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

Классы для создания виджетов:

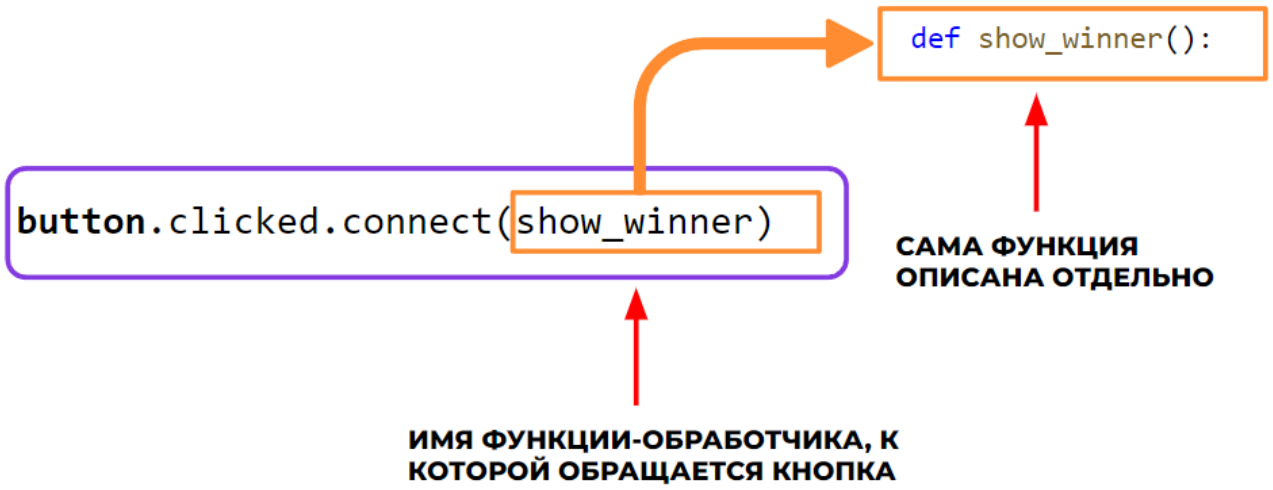
- **QLabel()** - текст или изображение

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

- **QPushButton()** - кнопка

Метод	*Назначение*
`btn = QPushButton('текст, который будет на кнопке')`	**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*
`victory_win = QMessageBox()`	**Конструктор**, создающий окно уведомления.
`victory_win.setText('ваш текст')`	**Метод,** отображающий указанный текст в окне.
`victory_win.exec_()`	Оставить окно открытым.

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

Метод	*Назначение*
`btn_answer = QRadioButton('2005')`	**Конструктор**, создающий объект типа «Переключатель» с подписью.

- **QListWidget()** - список элементов

Метод	*Назначение*
`list_tags = QListWidget()`	**Конструктор** для создания поля QListWidget для списка
`list_tags.addItems(['a', 'b'])`	Добавить список (строк) в список-виджет

`list_tags.clear()`	Очистить список QListWidget
`list_notes.itemClicked.connect (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*
`field_text = QTextEdit()`	Конструктор для создания поля QTextEdit для ввода текста
`field_text.setText('Текст')`	Установить в поле текст, указанный в скобках

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

Метод	*Назначение*
`field_tag = QLineEdit()`	Конструктор для создания поля QLineEdit для ввода текста
`field_tag.setPlaceholderText ('текст')`	Установить в поле текст, указанный в скобках

Линии(лэйауты): **QVBoxLayout()** **QHBoxLayout()**

на них мы “сажаем” виджеты

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