

Библиотека PyQt.

Загружаем библиотеку и модули в программу:

```
from PyQt5.QtWidgets import
```

 (здесь перечисляем классы, которые будем использовать)

```
from PyQt5.QtCore import Qt
```

Классы:

QApplication()

каждое приложение с графическим интерфейсом
должно иметь экземпляр этого класса

QWidget()

базовый класс для объектов интерфейса

!!!Обязательно создаем эти два объекта в начале программы:

`app = QApplication([])` - приложение

`main_win = QWidget()` - окно

!!!обязательно пишем это в конце программы:

`app.exec_()` - метод (оставляет приложение открытым)

`main_win.show()`

Методы для настройки главного окна:

Метод	Назначение
<code>main_win.setWindowTitle('Название')</code>	Установить заголовок окна
<code>main_win.move(900, 70)</code>	Появление окна в указанной точке (а не по центру экрана)
<code>main_win.resize(400, 200)</code>	Изменение размеров окна
<code>main_win.show()</code>	Сделать окно видимым
<code>main_win.hide()</code>	Скрыть окно

Классы для создания виджетов:

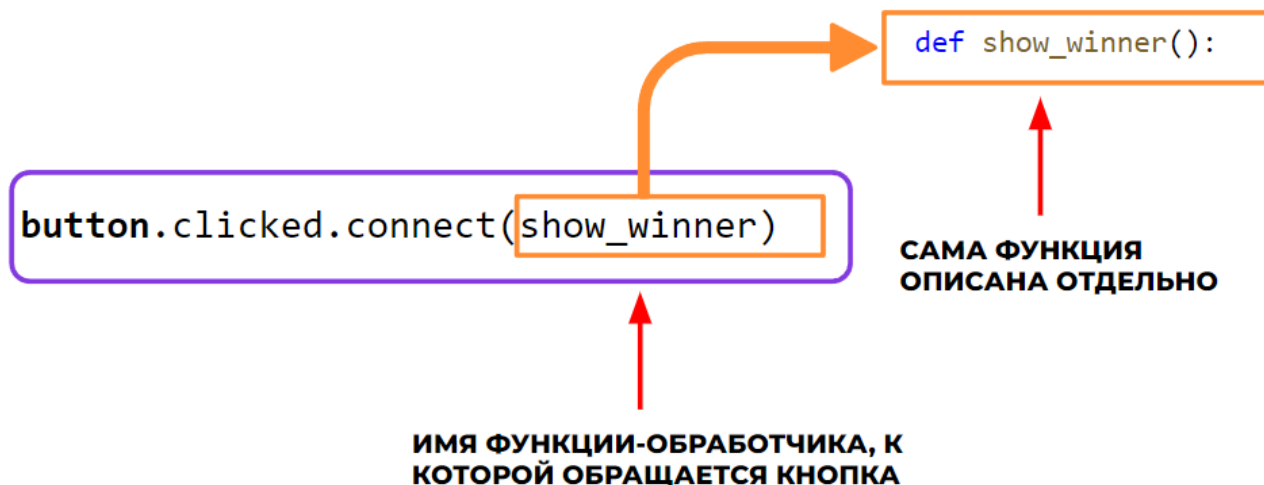
- **QLabel()** - текст или изображение

Метод	Назначение
<code>text = QLabel('тут ваша надпись')</code>	Конструктор , создающий объект типа «Надпись» с указанным текстом
<code>text.setText('34')</code>	Метод, изменяющий текст надписи

- **QPushButton()** - кнопка

Метод	Назначение
<code>btn = QPushButton('текст, который будет на кнопке')</code>	Конструктор , создающий объект типа «Кнопка» с пометкой

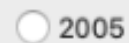
Чтобы при нажатии на кнопку что-то происходило, нужно написать функцию с подходящими командами и связать функцию с кнопкой специальной командой:



- **QMessageBox()** - окно с уведомлением

Метод	Назначение
<code>victory_win = QMessageBox()</code>	Конструктор , создающий окно уведомления.
<code>victory_win.setText('ваш текст')</code>	Метод , отображающий указанный текст в окне.
<code>victory_win.exec_()</code>	Оставить окно открытым.

- **QRadioButton()** - кнопка - переключатель



Метод	Назначение
<code>btn_answer = QRadioButton('2005')</code>	Конструктор , создающий объект типа «Переключатель» с подписью.

- **QListWidget()** - список элементов

Метод	Назначение
<code>list_tags = QListWidget()</code>	Конструктор для создания поля QListWidget для списка
<code>list_tags.addItem(['a', 'b'])</code>	Добавить список (строк) в список-виджет

<code>list_tags.clear()</code>	Очистить список QListWidget
<code>list_notes.itemClicked.connect</code> (имя функции)	*Использование метода в обработке событий (если выбран какой-то элемент списка, то что-то может происходить в программе)

- **QTextEdit()** - поле для ввода текста

Метод	Назначение
<code>field_text = QTextEdit()</code>	Конструктор для создания поля QTextEdit для ввода текста
<code>field_text.setText('Текст')</code>	Установить в поле текст, указанный в скобках

- **QLineEdit()** - поле ввода текста для запроса

Метод	Назначение
<code>field_tag = QLineEdit()</code>	Конструктор для создания поля QLineEdit для ввода текста
<code>field_tag.setPlaceholderText</code> (‘текст’)	Установить в поле текст, указанный в скобках

Линии(лэйауты): `QVBoxLayout()` `QHBoxLayout()`

на них мы “сажаем” виджеты

Метод	Назначение
<code>v_line = QVBoxLayout()</code> <code>h_line = QHBoxLayout()</code>	Конструктор , создающий объект типа «Вертикальная линия» и «Горизонтальная линия».
<code>v_line.addWidget(переменная с виджетом)</code>	Метод, добавляющий виджет к линии и располагающий по центру.
<code>main_line.addLayout(v_line)</code>	Добавить линию и её объекты на линию.
<code>main_win.setLayout(main_line)</code>	Установить получившуюся линию и её объекты в окно приложения.