



PyQt 5

Разработка приложений. Первая лекция

Верещагина Анна Дмитриевна (Аня)

lorelei.aether@gmail.com

Qt, PyQt, Qt Designer

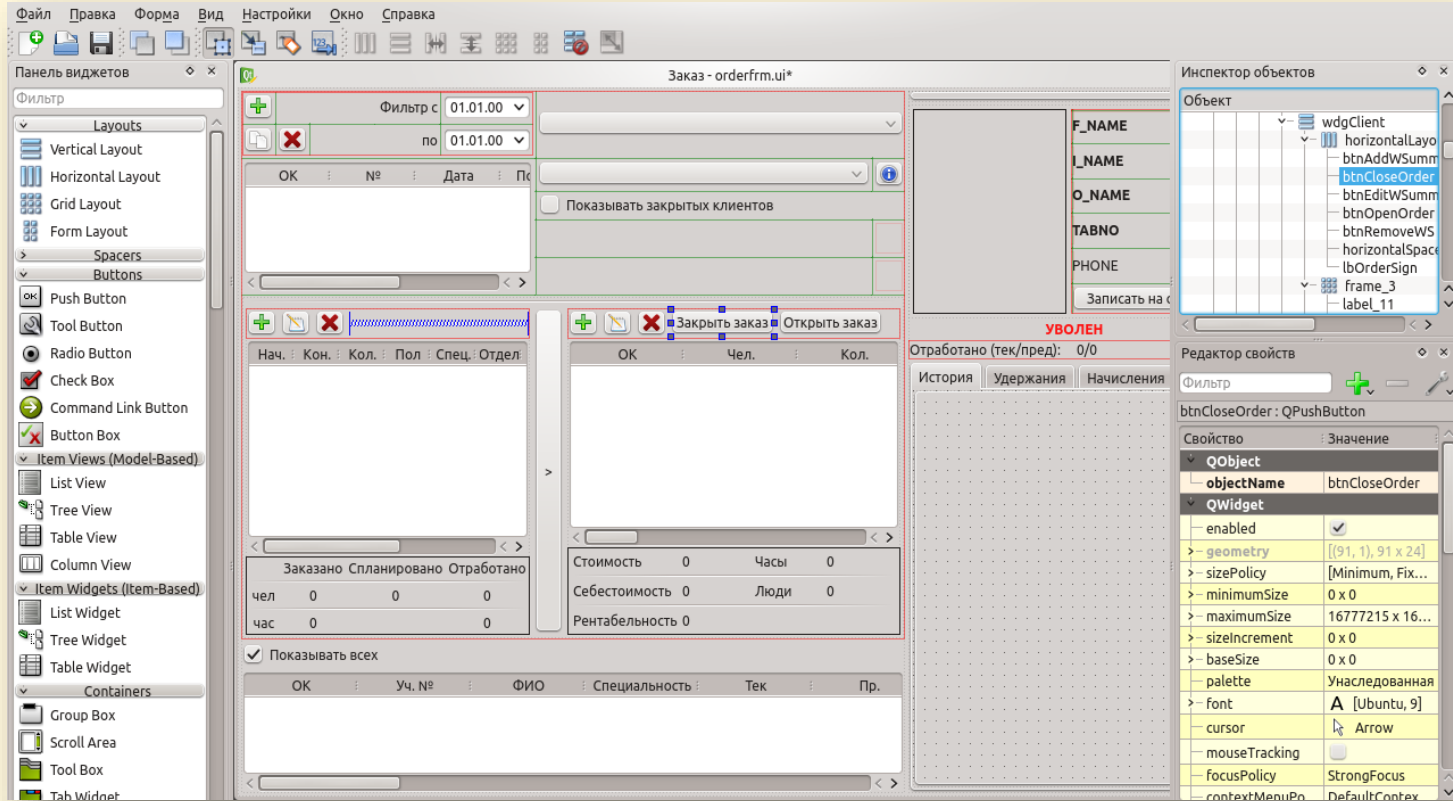
Qt – фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++.

PyQt – библиотека для Python, позволяющая использовать возможности Qt.

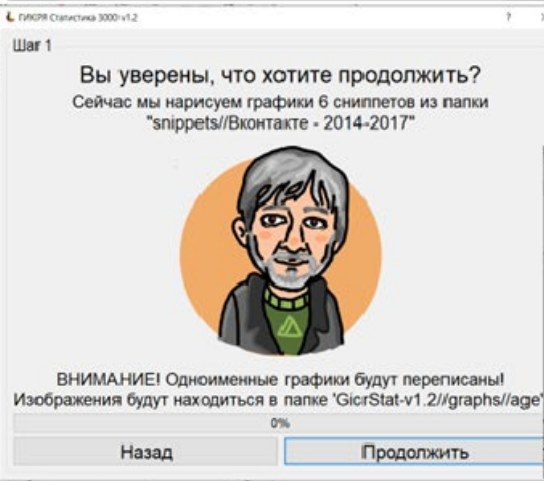
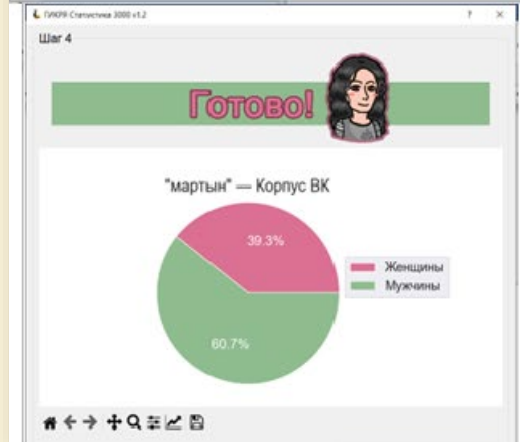
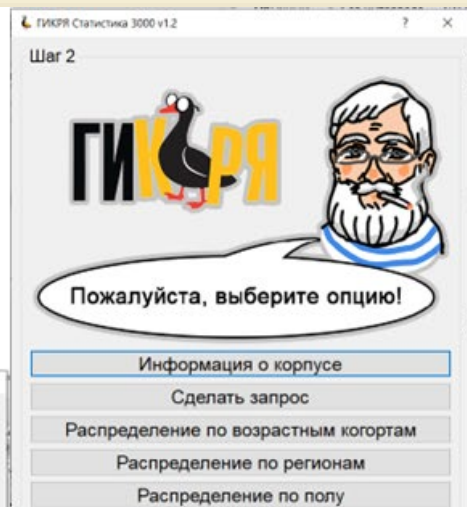
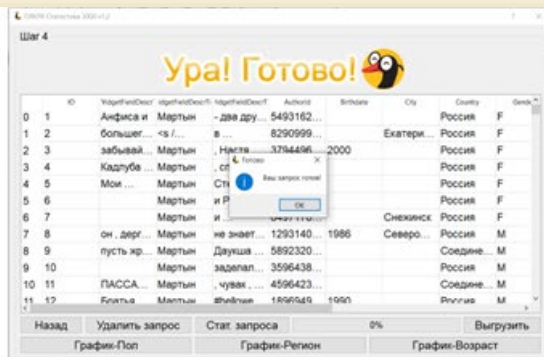
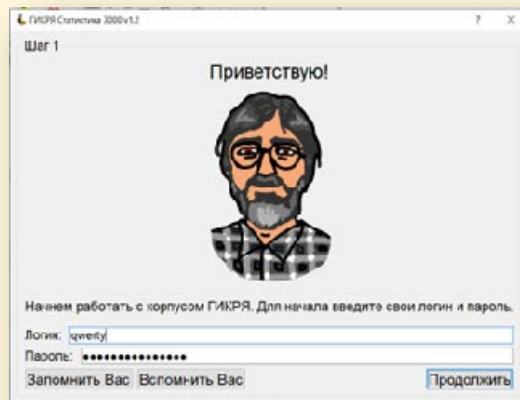
Qt Designer – среда для визуальной разработки (приложение-дизайнер) со своим интерфейсом для создания GUI с помощью ряда готовых инструментов. Файлы, созданные в Qt Designer, могут быть превращены в код Python.



Qt Designer




Гикря Статистика 3000



Гикря Статистика 3000 v1.2

Шаг 3

Актуальная статистика корпусов

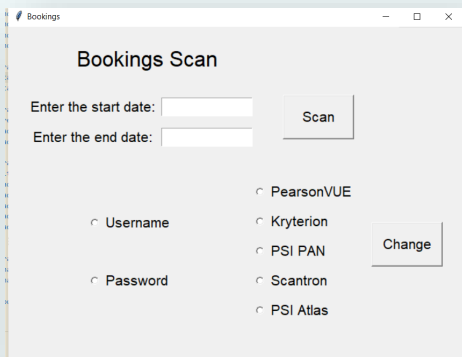


name	words	docs
1 Вконтакте - 2014-2017	5115640742	191570363
2 Журнальный Зап - 1990-2018	320139595	73900
3 Живой Журнал - 2001-2020	15987362740	354582844
4 Тестовый корпус ЖК	686757655	16626943

Назад Выгрузить

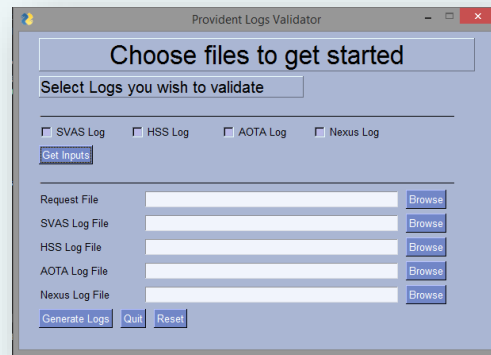
Основные альтернативы PyQt

tkinter



Один из первых графических фреймворков. PySimpleGui проще чем ткинтер, а PyQt – гибче и функциональнее.

PySimpleGUI



PySimpleGui – проще и интуитивнее, но он однооконный и не имеет зацепок для перехода на компилируемые языки. Есть опции интеграции с веб-страницами.



PyQt5 – основные модули

QtWidgets	основной модуль, содержащий классы всех доступных виджетов.
QtCore	основные не графические классы: система сигналов и слотов, платформонезависимые абстракции для Unicode, потоков, разделяемой памяти, регулярных выражений и т. д.
QtGui	компоненты графического интерфейса (элементы управления), основанные на визуальном представлении.
QtNetwork	классы для сетевого программирования. Например, клиентов и серверов через UDP и TCP.
QtOpenGL	классы, позволяющие использовать OpenGL и 3D-графику в приложениях PyQt.
QtScript	классы, позволяющие использовать встроенный в Qt интерпретатор JavaScript для управления приложением.
QtSql	классы для интеграции с базами данных с помощью SQL.
QtSvg	классы для отображения векторной графики в формате SVG.
QtXml	классы, реализующие обработку XML.
uic	реализация обработки XML-файлов, созданных в Qt Designer, для генерации из них Python-кода графического интерфейса.



Установка PyQt5



01

Откройте
командную строку

02

Введите
следующее:
`pip install PyQt5`

03

Дождитесь
установки
PyQt5 занимает
около 50 МБ



Основные окна GUI в PyQt5



QMainWindow

Содержит status-bar, menu-bar, tool-bar и основной виджет.



QWidget

Базовый класс всех объектов GUI.



QDialog


Для кратковременного общения с пользователем. Функции accept(), reject().

```
from PyQt5.QtWidgets import  
    QMainWindow, QWidget, QDialog
```





Построение окна в PyQt5



```
import sys
from PyQt5.QtWidgets import QApplication, QLabel, QVBoxLayout, QWidget

class Window1(QWidget): # Создаем класс от родителя QWidget - это будет наше первое окно.
    def __init__(self):
        super().__init__() # Наследуем методы и атрибуты класса QWidget.
        self.initUI()


    def initUI(self): # Основная функция для построения нашего окна.

        label1 = QLabel("Hello world!") # Создаем нашу надпись.

        layout = QVBoxLayout() # Вертикальный макет, в который мы будем класть наши виджеты.
        layout.addWidget(label1) # Добавляем виджет.
        self.setLayout(layout) # Устанавливаем наш макет в окно.

def open_window(): # Наша функция для открытия приложения.
    app = QApplication(sys.argv) # Создаем наше приложение с аргументами из командной строки.
    wind = Window1() # Создаем экземпляр первого окна.
    wind.show() # Показываем наше окно.
    sys.exit(app.exec_()) # Заканчиваем работу приложения в случае выхода.

open_window()
```



1 П С П С Н П С J

Версия для копирования:

```
import sys
from PyQt5.QtWidgets import QApplication, QLabel, QVBoxLayout, QWidget

class Window1(QWidget): # Создаем класс от родителя QWidget - это будет наше первое окно.
    def __init__(self):
        super().__init__() # Наследуем методы и атрибуты класса QWidget.
        self.initUI()

    def initUI(self): # Основная функция для построения нашего окна.

        label1 = QLabel("Hello world!") # Создаем нашу надпись.

        layout = QVBoxLayout() # Вертикальный макет, в который мы будем класть наши виджеты.
        layout.addWidget(label1) # Добавляем виджет.
        self.setLayout(layout) # Устанавливаем наш макет в окно.

def open_window(): # Наша функция для открытия приложения.
    app = QApplication(sys.argv) # Создаем наше приложение с аргументами из командной строки.
    wind = Window1() # Создаем экземпляр первого окна.
    wind.show() # Показываем наше окно.
    sys.exit(app.exec_()) # Заканчиваем работу приложения в случае выхода.

open_window()
```



Как сделать наше окно красивым?



```
self.setWindowTitle("Hello world")
```

Устанавливаем название нашему окну.

```
self.resize(500, 200)
```

Выбираем любой размер окна (ширина, высота)

```
self.setWindowIcon(QIcon('icon.png'))
```

Выбираем иконку для нашего окна. Размер иконки – примерно 48x48 пикселей.

```
self.setFont(QFont('Arial', 20))
```

Выбираем шрифт и его размер.



Построение окна в PyQt5




```
import sys
from PyQt5.QtWidgets import QApplication, QLabel, QVBoxLayout, QWidget
from PyQt5.QtGui import QFont, QIcon

class Window1(QWidget): # Создаем класс от родителя QWidget - это будет наше первое окно.
    def __init__(self):
        super().__init__() # Наследуем методы и атрибуты класса QWidget.

        self.setWindowTitle("Hello world") # Называем наше окно.
        self.resize(500, 200) # Назначаем размер окна.
        self.setWindowIcon(QIcon('icon.png')) # Выбираем иконку для нашего окна.
        self.setFont(QFont('Arial', 20)) # Назначаем шрифт и размер шрифта.

        self.initUI()
```



Версия для копирования:

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

```
class Window1(QWidget): # Создаем класс от родителя QWidget - это будет наше первое окно.  
    def __init__(self):  
        super().__init__() # Наследуем методы и атрибуты класса QWidget.
```

```
self.setWindowTitle("Hello world")# Называем наше окно.  
self.resize(500, 200) # Назначаем размер окна.  
self.setWindowIcon(QIcon('icon.png')) # Выбираем иконку для нашего окна.  
self.setFont(QFont('Arial', 20)) # Назначаем шрифт и размер шрифта.
```

Сохраните эту иконку в папку с Вашим кодом под названием «icon.png», и тогда она прикрепится к вашему окну! Можете использовать любую другую.





Основные виджеты PyQt5



QLabel

Метка. В нее можно поместить текст, картинку или анимацию.



QLineEdit

Поле для ввода.



QPushButton

Кнопка. Срабатывает в момент нажатия.



QCheckBox

Коробка с галочкой. Обычно служит как опция.



QRadioButton

Кнопка с точкой. Как правило служит для выбора из нескольких.



QProgressBar

Строка прогресса. Изменяется вручную в коде программы.



Основные виджеты PyQt5



QComboBox

Выпадающий список.
Служит для выбора из
большого кол-ва вариантов.



QTextEdit

Простейший
текстовый редактор.



QTableWidget

Виджет для
отображения таблиц.



QCalendar Widget

Виджет календаря.



QSlider

Слайдер. Служит для
выбора точки на шкале.



QSpinBox

Коробка со
стрелочками. Служит
для набора числа.





Основные методы виджетов (и для окон-виджетов)



`widget.show()`

Показать виджет.

`widget.hide()`

Спрятать виджет.

`widget.close()`

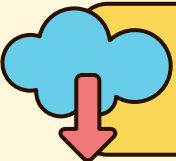
Заккрыть виджет.

`widget.setLayout(layout)`

Установить раскладку в виджет.

`widget.resize(ГлѢлĈ≥l ≠l ĚĈĚ≥)`

Назначить размер виджета.



Полезные методы виджетов:

`widget.setFont(QFont)`

Назначить шрифт виджету.
Для этого создайте QFont(«Arial», 32) для Вашего шрифта.

`widget.setDisabled(bool)`

«Отменить» виджет. Он будет неактивным.

`widget.setEnabled(bool)`

«Включить» виджет. Он будет активным.

`widget.setFixedSize(ГлѢт≠| ѐт)`

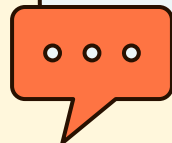
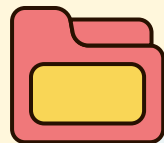
Еще один способ назначить размер виджету.

`widget.setStyleSheet(string)`

Установить стиль виджету. Можно выбрать цвет текста и фона. Смотреть документацию!

`widget.setCursor(QCursor)`

Выбрать стиль курсора при наведении на виджет. Например, можно вместо стрелочки поставить палец!



QLabel

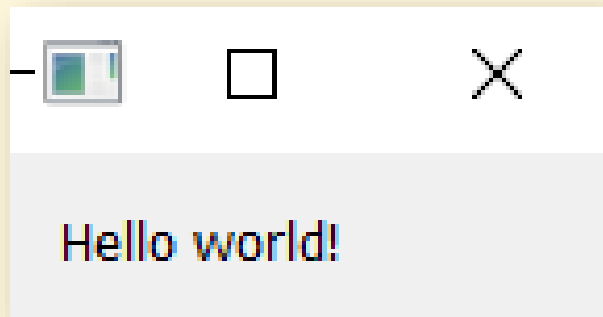
```
from PyQt5.QtWidgets import QLabel
```

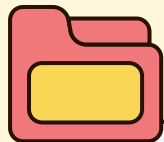
```
label = QLabel(«Hello World!»)
```

* Зеленым отмечено то, что можно изменить по своему усмотрению!



OUTPUT:





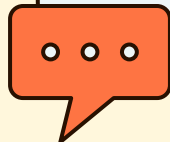
QLabel как изображение

+ from PyQt5.QtWidgets import QLabel
from PyQt5.QtGui import QPixmap

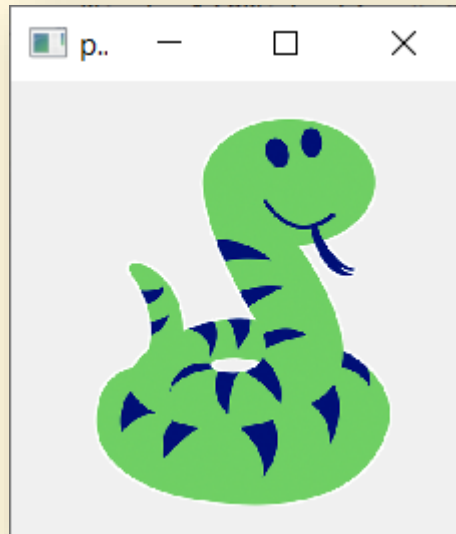
pic = QLabel()
pic.setPixmap(QPixmap("image.png").scaled(200, 201))



* Голубым отмечено то, что не обязательно использовать!



OUTPUT:



Выравнивание

Влево

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

Qt.AlignLeft

```
label.setAlignment(  
    Qt.AlignLeft)
```

Посередине

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

Qt.AlignCenter

```
label.setAlignment(  
    Qt.AlignCenter)
```

Вправо

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

Qt.AlignRight

```
label.setAlignment(  
    Qt.AlignRight)
```



Надпись и изображение в PyQt5



```
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtGui import QPixmap, QFont, QIcon
from PyQt5.QtCore import Qt
```

```
class Window1(QWidget): # Создаем класс от родителя QWidget - это будет наше первое окно.
    def __init__(self):
        super().__init__() # Наследуем методы и атрибуты класса QWidget.
```

```
        self.setWindowTitle("Hello world") # Называем наше окно.
        self.resize(500, 200) # Назначаем размер окна.
        self.setWindowIcon(QIcon('icon.png')) # Выбираем иконку для нашего окна.
        self.setFont(QFont('Arial', 20)) # Назначаем шрифт и размер шрифта.
```

```
        self.initUI()
```

```
    def initUI(self): # Основная функция для построения нашего окна.
```

```
        label1 = QLabel("Hello world!") # Создаем нашу надпись.
        label1.setAlignment(Qt.AlignCenter) # Выравниваем по центру.
        pic = QLabel() # Создаем надпись, которую потом заполним изображением.
        pic.setPixmap(QPixmap("image.png").scaled(200, 201)) # Устанавливаем изображение и подгоняем его размер.
        pic.setAlignment(Qt.AlignCenter) # Выравниваем по центру.
```

```
        layout = QVBoxLayout() # Вертикальный макет, в который мы будем класть наши виджеты.
        layout.addWidget(label1) # Добавляем виджет.
        layout.addWidget(pic)
        self.setLayout(layout) # Устанавливаем наш макет в окно.
```



Версия для копирования:

```
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtGui import QPixmap, QFont, QIcon
from PyQt5.QtCore import Qt
```

```
class Window1(QWidget): # Создаем класс от родителя QWidget - это будет наше первое окно.
    def __init__(self):
        super().__init__() # Наследуем методы и атрибуты класса QWidget.

        self.setWindowTitle("Hello world") # Называем наше окно.
        self.resize(500, 200) # Назначаем размер окна.
        self.setWindowIcon(QIcon('icon.png')) # Выбираем иконку для нашего окна.
        self.setFont(QFont('Arial', 20)) # Назначаем шрифт и размер шрифта.

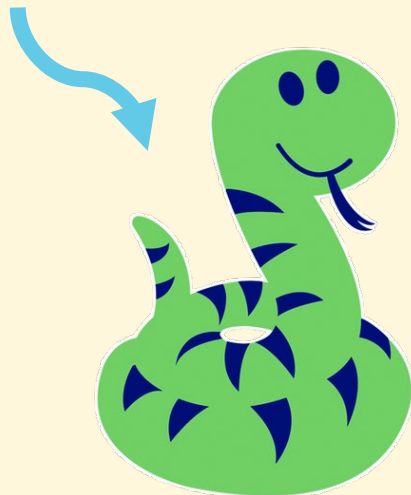
        self.initUI()

    def initUI(self): # Основная функция для построения нашего окна.
```

```
        label1 = QLabel("Hello world!") # Создаем нашу надпись.
        label1.setAlignment(Qt.AlignCenter) # Выравниваем по центру.
        pic = QLabel() # Создаем надпись, которую потом заполним изображением.
        pic.setPixmap(QPixmap("image.png").scaled(200, 201)) # Устанавливаем изображение и подгоняем его размер.
        pic.setAlignment(Qt.AlignCenter) # Выравниваем по центру.
```

```
        layout = QVBoxLayout() # Вертикальный макет, в который мы будем класть наши виджеты.
        layout.addWidget(label1) # Добавляем виджет.
        layout.addWidget(pic) # Добавляем картинку.
        self.setLayout(layout) # Устанавливаем наш макет в окно.
```

Сохраните эту картинку в папку с Вашим кодом под названием «image.png», и тогда она появится! Можете использовать любую другую.



Output:





Полезные методы QLabel:

label.text()

Получить текст (строку) из QLabel.

label.setText(string)

Установить текст (строку) в QLabel.

label.clear()

Очистить QLabel.

label.setPixmap(QPixmap)

Установить картинку в QLabel.
Для этого создайте QPixmap("image.png") для Вашего изображения.

label.setMovie(QMovie)

Установить анимацию в QLabel.
Для этого создайте QMovie("animation.gif") для Вашей анимации.



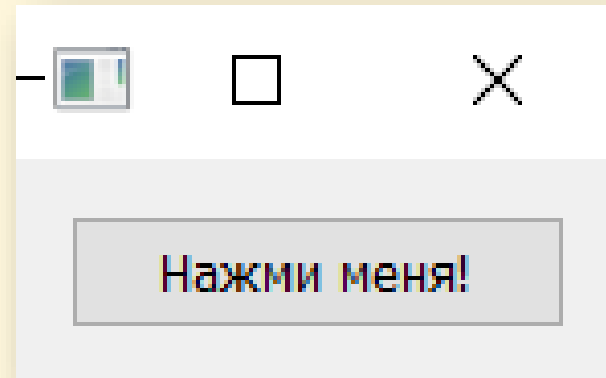
QPushButton

```
from PyQt5.QtWidgets import  
QPushButton
```

```
button = QPushButton(«Нажми меня!»)
```



OUTPUT:





Кнопка в PyQt5



```
def initUI(self): # Основная функция для построения нашего окна.

    labell = QLabel("Hello world!") # Создаем нашу надпись.
    labell.setAlignment(Qt.AlignCenter) # Выравниваем по центру.
    pic = QLabel() # Создаем надпись, которую потом заполним изображением.
    pic.setPixmap(QPixmap("image.png").scaled(200, 201)) # Устанавливаем изображение и подгоняем его размер.
    pic.setAlignment(Qt.AlignCenter) # Выравниваем по центру.
    button = QPushButton("Нажми меня!") # Создаем кнопку.

    layout = QVBoxLayout() # Вертикальный макет, в который мы будем класть наши виджеты.
    layout.addWidget(labell) # Добавляем надпись.
    layout.addWidget(pic) # Добавляем картинку.
    layout.addWidget(button) # Добавляем кнопку.
    self.setLayout(layout) # Устанавливаем наш макет в окно.
```



Версия для копирования:

```
def initUI(self): # Основная функция для построения нашего окна.
```

```
    label1 = QLabel("Hello world!") # Создаем нашу надпись.
```

```
    label1.setAlignment(Qt.AlignCenter)# Выравниваем по центру.
```

```
    pic = QLabel() # Создаем надпись, которую потом заполним изображением.
```

```
    pic.setPixmap(QPixmap("image.png").scaled(200, 201)) # Устанавливаем изображение и подгоняем его размер.
```

```
    pic.setAlignment(Qt.AlignCenter)# Выравниваем по центру.
```

```
    button = QPushButton("Нажми меня!") # Создаем кнопку.
```

```
    layout = QVBoxLayout()# Вертикальный макет, в который мы будем класть наши виджеты.
```

```
    layout.addWidget(label1) # Добавляем надпись.
```

```
    layout.addWidget(pic) # Добавляем картинку.
```

```
    layout.addWidget(button) # Добавляем кнопку.
```

```
    self.setLayout(layout) # Устанавливаем наш макет в окно.
```



QPushButton – привязка к функции

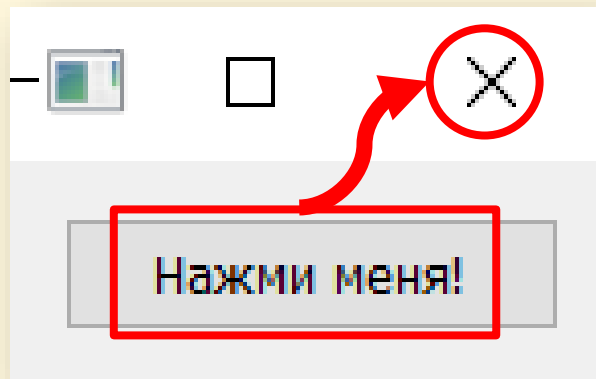
`button.clicked.connect(self.close)`
Это может быть любая другая функция!

В данном случае метод `.close` уже существует, мы наследовали его от класса `QWidget`.

Важно: мы не можем задавать подключаемой функции аргументы (и никогда не ставим скобки после нее)



OUTPUT:



Версия для копирования:

```
def initUI(self): # Основная функция для построения нашего окна.
```

```
    label1 = QLabel("Hello world!") # Создаем нашу надпись.
```

```
    label1.setAlignment(Qt.AlignCenter)# Выравниваем по центру.
```

```
    pic = QLabel() # Создаем надпись, которую потом заполним изображением.
```

```
    pic.setPixmap(QPixmap("image.png").scaled(200, 201)) # Устанавливаем изображение и подгоняем его размер.
```

```
    pic.setAlignment(Qt.AlignCenter)# Выравниваем по центру.
```

```
    button = QPushButton("Нажми меня!") # Создаем кнопку.
```

```
    button.clicked.connect(self.close) # Привязываем кнопку к методу "закреть".
```

```
    layout = QVBoxLayout()# Вертикальный макет, в который мы будем класть наши виджеты.
```

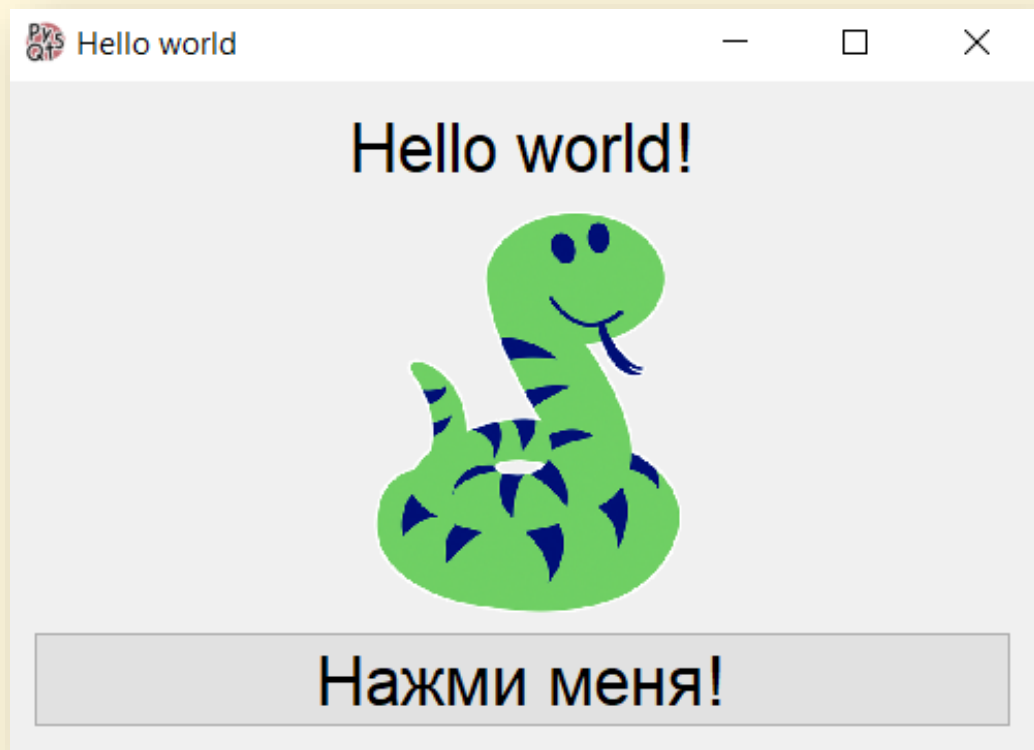
```
    layout.addWidget(label1) # Добавляем надпись.
```

```
    layout.addWidget(pic) # Добавляем картинку.
```

```
    layout.addWidget(button) # Добавляем кнопку.
```

```
    self.setLayout(layout) # Устанавливаем наш макет в окно.
```

Output:





Полезные методы QPushButton:

button.text()

Получить текст (строку) из QPushButton.

button.setText(string)

Установить текст (строку) в QPushButton.

button.setAutoDefault(bool)

Сделать кнопку предпочитаемой по умолчанию (срабатывает на пробел). Полезно, когда в окне есть более одной кнопки.

button.clicked.connect(def)

Связать кнопку с функцией.



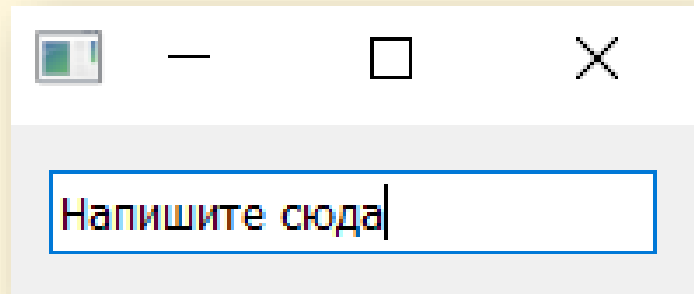
QLineEdit

```
from PyQt5.QtWidgets import QLineEdit
```

```
lineEdit = QLineEdit(«Напишите сюда»)
```



OUTPUT:



Версия для копирования:

```
def initUI(self): # Основная функция для построения нашего окна.
```

```
    label1 = QLabel("Hello world!") # Создаем нашу надпись.
```

```
    label1.setAlignment(Qt.AlignCenter)# Выравниваем по центру.
```

```
    pic = QLabel() # Создаем надпись, которую потом заполним изображением.
```

```
    pic.setPixmap(QPixmap("image.png").scaled(200, 201)) # Устанавливаем изображение и подгоняем его размер.
```

```
    pic.setAlignment(Qt.AlignCenter)# Выравниваем по центру.
```

```
    button = QPushButton("Нажми меня!") # Создаем кнопку.
```

```
    button.clicked.connect(self.close) # Привязываем кнопку к методу "закрыть".
```

```
    linedit = QLineEdit("Напишите сюда") # Создаем поле для ввода.
```

```
    layout = QVBoxLayout()# Вертикальный макет, в который мы будем класть наши виджеты.
```

```
    layout.addWidget(label1) # Добавляем надпись.
```

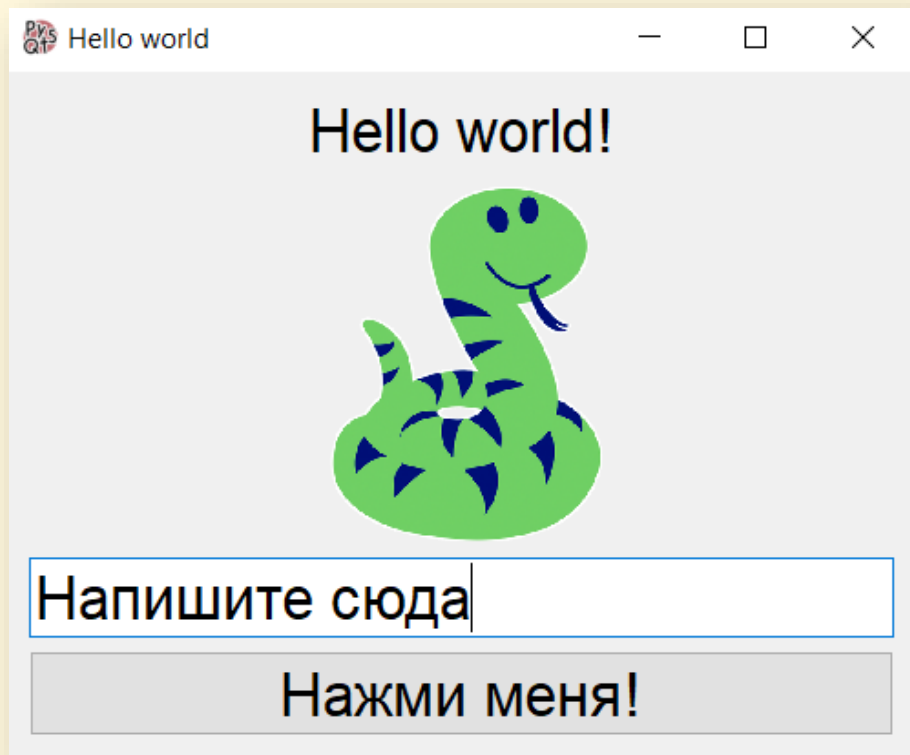
```
    layout.addWidget(pic) # Добавляем картинку.
```

```
    layout.addWidget(linedit) # Добавляем поле для ввода.
```

```
    layout.addWidget(button) # Добавляем кнопку.
```

```
    self.setLayout(layout) # Устанавливаем наш макет в окно.
```

Output:





Полезные методы QLineEdit:



lineEdit.text()

Получить введенный текст (строку) из QLineEdit.

lineEdit.setText(string)

Установить текст (строку) в QLineEdit.

lineEdit.clear()

Очистить QLineEdit.

lineEdit.setEchoMode()

Спрятать текст при вводе в QLineEdit. Чтобы спрятать его за звездочками, нужно ввести в скобки QLineEdit.Password.

lineEdit.setCursorPosition(int)

Установить позицию курсора (палочки) по умолчанию в QLineEdit.



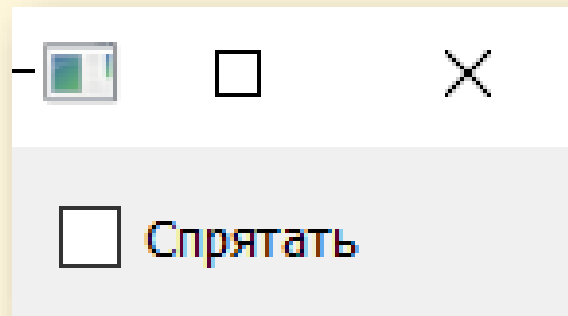
QCheckBox

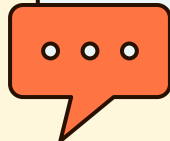
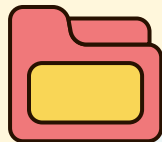
```
from PyQt5.QtWidgets import QCheckBox
```

```
checkbox = QCheckBox(«Спрятать»)
```



OUTPUT:





QCheckBox – Привязка к функции

```
checkbox.toggled.connect(lineedit.setEchoMode)
```

Опять же, любая функция!

Теперь при отметке чекбокса текст в нашей строке будет скрыт. А если убрать галочку, то будет показан снова!



OUTPUT:

Напишите сюда

☐ Спрятать



☒ Спрятать



Версия для копирования:

```
def initUI(self): # Основная функция для построения нашего окна.
```

```
    label1 = QLabel("Hello world!") # Создаем нашу надпись.
```

```
    label1.setAlignment(Qt.AlignCenter) # Выравниваем по центру.
```

```
    pic = QLabel() # Создаем надпись, которую потом заполним изображением.
```

```
    pic.setPixmap(QPixmap("image.png").scaled(200, 201)) # Устанавливаем изображение и подгоняем его размер.
```

```
    pic.setAlignment(Qt.AlignCenter) # Выравниваем по центру.
```

```
    button = QPushButton("Нажми меня!") # Создаем кнопку.
```

```
    button.clicked.connect(self.close) # Привязываем кнопку к методу "закрыть".
```

```
    lineedit = QLineEdit("Напишите сюда") # Создаем поле для ввода
```

```
    checkbox = QCheckBox("Спрятать") # Создаем чек-бокс.
```

```
    checkbox.toggled.connect(lineedit.setEchoMode) # Привязываем к методу lineedit "спрятать".
```

```
    layout = QVBoxLayout() # Вертикальный макет, в который мы будем класть наши виджеты.
```

```
    layout.addWidget(label1) # Добавляем надпись.
```

```
    layout.addWidget(pic) # Добавляем картинку.
```

```
    layout.addWidget(lineedit) # Добавляем поле для ввода.
```

```
    layout.addWidget(checkbox) # Добавляем чек-бокс.
```

```
    layout.addWidget(button) # Добавляем кнопку.
```

```
    self.setLayout(layout) # Устанавливаем наш макет в окно.
```



Полезные методы QCheckBox:



checkbox.text()

Получить текст (строку) из QCheckBox.

checkbox.setText(string)

Установить текст (строку) в QCheckBox.

checkbox.isChecked()

Проверка: подчеркнут ли QCheckBox.
Возвращает bool.

checkbox.setChecked(bool)

Сделать QCheckBox подчеркнутым
изначально.

checkbox.toggled.connect(def)

Связать QCheckBox с функцией.

Раскладки

QVBoxLayout

вертикальная
V - Vertical

Widget1

Widget2

Widget3

Widget1	Widget3	Widget5
Widget2	Widget4	Widget6

QGridLayout

сетка

QHBoxLayout

горизонтальная
H - Horizontal

Widget1

Widget2

Widget3

QGroupBox

группа (является виджетом)



QVBoxLayout

Вертикальная раскладка

```
from PyQt5.QtWidgets import  
QVBoxLayout
```

```
layout = QVBoxLayout()
```

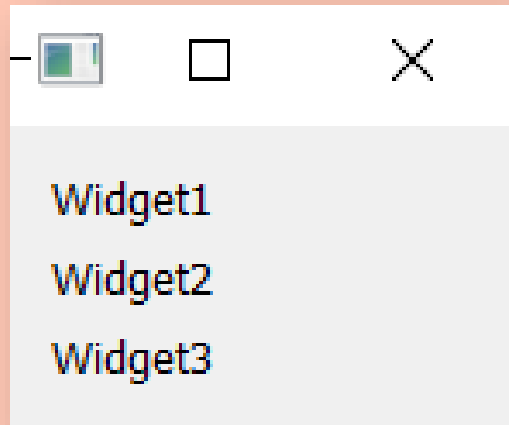
```
layout.addWidget(label1)
```

```
layout.addWidget(label2)
```

```
layout.addWidget(label3)
```



Output:





QHBoxLayout

Горизонтальная раскладка

```
from PyQt5.QtWidgets import  
QHBoxLayout
```

```
layout = QHBoxLayout()
```

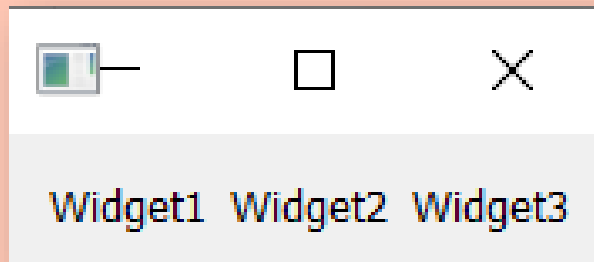
```
layout.addWidget(label1)
```

```
layout.addWidget(label2)
```

```
layout.addWidget(label3)
```



Output:



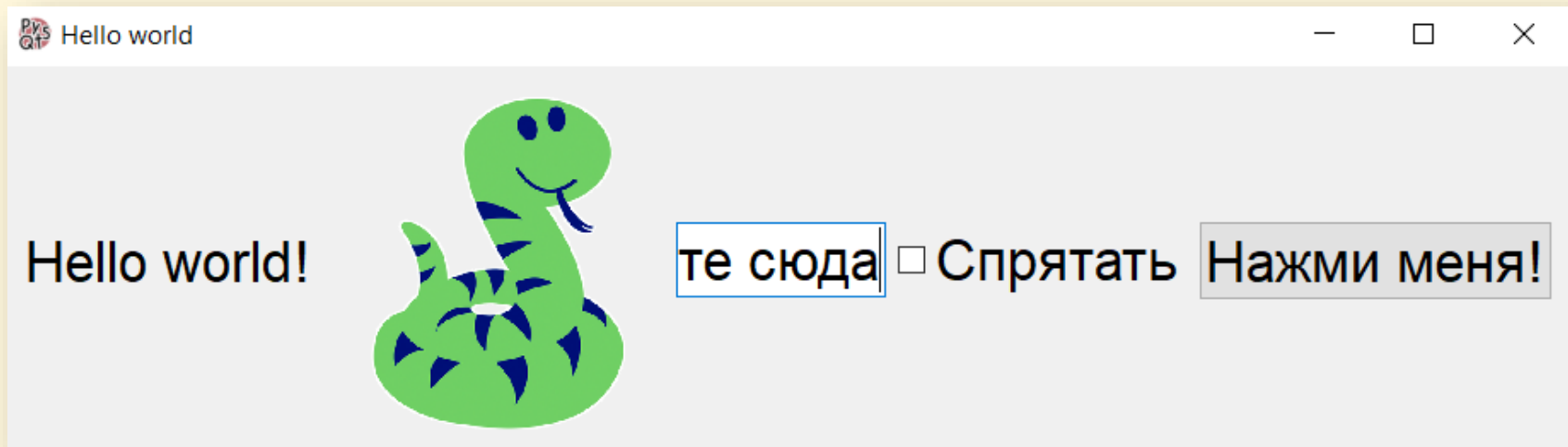
Версия для копирования:

```
def initUI(self): # Основная функция для построения нашего окна.
```

```
    label1 = QLabel("Hello world!") # Создаем нашу надпись.  
    label1.setAlignment(Qt.AlignCenter) # Выравниваем по центру.  
    pic = QLabel() # Создаем надпись, которую потом заполним изображением.  
    pic.setPixmap(QPixmap("image.png").scaled(200, 201)) # Устанавливаем изображение и подгоняем его размер.  
    pic.setAlignment(Qt.AlignCenter) # Выравниваем по центру.  
    button = QPushButton("Нажми меня!") # Создаем кнопку.  
    button.clicked.connect(self.close) # Привязываем кнопку к методу "закрыть".  
    linedit = QLineEdit("Напишите сюда") # Создаем поле для ввода  
    checkbox = QCheckBox("Спрятать") # Создаем чек-бокс.  
    checkbox.toggled.connect(linedit.setEchoMode) # Привязываем к методу linedit "спрятать".
```

```
    layout = QHBoxLayout() # Горизонтальный макет, в который мы будем класть наши виджеты.  
    layout.addWidget(label1) # Добавляем надпись.  
    layout.addWidget(pic) # Добавляем картинку.  
    layout.addWidget(linedit) # Добавляем поле для ввода.  
    layout.addWidget(checkbox) # Добавляем чек-бокс.  
    layout.addWidget(button) # Добавляем кнопку.  
    self.setLayout(layout) # Устанавливаем наш макет в окно.
```

Output:



QGridLayout

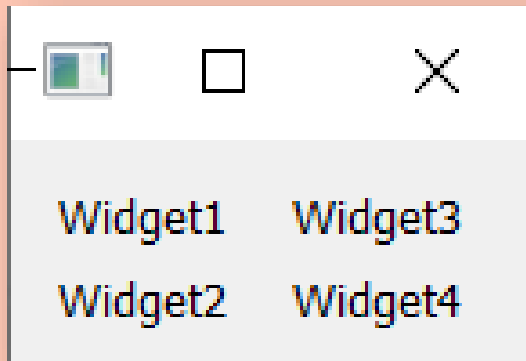
Раскладка-сетка

```
from PyQt5.QtWidgets import  
QGridLayout
```

```
layout = QGridLayout()
```

```
layout.addWidget(label1, 0, 0)  
layout.addWidget(label2, 1, 0)  
layout.addWidget(label3, 0, 1)  
layout.addWidget(label4, 1, 1)
```

Output:



Здесь мы указываем
координаты по x и y

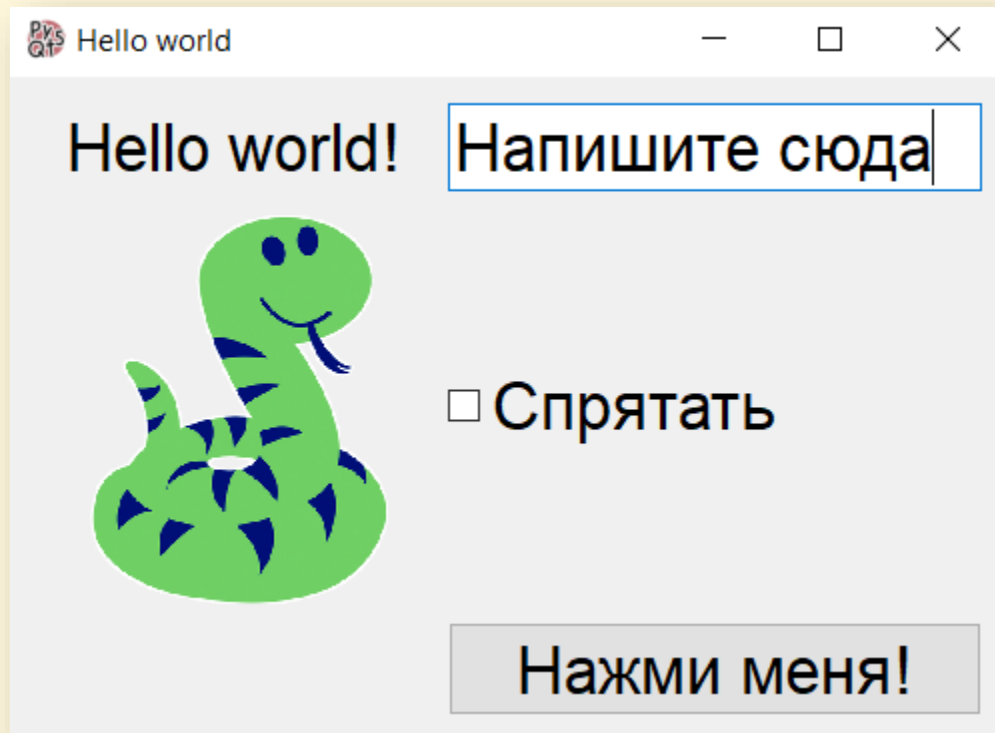
Версия для копирования:

```
def initUI(self): # Основная функция для построения нашего окна.
```

```
    label1 = QLabel("Hello world!") # Создаем нашу надпись.  
    label1.setAlignment(Qt.AlignCenter) # Выравниваем по центру.  
    pic = QLabel() # Создаем надпись, которую потом заполним изображением.  
    pic.setPixmap(QPixmap("image.png").scaled(200, 201)) # Устанавливаем изображение и подгоняем его размер.  
    pic.setAlignment(Qt.AlignCenter) # Выравниваем по центру.  
    button = QPushButton("Нажми меня!") # Создаем кнопку.  
    button.clicked.connect(self.close) # Привязываем кнопку к методу "закрыть".  
    lineedit = QLineEdit("Напишите сюда") # Создаем поле для ввода  
    checkbox = QCheckBox("Спрятать") # Создаем чек-бокс  
    checkbox.toggled.connect(lineedit.setEchoMode) # Привязываем к методу lineedit "спрятать"
```

```
    layout = QGridLayout() # Сетка.  
    layout.addWidget(label1, 0, 0) # Добавляем надпись.  
    layout.addWidget(pic, 1, 0) # Добавляем картинку.  
    layout.addWidget(lineedit, 0, 1) # Добавляем поле для ввода.  
    layout.addWidget(checkbox, 1, 1) # Добавляем чек-бокс.  
    layout.addWidget(button, 2, 1) # Добавляем кнопку.  
    self.setLayout(layout) # Устанавливаем наш макет в окно.
```

Output:






QGroupBox

Группа

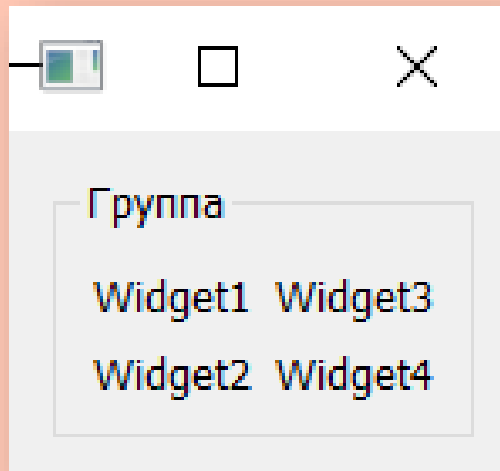
```
from PyQt5.QtWidgets import  
QGroupBox
```

```
mainlayout = QVBoxLayout()  
groupbox = QGroupBox("Группа")  
groupbox.setLayout(Ваш layout)  
mainlayout.addWidget(groupbox)
```

Объект QGroupBox является не layout'ом, а QWidget'ом!



Output:



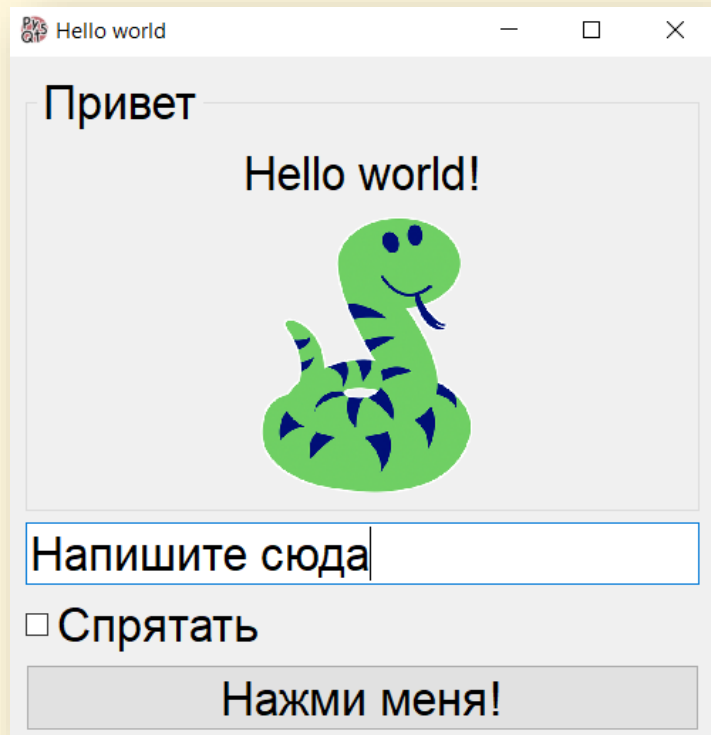
Версия для копирования:

```
def initUI(self): # Основная функция для построения нашего окна.
```

```
    label1 = QLabel("Hello world!") # Создаем нашу надпись.  
    label1.setAlignment(Qt.AlignCenter)# Выравниваем по центру.  
    pic = QLabel() # Создаем надпись, которую потом заполним изображением.  
    pic.setPixmap(QPixmap("image.png").scaled(200, 201)) # Устанавливаем изображение и подгоняем его размер.  
    pic.setAlignment(Qt.AlignCenter)# Выравниваем по центру.  
    button = QPushButton("Нажми меня!") # Создаем кнопку.  
    button.clicked.connect(self.close) # Привязываем кнопку к методу "закрыть".  
    linedit = QLineEdit("Напишите сюда")# Создаем поле для ввода  
    checkbox = QCheckBox("Спрятать") # Создаем чек-бокс  
    checkbox.toggled.connect(linedit.setEchoMode) # Привязываем к методу linedit "спрятать"
```

```
    groupbox = QGroupBox("Привет") # Создаем нашу группу.  
    layout2 = QVBoxLayout()# Создаем дополнительную раскладку для того, чтобы сделать ее группой потом.  
    layout = QVBoxLayout()# Вертикальная раскладка, в которую мы будем класть наши виджеты.  
    layout2.addWidget(label1) # Добавляем надпись в раскладку.  
    layout2.addWidget(pic) # Добавляем картинку в раскладку.  
    groupbox.setLayout(layout2) # Устанавливаем нашу раскладку в группу.  
    layout.addWidget(groupbox) # Добавляем группу как виджет.  
    layout.addWidget(linedit) # Добавляем поле для ввода  
    layout.addWidget(checkbox) # Добавляем чек-бокс  
    layout.addWidget(button) # Добавляем кнопку.  
    self.setLayout(layout) # Устанавливаем наш макет в окно.
```

Output:



PyQt5 – мистическая ошибка

```
Process finished with exit code -1073740791 (0xC0000409)
```

Распространенная ошибка интерпретатора (появляется в PyCharm, IDLE просто вылетает), всплывает из-за ошибки в коде. Возникает периодически на более сложных уровнях программы. Нет никакой подсказки о том, что не так – программа просто прекращает работу.

Что делать???

- **Искать баг вручную.** К сожалению, **Stack Overflow** и другие ресурсы Вам **не помогут** – ошибка слишком распространенная, ее даже иногда относят к **багам PyQt5**.
- Используйте **print()** в коде – в тех местах, в которых может оказаться ошибка. Если перед какой-то из печатей **программа прерывается** – значит проблема в **этой строке**.
- Проверяйте наличие или отсутствие скобок, проверяйте места, где Вы собираете раскладки. Например, если Вы кладете **виджет в раскладку** – должен присутствовать метод **.addWidget**, а не **.addLayout**. Если Вы не можете найти ошибку – **обращайтесь ко мне за помощью!**

PyQt5 – мистическая ошибка

Распространенные причины

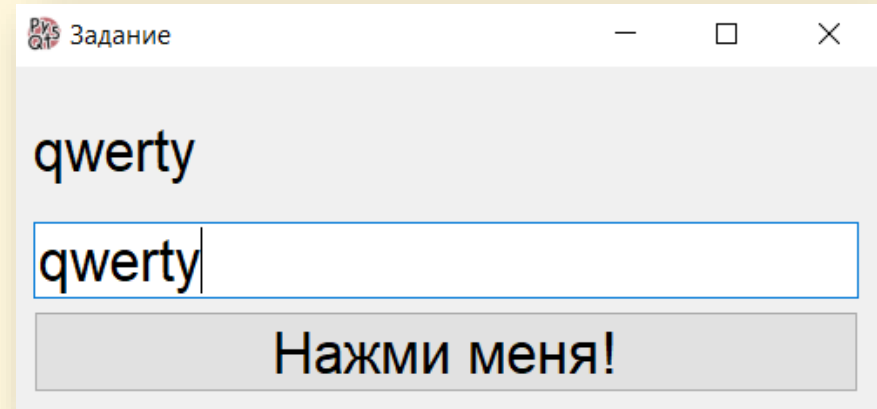
1. Создание экземпляра класса без скобок: `box = QHBoxLayout` вместо `box = QHBoxLayout()`.
2. Подключение функции со скобками: `button.toggled.connect(combobox.setEnabled())`.
3. Передача недостаточного или излишнего количества аргументов в класс нового окна.
4. Вы перепутали виджет с раскладкой. `layout.addWidget` VS `layout.addLayout`.
5. Вы не передали координаты при построении `QGridLayout`. `grid.addWidget(button)` вместо `grid.addWidget(button, x, y)`
6. Вы использовали на каком-то классе метод чужого класса. Например, у `QPushButton` нет метода `.setAlignment()`.
7. Если в методе построения окна (наш `initUI`) нет ошибок, то ошибка может скрываться в подключаемой и созданной Вами функции в классе этого окна.
8. Вы создаете слишком много всплывающих виджетов в другой функции (не `initUI`). Если Вам нужно маленькое дополнительное окошко, которое содержало бы **один-два виджета**, все равно стоит создать для него **отдельное окно**.

Задание

Попробуйте сделать свою программу любого вида и дизайна!

1. Создайте окно `QWidget`, используя шаблон, данный на слайде 10.
2. Создайте `QPushButton`, `QLineEdit` и `QLabel` (можете добавить картинку).
3. Свяжите виджеты с помощью функции: пусть при нажатии на `QPushButton` текст, введенный в `QLineEdit` появляется в `QLabel`. Выглядеть это должно примерно так:

Чтобы это реализовать, Вам нужно будет создать свою функцию в классе Вашего окна и подключить ее к кнопке. Превратите виджеты в атрибуты класса (добавляйте **self.**), чтобы к ним был доступ в новой функции!





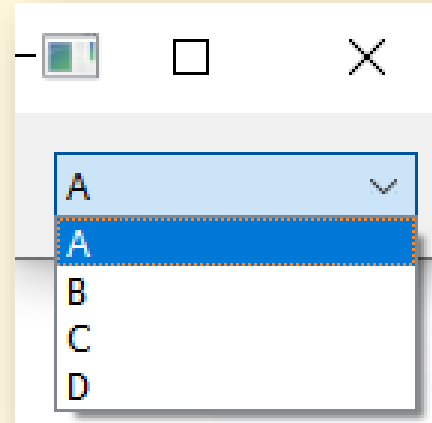
QComboBox

```
from PyQt5.QtWidgets import QComboBox
```

```
combobox = QComboBox()  
combobox.addItem("A", "B", "C", "D")
```



OUTPUT:



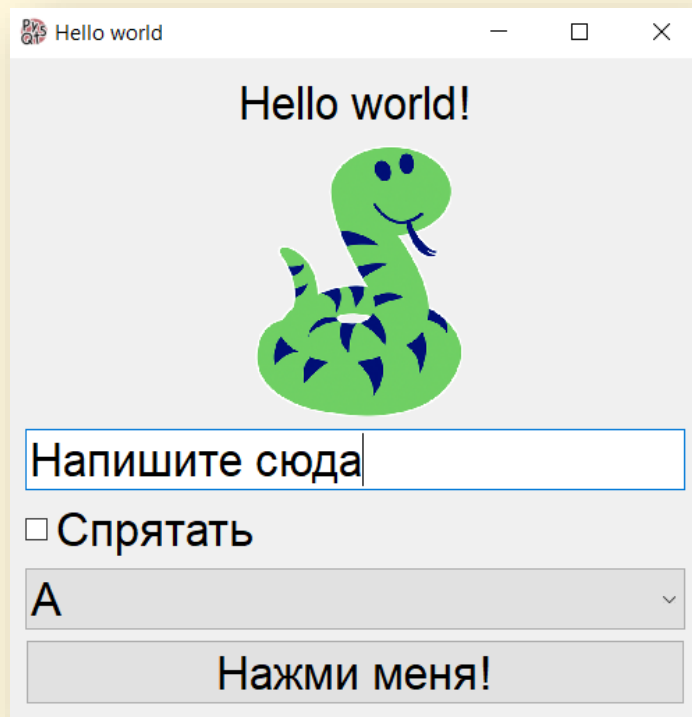
Версия для копирования:

```
def initUI(self): # Основная функция для построения нашего окна.

    label1 = QLabel("Hello world!") # Создаем нашу надпись.
    label1.setAlignment(Qt.AlignCenter) # Выравниваем по центру.
    pic = QLabel() # Создаем надпись, которую потом заполним изображением.
    pic.setPixmap(QPixmap("image.png").scaled(200, 201)) # Устанавливаем изображение и подгоняем его размер.
    pic.setAlignment(Qt.AlignCenter) # Выравниваем по центру.
    button = QPushButton("Нажми меня!") # Создаем кнопку.
    button.clicked.connect(self.close) # Привязываем кнопку к методу "закрыть".
    linedit = QLineEdit("Напишите сюда") # Создаем поле для ввода
    checkbox = QCheckBox("Спрятать") # Создаем чек-бокс
    checkbox.toggled.connect(linedit.setEchoMode) # Привязываем к методу linedit "спрятать"
    combobox = QComboBox() # Создаем комбо-бокс.
    combobox.addItem("A")
    combobox.addItem("B")
    combobox.addItem("C")
    combobox.addItem("D") # Кладем элементы в комбо-бокс.

    layout = QVBoxLayout() # Вертикальная раскладка, в которую мы будем класть наши виджеты.
    layout.addWidget(label1) # Добавляем надпись.
    layout.addWidget(pic) # Добавляем картинку.
    layout.addWidget(linedit) # Добавляем поле для ввода.
    layout.addWidget(checkbox) # Добавляем чек-бокс.
    layout.addWidget(combobox) # Добавляем комбо-бокс.
    layout.addWidget(button) # Добавляем кнопку.
    self.setLayout(layout) # Устанавливаем наш макет в окно.
```

Output:





Полезные методы QComboBox:



combobox.addItem(list)

Добавить элементы (строки) в QComboBox.

combobox.addItem(str)

Добавить один элемент в QComboBox.

combobox.clear()

Удалить все элементы из QComboBox.

combobox.current_text()

Получить выбранный элемент из QComboBox.

combobox.current_index()

Получить индекс выбранного элемента.

combobox.count()

Получить число всех элементов в QComboBox.



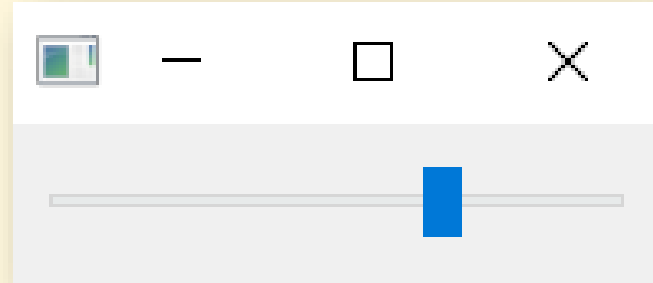
QSlider

```
from PyQt5.QtWidgets import QSlider
```

```
slider = QSlider(Qt.Horizontal)
```



OUTPUT:





QSlider – кастомизация

```
slider = QSlider(Qt.Horizontal)  
slider.setRange(0, 10)  
slider.setTickPosition(QSlider.TicksBothSides)  
slider.setTickInterval(1)
```



OUTPUT:



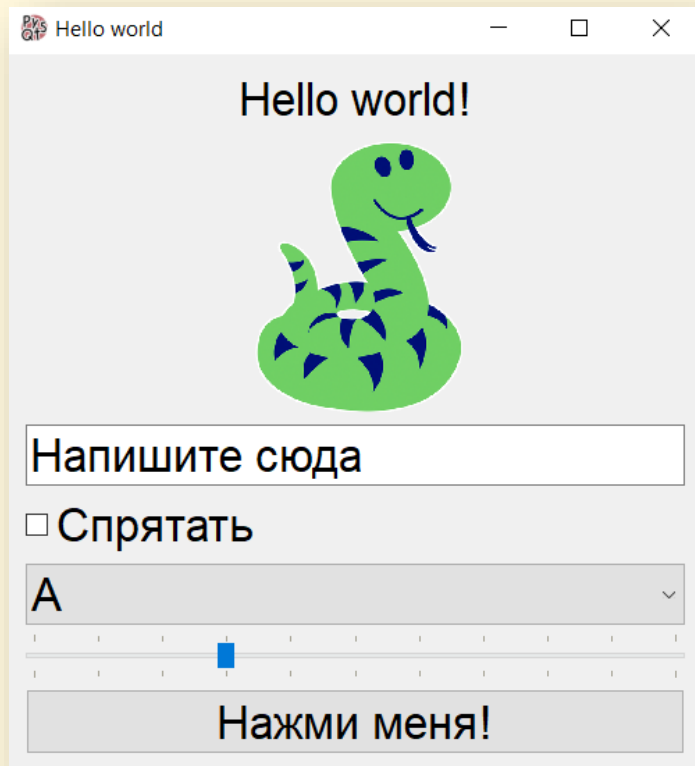
Версия для копирования:

```
def initUI(self): # Основная функция для построения нашего окна.
```

```
    label1 = QLabel("Hello world!") # Создаем нашу надпись.
    label1.setAlignment(Qt.AlignCenter) # Выравниваем по центру.
    pic = QLabel() # Создаем надпись, которую потом заполним изображением.
    pic.setPixmap(QPixmap("image.png").scaled(200, 201)) # Устанавливаем изображение и подгоняем его размер.
    pic.setAlignment(Qt.AlignCenter) # Выравниваем по центру.
    button = QPushButton("Нажми меня!") # Создаем кнопку.
    button.clicked.connect(self.close) # Привязываем кнопку к методу "закрыть".
    linedit = QLineEdit("Напишите сюда") # Создаем поле для ввода
    checkbox = QCheckBox("Спрятать") # Создаем чек-бокс
    checkbox.toggled.connect(linedit.setEchoMode) # Привязываем к методу linedit "спрятать"
    combobox = QComboBox() # Создаем комбо-бокс.
    combobox.addItem("A")
    combobox.addItem("B")
    combobox.addItem("C")
    combobox.addItem("D") # Кладем элементы в комбо-бокс.
    slider = QSlider(Qt.Horizontal) # Создаем слайдер
    slider.setRange(0, 10) # Назначаем шкалу
    slider.setTickPosition(QSlider.TicksBothSides) # Добавляем отметки с двух сторон
    slider.setTickInterval(1) # Выбираем интервал этих отметок
```

```
    layout = QVBoxLayout() # Вертикальная раскладка, в которую мы будем класть наши виджеты.
    layout.addWidget(label1) # Добавляем надпись.
    layout.addWidget(pic) # Добавляем картинку.
    layout.addWidget(linedit) # Добавляем поле для ввода.
    layout.addWidget(checkbox) # Добавляем чек-бокс.
    layout.addWidget(combobox) # Добавляем комбо-бокс.
    layout.addWidget(slider) # Добавляем слайдер.
    layout.addWidget(button) # Добавляем кнопку.
    self.setLayout(layout) # Устанавливаем наш макет в окно.
```

Output:





Полезные методы QSlider:

slider.setRange(int, int)

Назначить область значений на шкале QSlider.

slider.value()

Получить выбранное значение на QSlider.

slider.setValue(int)

Выбрать изначальное положение ручки.

slider.setMaximum(int)

Назначить максимум (бессмысленно, если есть range).

slider.setMinimum(int)

Назначить минимум.

Домашнее задание

Сделайте свою программу любого вида и дизайна!

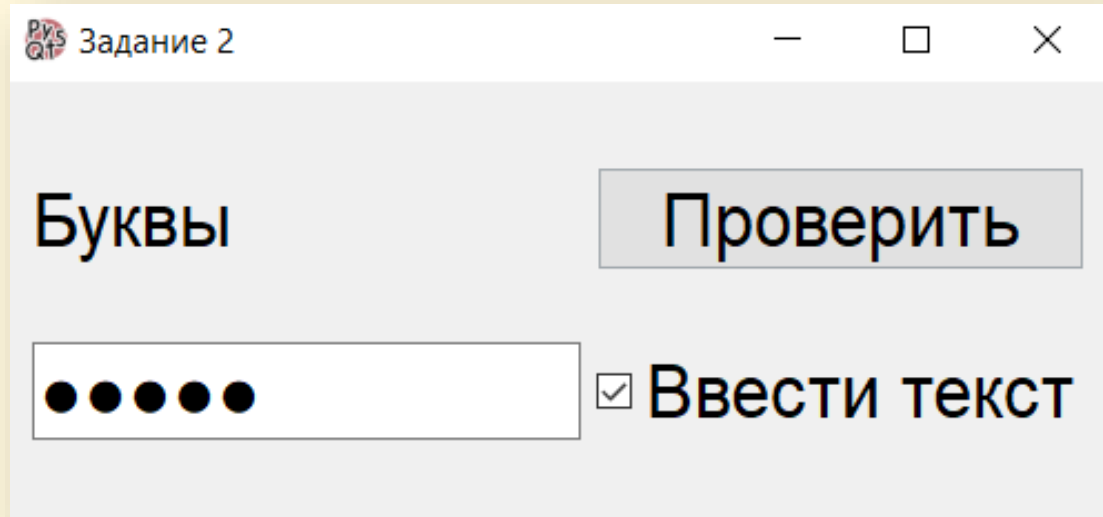
Принимаются любые Ваши задумки – простые или сложные. Напишите мне и пришлите Ваши работы в Telegram или на почту!

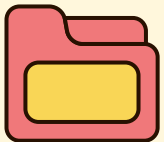
Но если у Вас нет идей, то можете исполнить следующее:

1. **Создайте окно по шаблону, главной разверткой которого будет сетка (QGridLayout).**
2. **Поместите в сетку QLabel, QLineEdit, QCheckBox и QPushButton.**
3. **Запретите доступ (disable) к кнопке QLineEdit, если QCheckBox не подчеркнута.**
4. **Закройте текст при вводе в QLineEdit звездочками.**
5. **Привяжите функцию к QPushButton: функция должна проверять введенный в QLineEdit текст. Если это цифры, то пусть QLabel покажет «Цифры». Если это буквы, то QLabel покажет «Буквы». Иначе в QLabel должно появиться «Другое».**



Домашнее задание

Примерно так будет выглядеть описанная в задании программа:





Спасибо за внимание!



По любым вопросам писать на почту или в Telegram: @lorelei_ether

