също разпознава различни имена на цветове; статичната функция [colorNames()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qcolor.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#colorNames) връща имена на цветове от QStringList, за които [QColor](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qcolor.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) знае.

Освен това цветовете [color0](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#GlobalColor-color0) , [color1](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#GlobalColor-color1) и [прозрачните](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#GlobalColor-transparent) цветове се използват за специални цели.

[color0](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#GlobalColor-color0) (нулева пикселна стойност) и [color1](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#GlobalColor-color1) (ненулева пикселна стойност) са специални цветове за рисуване в QBitmaps. Рисуването с [цвят0](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#GlobalColor-color0) задава битовете на растерното изображение на 0 (прозрачни; т.е. фон), а рисуването с [цвят1](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#GlobalColor-color1) задава битовете на 1 (непрозрачни; т.е. преден план).

[transparent](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#GlobalColor-transparent) се използва за обозначаване на прозрачен пиксел. Когато рисувате с тази стойност, ще се използва стойност на пиксел, която е подходяща за основния използван пикселен формат.

## QDate

Класът [QDate](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя функции за дата.

Обект [QDate](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) кодира календарна дата, т.е. номера на година, месец и ден, в пролептичния григориански календар по подразбиране. Може да чете текущата дата от системния часовник. Предоставя функции за сравняване на дати и за манипулиране на дати. Например, възможно е да добавяте и изваждате дни, месеци и години към дати.

Обект [QDate](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) обикновено се създава чрез изрично посочване на номерата на годината, месеца и деня. Имайте предвид, че [QDate](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) интерпретира двуцифрените години, както са представени, т.е. като години от 0 до 99, без да добавя никакво отместване. QDate може също да бъде конструиран със статичната функция [currentDate()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#currentDate) , която създава обект [QDate ,](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) съдържащ датата на системния часовник. Изрична дата може също да бъде зададена с помощта на [setDate()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#setDate) . Функцията [fromString()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#fromString) връща [QDate](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) даден низ и формат на дата, който се използва за интерпретиране на датата в низа.

Функциите year [()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#year) , [month()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#month) и [day()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#day) осигуряват достъп до номерата на годината, месеца и деня. Освен това са предоставени функциите [dayOfWeek()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#dayOfWeek) и [dayOfYear() .](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#dayOfYear)Същата информация се предоставя в текстов формат от функциите [toString()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#toString) , [shortDayName()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#shortDayName) , [longDayName()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#longDayName) , [shortMonthName()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#shortMonthName) и [longMonthName()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#longMonthName) .

[QDate](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя пълен набор от оператори за сравняване на два [QDate](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) обекта, където по-малък означава по-рано, а по-голям означава по-късно.

Можете да увеличите (или намалите) дата с определен брой дни, като използвате [addDays()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#addDays) . По подобен начин можете да използвате [addMonths()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#addMonths) и [addYears()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#addYears) . Функцията [daysTo()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#daysTo) връща броя на дните между две дати.

Функциите daysInMonth [()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#daysInMonth) и [daysInYear()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#daysInYear) връщат съответно колко дни има в месеца и годината на тази дата. Функцията [isLeapYear()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#isLeapYear) показва дали дадена дата е във високосна година.

Диапазон от валидни дати

Датите се съхраняват вътрешно като номер на юлианския ден, цяло число за всеки ден в непрекъснат диапазон, като 24 ноември 4714 г. пр. н. е. в григорианския календар е юлиански ден 0 (1 януари 4713 г. пр. н. е. в юлианския календар). Освен че е ефективен и точен начин за съхраняване на абсолютна дата, той е подходящ за преобразуване на дата в други календарни системи като иврит, ислямска или китайска. Номерът на юлианския ден може да бъде получен с помощта на [toJulianDay()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#toJulianDay) и може да бъде зададен с помощта на [fromJulianDay()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#fromJulianDay) .

Диапазонът от дати, които могат да бъдат съхранени от [QDate](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) като число на юлианския ден, е по технически причини ограничен до между -784350574879 и 784354017364, което означава от преди 2 милиарда пр. н. е. до след 2 милиарда н. е.

## QDateTime

Класът [QDateTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя функции за дата и час.

Обект [QDateTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) кодира календарна дата и час на часовника („дата и час“). Той съчетава характеристиките на класовете [QDate](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdate.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) и [QTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) . Може да чете текущата дата и час от системния часовник. Предоставя функции за сравняване на дати и часове и за манипулиране на дати и часове чрез добавяне на брой секунди, дни, месеци или години.

Обект [QDateTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) обикновено се създава или чрез изрично даване на дата и час в конструктора, или чрез използване на статичната функция [currentDateTime()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#currentDateTime) , която връща обект [QDateTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) , зададен на времето на системния часовник. Датата и часът могат да се променят с [setDate()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#setDate) и [setTime()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#setTime) . Дата и час може също да бъде зададена с помощта на функцията [setTime\_t()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#setTime_t) , която приема стандартна за POSIX стойност „брой секунди от 00:00:00 на 1 януари 1970 г.“. Функцията [fromString()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#fromString) връща [QDateTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) , даден низ и формат на датата, използвани за интерпретиране на датата в низа.

Функциите date [()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#date) и [time()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#time) осигуряват достъп до частите за дата и час на datetime. Същата информация се предоставя в текстов формат от функцията [toString()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#toString) .

[QDateTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя пълен набор от оператори за сравняване на два обекта [QDateTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) , където по-малък означава по-рано, а по-голям означава по-късно.

Можете да увеличите (или намалите) дадена дата и час с определен брой милисекунди с помощта на [addMSecs()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#addMSecs) , секунди с помощта на [addSecs()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#addSecs) или дни с помощта на [addDays()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#addDays) . По подобен начин можете да използвате [addMonths()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#addMonths) и [addYears()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#addYears) . Функцията [daysTo()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#daysTo) връща броя на дните между две дати, [secsTo()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#secsTo) връща броя секунди между две дати, а [msecsTo()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#msecsTo) връща броя милисекунди между две дати.

[QDateTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) може да съхранява датите като [LocalTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#TimeSpec-LocalTime) или като [UTC](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#TimeSpec-UTC) . [currentDateTime()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#currentDateTime) връща [QDateTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) , изразено като местно време; използвайте [toUTC()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#toUTC) , за да го конвертирате в UTC. Можете също да използвате [timeSpec()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#timeSpec) , за да разберете дали обект [QDateTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) съхранява UTC време или местно време. Операции като [addSecs()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#addSecs) и [secsTo()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#secsTo) са наясно с лятното часово време (DST).

Няма година 0

Няма година 0. Датите в тази година се считат за невалидни. Годината -1 е годината „1 преди Христа“ или „1 преди настоящата ера“. Денят преди 1 януари 1 г. е 31 декември 1 г. пр. н. е

## QKeySequence

Класът [QKeySequence](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qkeysequence.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) капсулира последователност от клавиши, използвана от преки пътища.

В най-често срещаната си форма последователността от клавиши описва комбинация от клавиши, които трябва да се използват заедно, за да извършат някакво действие. Клавишните последователности се използват с QAction обекти, за да се укаже кои клавишни комбинации могат да се използват за задействане на действия.

Клавишните последователности могат да бъдат конструирани за използване като клавишни комбинации по три различни начина:

* За стандартни преки пътища може да се използва [StandardKey за заявка на специфичната за платформата клавишна последователност, свързана с всеки пряк път.](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qkeysequence.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#StandardKey)
* За персонализирани преки пътища могат да се използват четливи низове като „Ctrl+X“ и те могат да бъдат преведени в подходящите преки пътища за потребители на различни езици. Преводите се правят в контекста „QShortcut“.
* За твърдо кодирани преки пътища целочислените клавишни кодове могат да бъдат зададени с комбинация от стойности, определени от стойностите на [Key](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#Key) и [Modifier](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#Modifier) enum. Всеки ключов код се състои от една стойност [на ключ](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#Key) и нула или повече модификатори, като [SHIFT](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#Modifier-SHIFT) , [CTRL](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#Modifier-CTRL) , [ALT](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#Modifier-ALT) и [META](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qt.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#Modifier-META) .

## QLine

Класът [QLine](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qline.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя двуизмерен вектор, използващ целочислена точност.

QLine описва линия с крайна дължина ( [или](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qline.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) линеен сегмент) върху двуизмерна повърхност. Началната и крайната точка на линията се определят с целочислена точност на точките за координати. Използвайте конструктора [QLineF](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qlinef.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) , за да извлечете копие с плаваща запетая.

Позициите на началната и крайната точка на линията могат да бъдат извлечени с помощта на функциите [p1()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qline.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#p1) , [x1()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qline.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#x1) , [y1()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qline.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#y1) , [p2()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qline.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#p2) , [x2()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qline.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#x2) и [y2()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qline.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#y2) . Функциите [dx()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qline.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#dx) и [dy()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qline.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#dy) връщат хоризонталните и вертикалните компоненти на линията. Използвайте [isNull()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qline.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#isNull) , за да определите дали [QLine](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qline.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) представлява валиден ред или нулев ред.

И накрая, линията може да бъде преведена с дадено отместване с помощта на функцията [translate()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qline.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#translate).

## QPoint

Класът [QPoint](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qpoint.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) дефинира точка в равнината с целочислена точност.

Точка се определя чрез координата x и координата y, които могат да бъдат достъпни чрез функциите [x()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qpoint.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#x) и [y()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qpoint.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#y) . Функцията isNull [()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qpoint.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#isNull) връща, true ако x и y са зададени на 0. Координатите могат да бъдат зададени (или променени) с помощта на функциите [setX()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qpoint.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#setX) и [setY()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qpoint.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#setY) или алтернативно функциите rx() и ry(), които връщат препратки към координатите (позволяващи директна манипулация).

QPoint обект може да се използва и като вектор: Събирането и изваждането се дефинират като за вектори (всеки компонент се добавя отделно) [. QPoint](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qpoint.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) обект може също да бъде разделен или умножен по int или qreal.

В допълнение, класът [QPoint](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qpoint.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя функцията [manhattanLength()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qpoint.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#manhattanLength) , която дава евтино приближение на дължината на обекта [QPoint](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qpoint.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) , интерпретиран като вектор. И накрая, обектите [на QPoint](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qpoint.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) могат да се предават поточно, както и да се сравняват.

## QPolygon

Класът [QPolygon](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qpolygon.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя вектор от точки с целочислена точност.

Обект [QPolygon](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qpolygon.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) е QVector< [QPoint](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qpoint.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) >. Най-лесният начин за добавяне на точки към [QPolygon](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qpolygon.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) е да използвате стрийминг оператора на QVector, както е илюстрирано по-долу:

В допълнение към функциите, предоставени от QVector, [QPolygon](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qpolygon.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя някои специфични за точки функции.

Всяка точка в многоъгълник може да бъде извлечена чрез предаване на нейния индекс на функцията [point()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qpolygon.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#point) . За да попълни многоъгълника, [QPolygon](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qpolygon.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя функцията [setPoint()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qpolygon.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#setPoint) за задаване на точката при даден индекс, функцията [setPoints()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qpolygon.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#setPoints) за задаване на всички точки в многоъгълника (преоразмеряването му до дадения брой точки) и putPoints [()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qpolygon.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#putPoints) функция, която копира определен брой дадени точки в многоъгълника от определен индекс (преоразмеряване на многоъгълника, ако е необходимо).

[QPolygon](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qpolygon.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя функциите [boundingRect()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qpolygon.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#boundingRect) и [translate()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qpolygon.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#translate) за геометрични функции. Използвайте функцията QMatrix::map() за по-общи трансформации на QPolygons.

Класът [QPolygon](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtgui/qpolygon.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) е [имплицитно споделен](https://translate.google.com/website?sl=en&tl=bg&hl=bg&client=webapp&u=https://doc.qt.io/qt-5/implicit-sharing.html).

## QSize

Класът [QSize](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) дефинира размера на двуизмерен обект, използвайки прецизност на целите точки.

Размерът се определя от [width()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#width) и [height()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#height) . Може да се зададе в конструктора и да се промени с помощта на функциите [setWidth()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#setWidth) , [setHeight()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#setHeight) или [scale()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#scale) или с помощта на аритметични оператори. Размерът може също да се манипулира директно чрез извличане на препратки към ширината и височината с помощта на функциите rwidth() и rheight(). И накрая, ширината и височината могат да бъдат разменени с помощта на функцията [transpose()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#transpose) .

Функцията [isValid()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#isValid) определя дали даден размер е валиден (валидният размер има ширина и височина, по-големи или равни на нула). Функцията [isEmpty()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#isEmpty) връща, true ако някоя от ширината и височината е по-малка или равна на нула, докато функцията [isNull()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#isNull) връща true само ако и ширината, и височината са нула.

Използвайте функцията [expandedTo(),](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#expandedTo) за да извлечете размер, който съдържа максималната височина и ширина на този размер и даден размер. По подобен начин функцията [boundedTo()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#boundedTo) връща размер, който съдържа минималната височина и ширина на този размер и даден размер.

[QSize](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) обектите могат да се предават поточно, както и да се сравняват.

## QSizeF

Класът [QSizeF](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsizef.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) дефинира размера на двуизмерен обект, използвайки прецизност с плаваща запетая.

Класът [QSizeF](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsizef.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) също предоставя функцията [toSize()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsizef.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#toSize) , връщаща копие [на QSize](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsize.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) с този размер, конструирано чрез закръгляване на ширината и височината до най-близките цели числа.

[QSizeF](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qsizef.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) обектите могат да се предават поточно, както и да се сравняват.

## QTime

Класът [QTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя функции за време на часовника.

Обект [QTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) съдържа време на часовника, което може да изрази като брой часове, минути, секунди и милисекунди от полунощ. Той може да чете текущото време от системния часовник и да измерва диапазон от изминало време. Той предоставя функции за сравняване на времена и за манипулиране на време чрез добавяне на брой милисекунди.

[QTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) използва 24-часов формат на часовника; няма концепция за AM/PM. За разлика от [QDateTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qdatetime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) , [QTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) не знае нищо за часовите зони или лятното часово време (DST).

Обект [QTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) обикновено се създава или чрез изрично посочване на броя часове, минути, секунди и милисекунди, или чрез използване на статичната функция [currentTime()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#currentTime) , която създава обект [QTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) , който съдържа локалното време на системата. Имайте предвид, че точността зависи от точността на основната операционна система; не всички системи осигуряват точност от 1 милисекунда.

Функциите hour [()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#hour) , [minute()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#minute) , [second()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#second) и [msec()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#msec) осигуряват достъп до броя часове, минути, секунди и милисекунди от времето. Същата информация се предоставя в текстов формат от функцията [toString()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#toString) .

Функциите addSecs [()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#addSecs) и [addMSecs()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#addMSecs) предоставят времето с определен брой секунди или милисекунди по-късно от дадено време. Съответно, броят секунди или милисекунди между два пъти може да бъде намерен с помощта на [secsTo()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#secsTo) или [msecsTo()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#msecsTo) .

[QTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя пълен набор от оператори за сравняване на два [QTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) обекта; по-ранен момент се счита за по-малък от по-късен; ако A. [msecsTo()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#msecsTo) (B) е положителен, тогава A < B.

[QTime](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) може да се използва за измерване на диапазон от изминало време с помощта на функциите [start()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#start) , [restart()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#restart) и [elapsed() .](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtime.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#elapsed)

# Използване на PyQt5 от Python Shell

PyQt5 инсталира кука за въвеждане (използвайки PyOS\_InputHook), която обработва събития, когато интерактивен интерпретатор чака въвеждане от потребителя. Това означава, че можете например да създавате джаджи от подканата на обвивката на Python, да взаимодействате с тях и все пак да можете да въвеждате други команди на Python.

Интернационализация на приложенията на PyQt5

PyQt5 и Qt включват изчерпателен набор от инструменти за превод на приложения на местни езици. За пълно описание вижте Qt Linguist Manual в документацията на Qt.

Процесът на интернационализация на приложение включва следните стъпки.

* Програмистът използва **pylupdate5,** за да създаде или актуализира .ts файл за превод за всеки език, на който приложението трябва да бъде преведено. Файлът .ts е XML файл, който съдържа низовете за превод и съответните преводи, които вече са направени. **pylupdate5** може да се изпълнява произволен брой пъти по време на разработката, за да се актуализират .ts файловете с най-новите низове за превод.
* Преводачът използва Qt Linguist, за да актуализира .ts файловете с преводи на низовете.
* След това мениджърът на версиите използва помощната програма **lrelease** на Qt , за да конвертира .ts файловете във .qm файлове, които са компактни двоични еквиваленти, използвани от приложението. Ако дадено приложение не може да намери подходящ .qm файл или определен низ не е преведен, вместо това се използват низовете, използвани в оригиналния изходен код.
* Управлението на изданието може по желание да използва **pyrcc5** за вграждане на .qm файловете, заедно с други ресурси на приложението, като икони, в Python модул. Това може да улесни опаковането и разпространението на приложението.

## **pylupdate5**

**pylupdate5** е еквивалентът на PyQt5 на помощната програма **lupdate** на Qt и се използва по абсолютно същия начин. Чете се проектен файл на Qt .pro, който посочва изходните файлове на Python и интерфейсните файлове на Qt Designer, от които се извлича текстът, който трябва да бъде преведен. Файлът .pro също така посочва .ts файловете за превод, които **pylupdate5** актуализира (или създава, ако е необходимо) и впоследствие се използват от Qt Linguist.

## Разлики между PyQt5 и Qt

Qt реализира поддръжка за интернационализация чрез класа [QTranslator](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qtranslator.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) и методите [translate()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qcoreapplication.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#translate) и [tr()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qobject.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#tr) . Обикновено [tr()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qobject.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#tr) се използва за получаване на правилния превод на съобщение. Процесът на превод използва контекст на съобщение, за да позволи едно и също съобщение да бъде преведено по различен начин. В Qt [tr()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qobject.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#tr) всъщност се генерира от mocи използва твърдо кодираното име на клас като контекст. От друга страна, translate позволява контекстът да бъде посочен изрично.

За съжаление, поради начина, по който Qt имплементира [tr(),](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qobject.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#tr) не е възможно PyQt5 да възпроизведе точно поведението му. Реализацията на [tr()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qobject.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#tr) в PyQt5 използва името на класа на екземпляра като контекст. Ключовата разлика и източникът на потенциални проблеми е, че контекстът се определя динамично в PyQt5, но е твърдо кодиран в Qt. С други думи, контекстът на превод може да се промени в зависимост от йерархията на класовете на екземпляра

Поведението на PyQt5 е незадоволително и може да бъде променено в бъдеще. Препоръчително е [translate()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qcoreapplication.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#translate) да се използва с предпочитание пред [tr()](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtcore/qobject.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp#tr) . Това гарантирано работи с настоящи и бъдещи версии на PyQt5 и прави много по-лесно споделянето на файлове със съобщения между Python и C++ код.

Поддръжка на DBus

PyQt5 предоставя два различни модула, които реализират поддръжка за DBus. Модулът [QtDBus](https://www-riverbankcomputing-com.translate.goog/static/Docs/PyQt5/api/qtdbus/qtdbus-module.html?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) предоставя обвивки за стандартните класове Qt DBus. Модулът dbus.mainloop.pyqt5 добавя поддръжка за Qt цикъл на събития към стандартния dbus-python модул на Python.

Внедряване на търговски приложения на PyQt5

Внедряването на комерсиални приложения на PyQt5 може да бъде сложен процес поради редица причини:

* Обикновено е по-добре да не разчитате на предварително необходимите пакети, които вече са инсталирани в системата на потребителя. Това означава, че освен кода на вашето приложение, трябва да включите интерпретатора на Python, стандартната библиотека, пакети и модули за разширение на трети страни и самия Qt.
* Някои целеви платформи (например iOS) имат ограничения за това как се създава приложение, за да бъде включено в магазините за приложения.
* Необходимо е да се разубедят потребителите да имат сами достъп до основните модули на PyQt5. Потребител, който е използвал модулите, доставени с вашето приложение, за разработване на нови приложения, сам ще се счита за разработчик и ще се нуждае от собствен търговски лиценз за PyQt5.

Препоръчителното решение за всички тези проблеми е да използвате [pyqtdeploy](https://www-riverbankcomputing-com.translate.goog/software/pyqtdeploy/?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) .

# API за разширение на PyQt5

Важна характеристика на PyQt5 (и SIP генерираните модули като цяло) е възможността други модули за разширение да надграждат върху него. [QScintilla](https://www-riverbankcomputing-com.translate.goog/software/qscintilla/?_x_tr_sl=en&_x_tr_tl=bg&_x_tr_hl=bg&_x_tr_pto=wapp) е такъв пример.

PyQt5 предоставя API за разширение, което може да се използва от други модули. Това има предимството на споделяне на код и също така налагане на последователно поведение. Част от API е достъпна от Python, а част от C++.

Списък със използвана литература:

https://www.riverbankcomputing.com/static/Docs/PyQt5/index.html