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()**  **QWidget()**

каждое приложение с графическим интерфейсом базовый класс для объектов интерфейса

должно иметь экземпляр этого класса

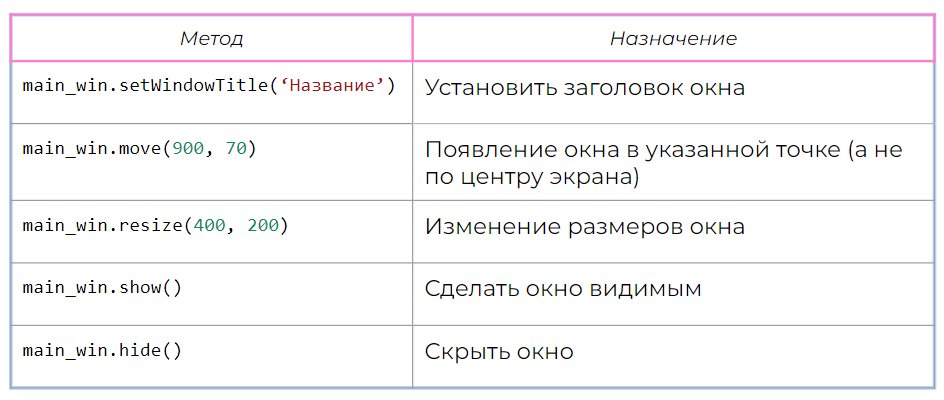
!!!Обязательно создаем эти два объекта в начале программы:

**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:**



**Классы для создания виджетов:**

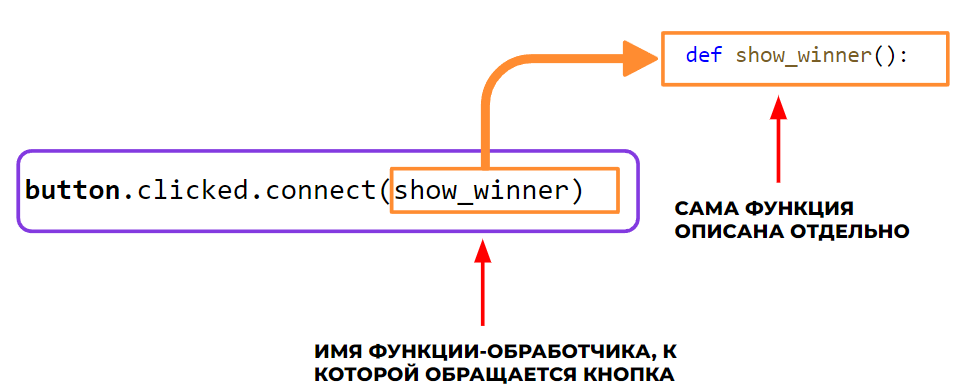
* **QLabel()** - текст или изображение

|  |  |
| --- | --- |
| *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. |