

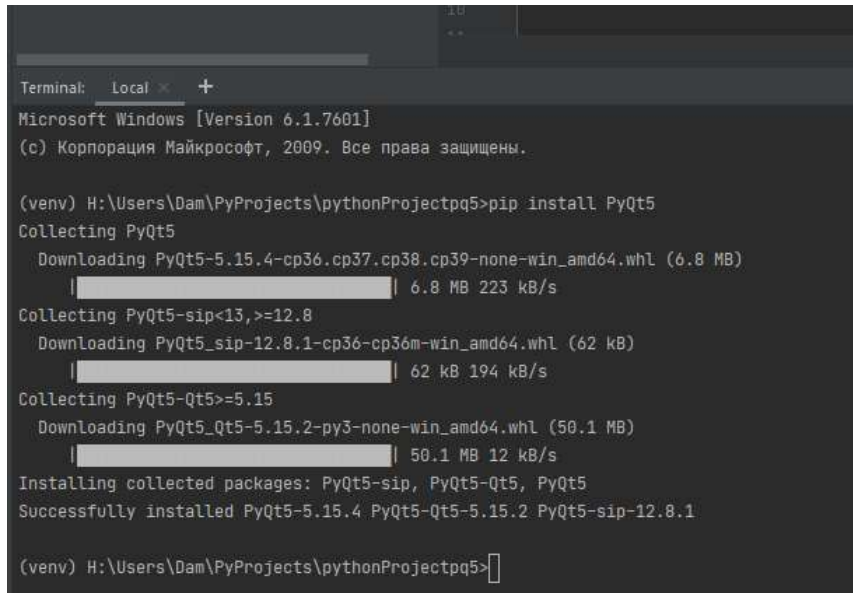
Основы оконного интерфейса в Python.

Для разработки пользовательского интерфейса в Python используется библиотека PyQt5.

1. Установка PyQt5

Вводим в терминале PyCharm команду:

```
pip install PyQt5
```

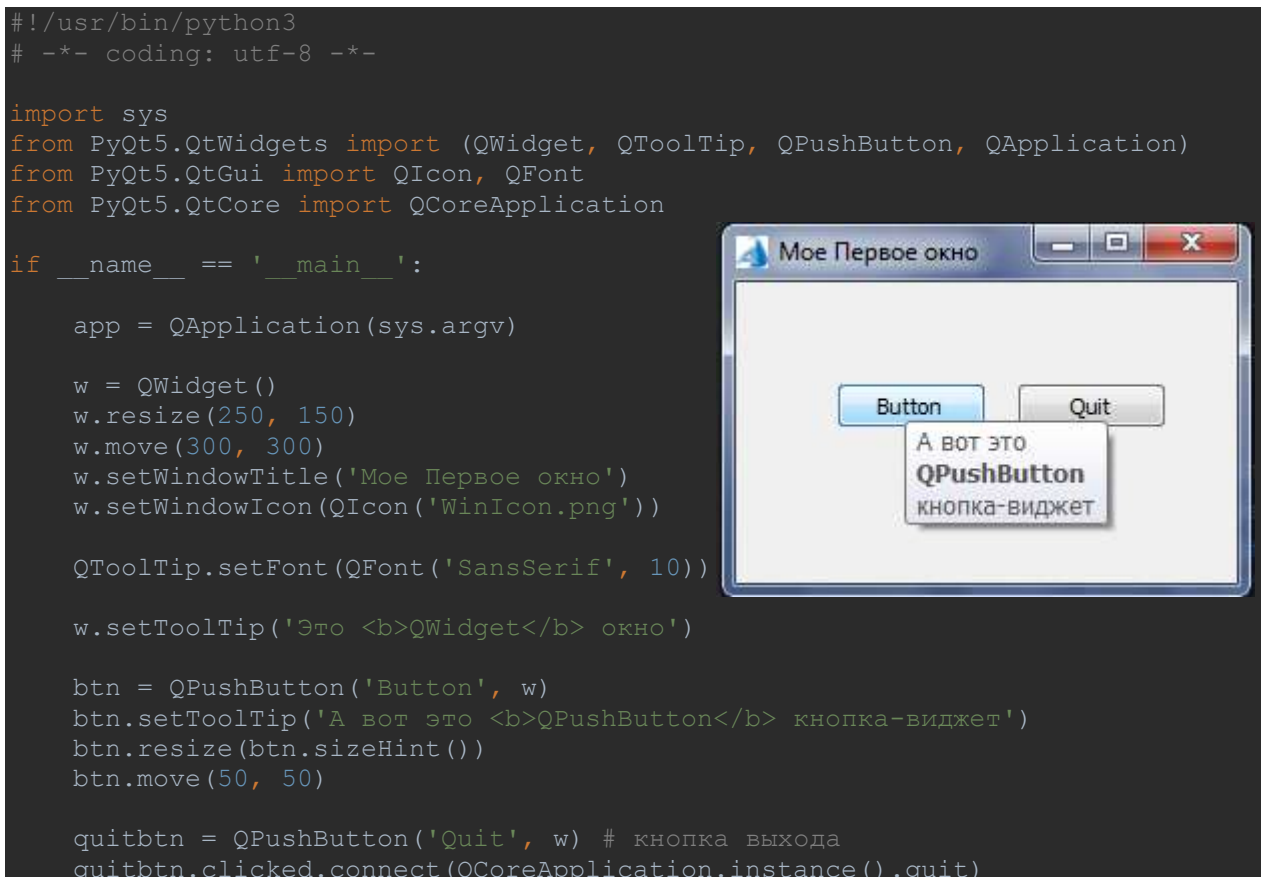


```
Terminal: Local x +
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт, 2009. Все права защищены.

(venv) H:\Users\Dam\PyProjects\pythonProjectpq5>pip install PyQt5
Collecting PyQt5
  Downloading PyQt5-5.15.4-cp36.cp37.cp38.cp39-none-win_amd64.whl (6.8 MB)
    | 6.8 MB 223 kB/s
Collecting PyQt5-sip<13,>=12.8
  Downloading PyQt5_sip-12.8.1-cp36-cp36m-win_amd64.whl (62 kB)
    | 62 kB 194 kB/s
Collecting PyQt5-Qt5>=5.15
  Downloading PyQt5_Qt5-5.15.2-py3-none-win_amd64.whl (50.1 MB)
    | 50.1 MB 12 kB/s
Installing collected packages: PyQt5-sip, PyQt5-Qt5, PyQt5
Successfully installed PyQt5-5.15.4 PyQt5-Qt5-5.15.2 PyQt5-sip-12.8.1

(venv) H:\Users\Dam\PyProjects\pythonProjectpq5>
```

2. Создаем первое окно с двумя кнопками:



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

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

if __name__ == '__main__':

    app = QApplication(sys.argv)

    w = QWidget()
    w.resize(250, 150)
    w.move(300, 300)
    w.setWindowTitle('Мое Первое окно')
    w.setWindowIcon(QIcon('WinIcon.png'))

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

    w.setToolTip('Это <b>QWidget</b> окно')

    btn = QPushButton('Button', w)
    btn.setToolTip('А вот это <b>QPushButton</b> кнопка-виджет')
    btn.resize(btn.sizeHint())
    btn.move(50, 50)

    quitbtn = QPushButton('Quit', w) # кнопка выхода
    quitbtn.clicked.connect(QCoreApplication.instance().quit)
```

```
quitbtn.resize(quitbtn.sizeHint())
quitbtn.move(140, 50)

w.show()

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

3. Разберем код:

```
import sys
```

```
from PyQt5.QtWidgets import (QWidget, QToolTip, QPushButton,
                              QApplication)
```

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

```
from PyQt5.QtGui import QIcon, QFont
```

```
from PyQt5.QtCore import QApplication
```

Импортируем работу с иконкой окна, настройки шрифта для подсказок и метод закрытия окна.

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

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

```
w = QWidget()
```

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

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

Метод resize() изменяет размеры виджета. Он стал 250 пикселей в ширину и 150 в высоту.

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

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

```
w.setWindowTitle('Мое Первое окно')
```

```
w.setWindowIcon(QIcon('WinIcon.png'))
```

Здесь мы задаём заголовок нашего окна и подключаем иконку. Заметим, что иконка должна находиться в той же папке что и программа на Python, либо нужно указать точный путь к файлу иконки перед ее именем.

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

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

```
w.setToolTip('Это <b>QWidget</b> окно')
```

Чтобы создать всплывающую подсказку, мы вызываем метод setToolTip(). Мы можем использовать форматирование текста (например - выделить жирным). Подсказка будет появляться возле курсора мышки в любом месте окна, когда вы приостанавливаете движение мышки.

```
btn = QPushButton('Button', w)
```

```
btn.setToolTip('А вот это <b>QPushButton</b> кнопка-виджет')
```

Мы создаем виджет кнопки и устанавливаем подсказку для него. Подсказка появится, если вы остановите курсор мышки на кнопке.

```
btn.resize(btn.sizeHint())
```

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

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

```
quitbtn = QPushButton('Quit', w)
```

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

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

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

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

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

```
w.show()
```

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

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

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

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

4. Message Box

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

[illegible]

```

        if reply == QMessageBox.Yes:

            event.accept()

        else:

            event.ignore()

if __name__ == '__main__':

    app = QApplication(sys.argv)

    ex = Example()

    sys.exit(app.exec_())

```

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

```

reply = QMessageBox.question(self, 'Message',

    "Are you sure to quit?", QMessageBox.Yes |

    QMessageBox.No, QMessageBox.No)

```

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

```

if reply == QtGui.QMessageBox.Yes:

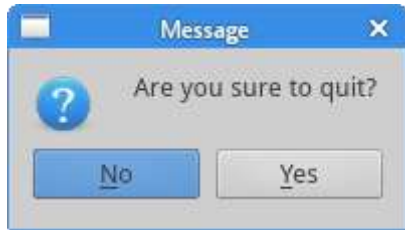
    event.accept()

else:

    event.ignore()

```

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



Центрирование окна на экране

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

```
#!/usr/bin/python3

# -*- coding: utf-8 -*-

import sys

from PyQt5.QtWidgets import QWidget, QDesktopWidget, QApplication

class Example(QWidget):

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

        self.initUI()
```

```
def initUI(self):
```

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

```
    self.center()
```

```
    self.setWindowTitle('Center')
```

```
    self.show()
```

```
def center(self):
```

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

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

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

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

```
if __name__ == '__main__':
```



```
app = QApplication(sys.argv)

ex = Example()

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

Задание на ЛР: выполнять на языке Python

1. Создайте не менее 3х форм.
2. Создайте кнопки для вызова и перехода между различными формами.
3. Создайте несколько кнопок для вызова окна сообщений MessageBox с различным набором параметров.
4. Количество различных вариантов вызова окна сообщений – не менее 5.
5. Добавить обработчики разных ответов в окне MessageBox
6. Используя формы и MessageBox запрограммировать простой «квест», тест или задачу принятия решения на основе ответов пользователя (метод дерева решений), например, выбор меню в ресторане, оценка темперамента личности и т.д.
7. Нарисовать блок-схему алгоритма (дерево решений) и обязательно указать какие формы вызываются и какие сообщения MessageBox выдаются (не менее 6ти MessageBox и не менее 4х форм).
8. Напишите программу-калькулятор:



9. Подготовьте отчет по всем пунктам