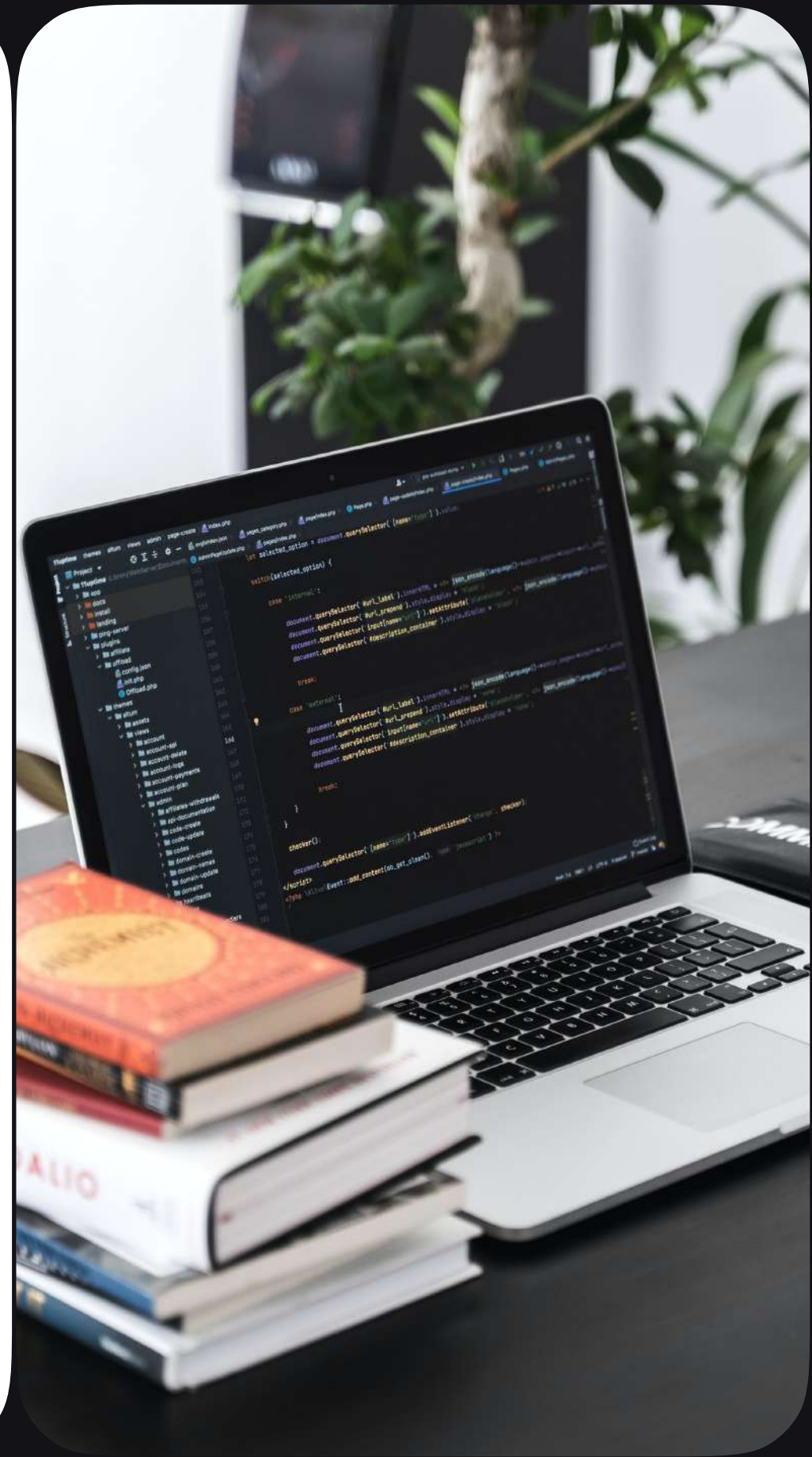
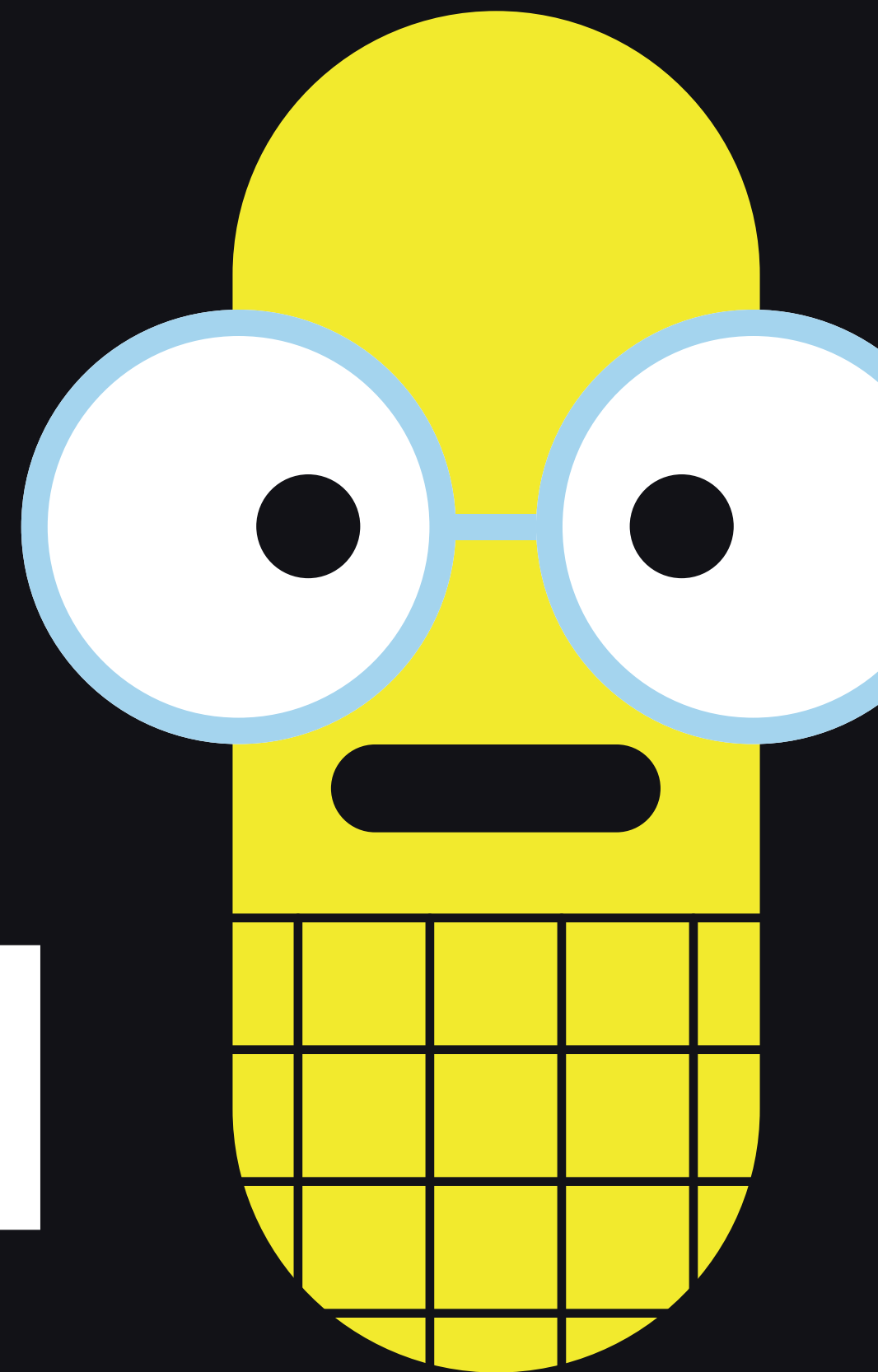


Занятие 5

Введение в PyQt



Теория



PyQt

PyQt — реализация мощного фреймворка **Qt** для языка Python. В основном, для создания GUI, используется именно он, хотя в питоне есть и встроенный фреймворк — **tkinter**. Основное понятие в PyQt — это виджет.



Виджет — это своеобразный элемент управления, который мы можем «подогнать» под свой дизайн, свой интерфейс.

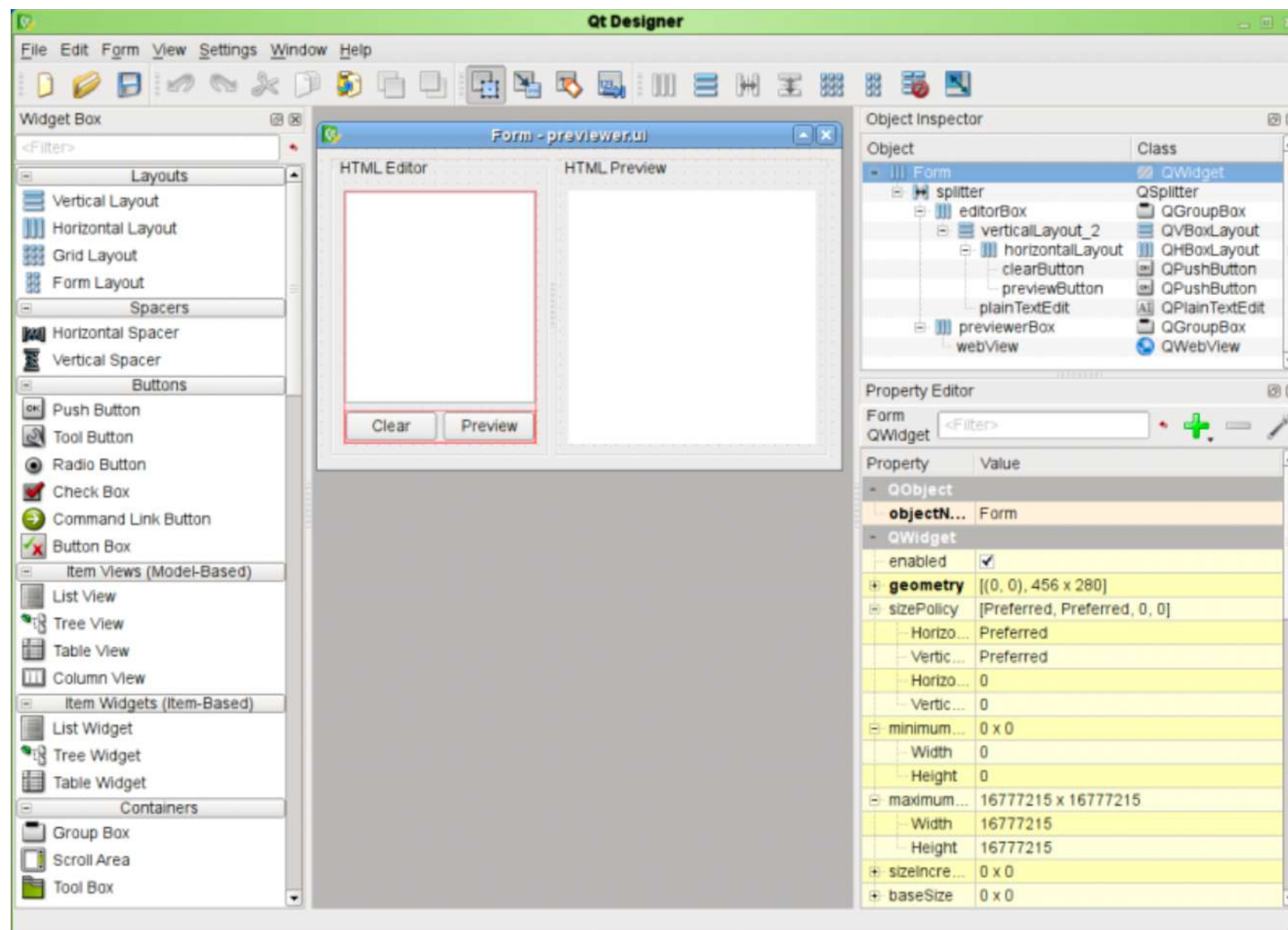
Как же создается интерфейс для приложения в PyQt?

Сделать это можно как программно, через код, так и в дизайнере — специальном **GUI-конструкторе**.

Продвинутые разработчики рекомендуют использовать именно программный способ — в этом случае вы получаете полный контроль над интерфейсом и понимаете суть работы с ним.



Установить PyQt можно через консоль — командой `pip install pyqt5`, либо вбив PyQt5 в настройках интерпретатора.





Создание простого приложения с графическим интерфейсом с использованием PyQt включает в себя **следующие шаги**:



Импортировать модуль **QtGui**



Создайте объект приложения



Объект **QWidget** создает окно верхнего уровня.
Добавьте в него объект **QLabel**



Установите заголовок ярлыка «Привет, мир»



Определите размер и положение окна методом **setGeometry ()**



Введите основной цикл приложения методом **app.exec_ ()**

- ✓ Импортировать модуль `QtGui`
- ✓ Создайте объект приложения
- ✓ Объект `QWidget` создает окно верхнего уровня. Добавьте в него объект `QLabel`
- ✓ Установите заголовок ярлыка «Привет, мир»
- ✓ Определите размер и положение окна методом `setGeometry()`
- ✓ Введите основной цикл приложения методом `app.exec_()`

```
import sys
from PyQt4 import QtGui

def window():
    app = QtGui.QApplication(sys.argv)
    w = QtGui.QWidget()
    b = QtGui.QLabel(w)
    b.setText("Hello World")
    w.setGeometry(100,100,200,50)
    b.move(50,20)
    w.setWindowTitle("PyQt")
    w.show()
    sys.exit(app.exec_())

if __name__=='__main__':
    window()
```


PyQt API это...

— большая коллекция классов и методов. Эти классы определены в более чем 20 модулях. Ниже представлены наиболее популярные:

QtCore

Основные не-GUI классы,
используемые другими модулями

QtGui

Компоненты графического
интерфейса пользователя

QtOpenGL

Поддержка классов OpenGL

QtSql

Классы для интеграции баз данных
с использованием SQL

QtSvg

Классы для отображения
содержимого файлов SVG

QtWebKit

Классы для рендеринга
и редактирования HTML

QtXml

Классы для обработки XML

QtDesigner

Классы для расширения Qt Designer

QtAssistant

Поддержка онлайн-справки

Наиболее популярные виджеты:

QLineEdit

Позволяет пользователю ввести одну строку текста

QRadioButton

Позволяет выбрать один из нескольких вариантов

QLabel

используется для отображения текста или изображения

QPushButton

Командная кнопка для вызова действия

QLabel

Используется для отображения текста или изображения

QTextEdit

Позволяет пользователю вводить многострочный текст

QCheckBox

Позволяет выбрать более одного варианта



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

Каждое приложение **PyQt5** должно создать объект приложения. Параметр **sys.argv** — это список аргументов из командной строки.

Скрипты Python могут быть запущены из программной оболочки. Это один из способов, как мы можем контролировать запуск наших скриптов.



```
w = QWidget()
```

Виджет **QWidget** — это основной класс всех объектов пользовательского интерфейса в PyQt5. Мы обеспечиваем конструктор по умолчанию для QWidget.

Конструктор по умолчанию не имеет родителя. Виджет без родителя называется окном.

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

Метод `resize()` изменяет размер виджета. Здесь задана ширина 250px и высота 150px.

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

Метод `move()` перемещает виджет на позицию с координатами $x = 300$ и $y = 300$ на экране.

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

Здесь мы устанавливаем название нашего окна. Оно показывается в строке заголовка.

```
w.show()
```

Метод `show()` отображает виджет на экране. Виджет сначала создаётся в памяти и позже показывается на экране.



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

Наконец, мы входим в главный цикл приложения. Обработка событий начинается в этой точке. Главный цикл получает события из системы и отправляет их виджетам приложения. Цикл завершается, если мы вызываем метод **exit()** или главное окно было закрыто.

Метод **sys.exit()** гарантирует чистый выход. Среда будет проинформирована, когда приложение завершится.

Метод **exec_()** содержит нижнее подчеркивание, потому что **exec_** уже используемое имя в Python. И, соответственно, имя **exec_()** было использовано взамен.

Объектно-ориентированное программирование


Есть три важных вещи в ООП — это классы, данные и методы. Здесь мы создаём новый класс с именем **Example**. Класс **Example** наследуется из класса **QWidget**. Это значит, что мы вызываем два конструктора: первый для класса **Example** и второй для унаследованного класса.

Метод `super()` возвращает объект родителя класса **Example** и мы вызываем его конструктор. Метод `__init__()` — это конструктор класса в языке Python.

```
class Example(QWidget):  
  
    def __init__(self):  
        super().__init__()  
  
        ...
```

self.initUI()

Создание GUI поручено методу `initUI()`.



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

Все три метода были унаследованы из класса **QWidget**.

setGeometry делает две вещи: он определяет место окна на экране и устанавливает его размер. Первые два параметра — позиции *x* и *y* нашего окна. Третий — ширина, и четвёртый — высота окна.

Фактически, **setGeometry** сочетает методы **resize()** и **move()** в одном. Последний метод устанавливает иконку приложения. Чтобы сделать это, мы создали объект **QIcon**. **QIcon** принимает путь к нашей иконке для её отображения.

Показ всплывающих подсказок

```
import sys
from PyQt5.QtWidgets import (QWidget, QToolTip, QPushButton, QApplication)
from PyQt5.QtGui import QFont
class Example(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        QToolTip.setFont(QFont('SansSerif', 10))
        self.setToolTip('This is a <b>QPushButton</b> widget')

        btn = QPushButton('Button', self)
        btn.setToolTip('This is a <b>QPushButton</b> widget')
        btn.resize(btn.sizeHint())
        btn.move(50, 50)

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

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

Мы можем обеспечить любой из наших виджетов всплывающей подсказкой.

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

Этот статический метод устанавливает шрифт, используемый для показа всплывающих подсказок. Мы используем шрифт 10px SansSerif.

```
self.setToolTip('This is a <b>QWidget</b> widget')
```

Чтобы создать подсказку, мы вызываем метод `setTooltip()`. Мы можем использовать HTML форматирование текста.

```
btn.setToolTip('This is a <b>QPushButton</b> widget')
```

Мы создаём виджет кнопки и устанавливаем всплывающую подсказку для неё.

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

Меняем размер у кнопки, перемещаем её в окно.

Метод **sizeHint()** даёт рекомендуемый размер для кнопки.



пример:

