

# Разработка GUI с помощью PyQt5

PyQt позволяет разрабатывать десктопный GUI – оконное приложение – посредством размещения и настройки виджетов (элементарных элементов интерфейса).

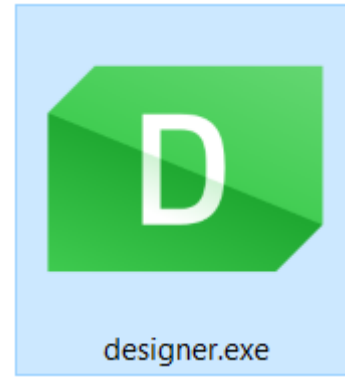
Для разработки графического интерфейса с использованием PyQt5 необходимо установить следующие библиотеки:

```
pip install pyqt5==5.11.3
```

```
pip install pyqt5-tools==5.11.3.1.4
```



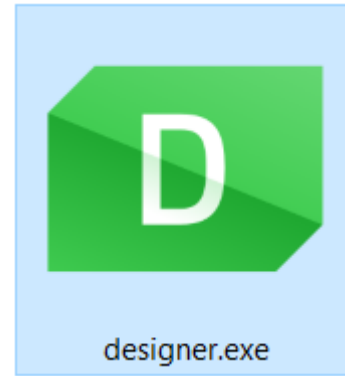
# Qt Designer (Вариант 1)



- Скачиваем и устанавливаем как самостоятельный софт с сайта

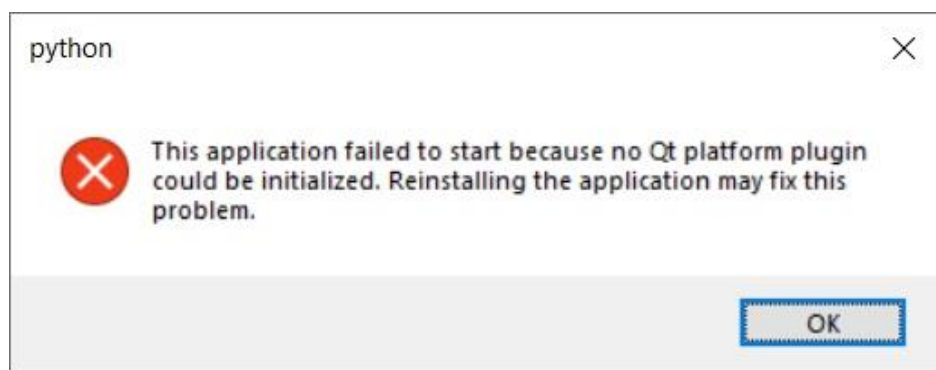
<https://build-system.fman.io/qt-designer-download>

# Qt Designer (Вариант 2)



- После установки необходимых пакетов переходим в следующий каталог:  
C:\Users\<имя пользователя>\PycharmProjects\<название проекта>\venv\Lib\site-packages\qt5\_applications\Qt\bin\
- Запускаем исполняемый файл designer.exe
- Для разработки стартового окна приложения выбираем «Main Window»

# При возникновении проблемы



Запуск из PyCharm починился после того, как прошел следующую инструкцию (взято отсюда: <https://stackoverflow.com/questions/41994485/error-co..>):

1. Set the working directory. File -> Settings -> Build, Execution, Deployment -> Console -> Python Console -> Working directory. Set it to parent directory where your all codes are present.
2. Open Control Panel -> System Settings -> Advanced System Settings -> Environment Variables -> New. Set Variable Name: QT\_PLUGIN\_PATH , Variable Directory: Users\AppData\Local\Continuum\Anaconda2\Library\plugins
3. Restart Pycharm.

папка проекта\venv\Lib\site-packages\PyQt5\Qt\plugins



## Widget Box

Filter

## Layouts

- Vertical Layout
- Horizontal Layout
- Grid Layout
- Form Layout

## Spacers

- Horizontal Spacer
- Vertical Spacer

## Buttons

- Push Button
- Tool Button
- Radio Button
- Check Box
- Command Link Button
- Dialog Button Box

## Item Views (Model-Based)

- List View
- Tree View
- Table View
- Column View
- Undo View

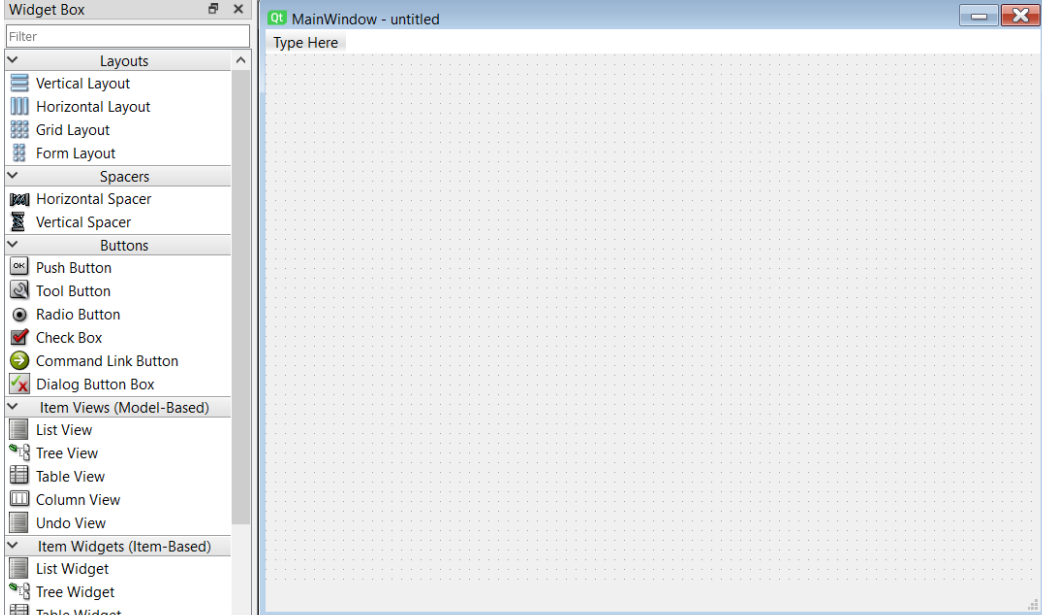
## Item Widgets (Item-Based)

- List Widget
- Tree Widget
- Table Widget

## Containers

- Group Box
- Scroll Area
- Tool Box
- Tab Widget
- Stacked Widget
- Frame
- Widget
- MDI Area
- Dock Widget

## Input Widgets



## Object Inspector

Filter

Object	Class
MainWindow	QMainWindow
centralwidget	QWidget
menubar	QMenuBar
statusbar	QStatusBar

## Property Editor

Filter

MainWindow : QMainWindow

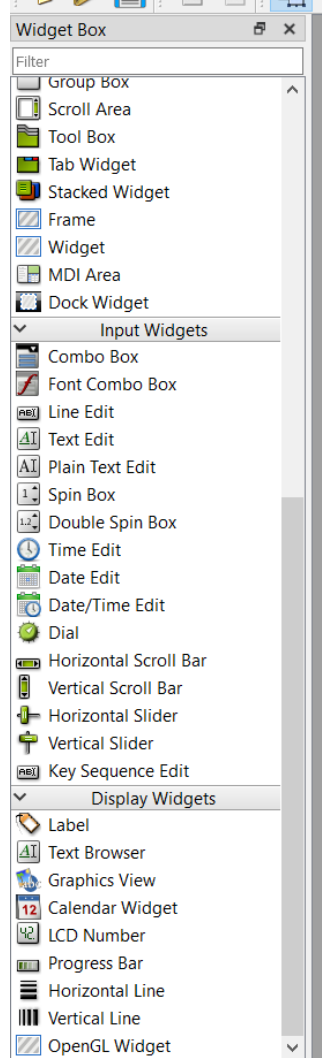
Property	Value
QObject	
objectName	MainWindow
QWidget	
QMainWindow	
iconSize	30 x 30
toolButtonStyle	ToolButtonIconOnly
animated	<input checked="" type="checkbox"/>
documentMode	<input type="checkbox"/>
tabShape	Rounded
dockNestingEnabled	<input type="checkbox"/>
dockOptions	AnimatedDocks/AllowTabbe...

## Resource Browser

Filter

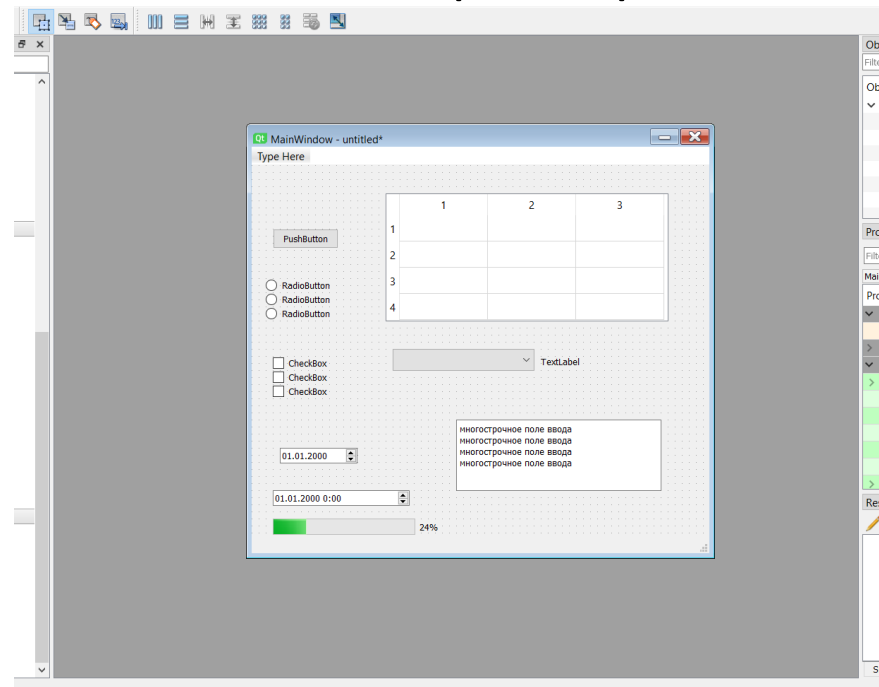


Signal/Slot Editor Action Editor Resource Browser

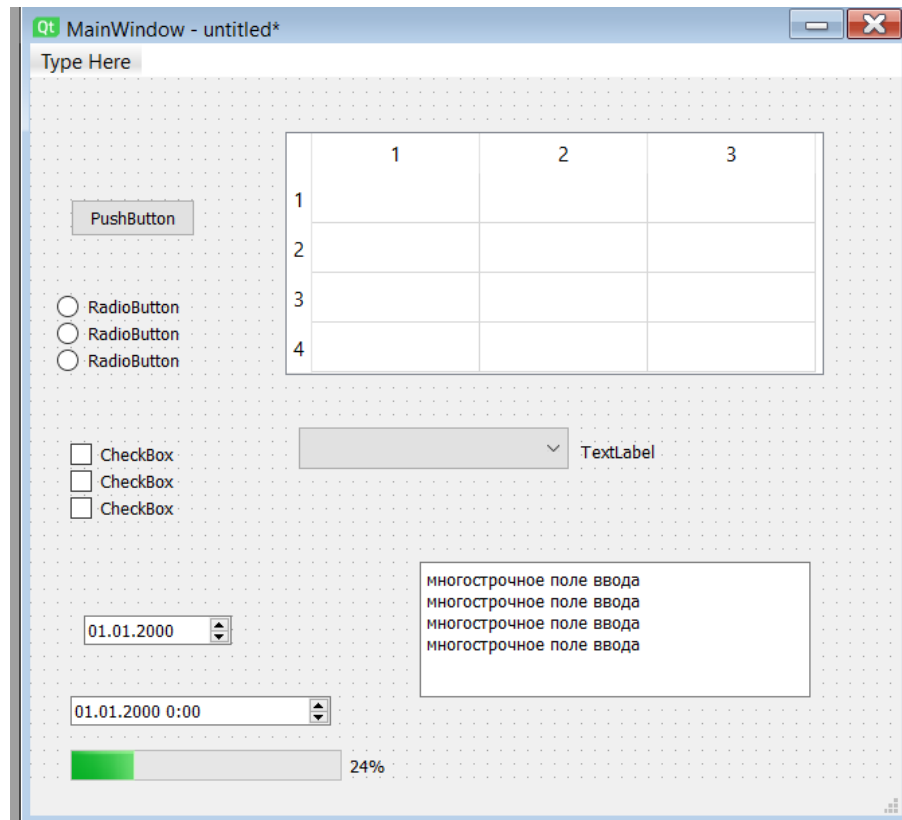


# Панель виджетов

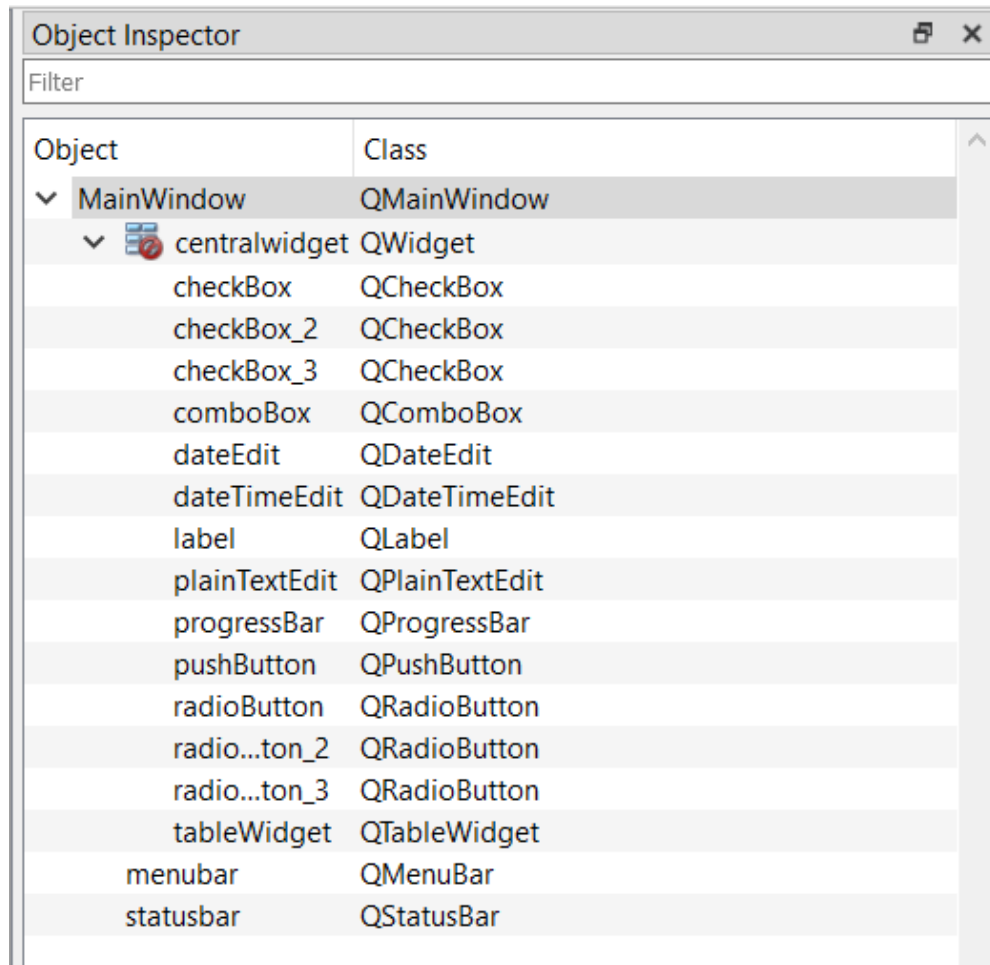
# Рабочее пространство



# Получаемое отображение

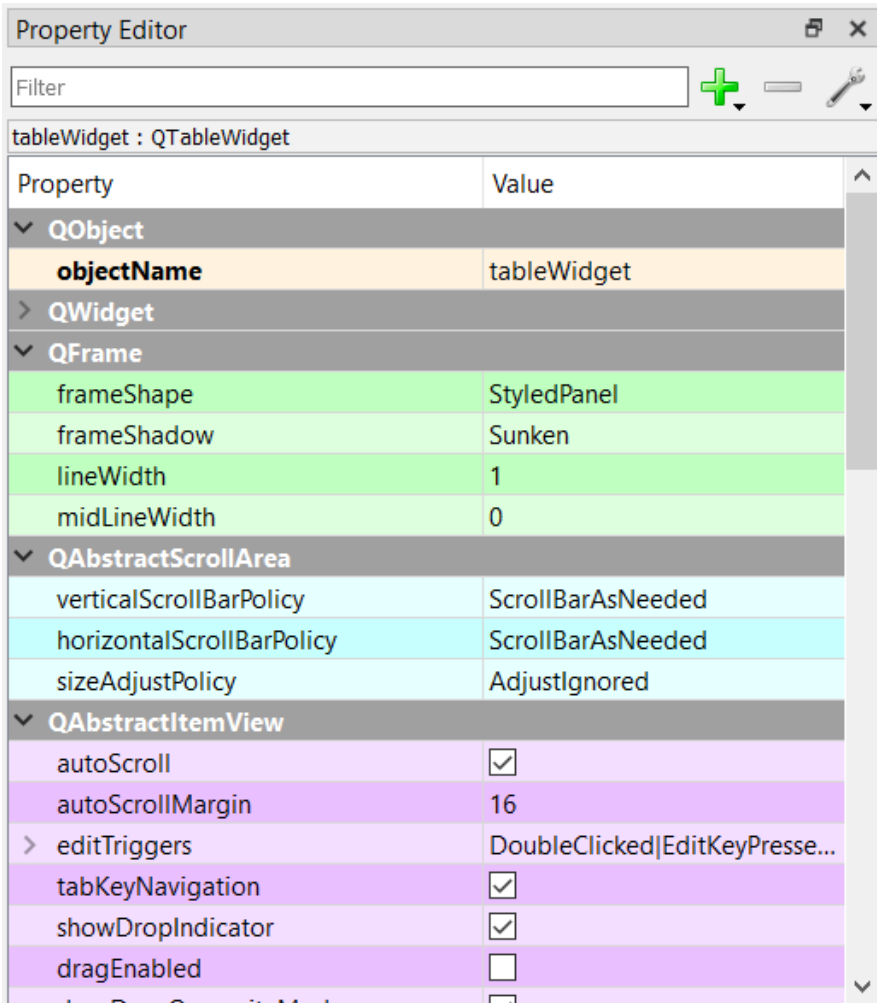


# Панель объектов





# Панель свойств виджета



## Вариант 1

Разработанный макет сохраняется в формате .ui

После чего его необходимо преобразовать в объект класса Qt с помощью команды

```
python -m PyQt5.uic.pyuic -x [FILENAME].ui -o [FILENAME].py
```

Команду лучше записать в **.bat** файл для многократного использования.

В новом файле *filename.py* получаем описание разработанного интерфейса.

Данный файл удобно менять путем повторного преобразования .ui файла в designer.exe, но не в редакторе кода.

## Вариант 2

Копируем из либ файл **pyuic5.exe** и кладем в папку с интерфейсами

После чего его необходимо преобразовать в объект класса Qt с помощью команды

```
pyuic5 filename.ui -o filename.py
```

Для корректной работы команды **pyuic5** необходимо наличие в папке

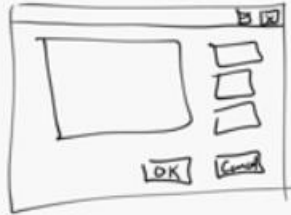
**C:\Program Files\Python3x\Scripts**

файла **pyuic5.exe**

Подробно и с примерами:

<http://projects.skylogic.ca/blog/how-to-install-pyqt5-and-build-your-first-gui-in-python-3-4/>

DESIGN via Qt Designer



Save As

mygui.ui

CONVERT via PyQt

```
> pyuic5 -x mygui.ui -o mygui.py
```

CODE in Python

```
[ ...  
  from mygui import MyGui  
  class MyGuiHolder(MyGui):  
      Business logic here...  
  ... ]
```

Save As

prog.py

EXECUTE

```
> python prog.py
```

# Объявление класса интерфейса

```
from interfase import Ui_Account  
from PyQt5 import QtCore, QtGui, QtWidgets  
import sys
```

```
class Account(QtWidgets.QMainWindow):
```

```
def __init__(self, parent=None):  
    QtWidgets.QWidget.__init__(self)  
    self.ui = Ui_Account()  
    self.ui.setupUi(self)
```

```
def get_difficulty(self):  
    pass
```

```
if __name__ == "__main__":  
    app = QtWidgets.QApplication(sys.argv)  
    myapp = Account()  
    myapp.show()  
    sys.exit(app.exec())
```

Сгенерированный с помощью ruic5 файл  
и класс в нем

Класс интерфейса

Инициализация интерфейса

Метод интерфейса

Команды выполняются если запускается  
именно этот файл

Создается экземпляр класса  
интерфейса, интерфейс запускается

# Объявление с параметром

```
from interfase import Ui_Account
from PyQt5 import QtCore, QtGui, QtWidgets
import sys

class Account(QtWidgets.QMainWindow):

    def __init__(self, user):
        self.user = user
        QtWidgets.QWidget.__init__(self)
        self.ui = Ui_Account()
        self.ui.setupUi(self)

    def get_difficulty(self):
        pass

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    myapp = Account(user)
    myapp.show()
    sys.exit(app.exec())
```

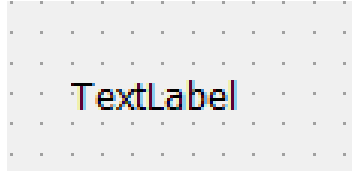
Если класс должен быть объявлен с параметром, то передаем параметр в `init`, дальше используем как переменную класса

# **O self. переменных**

<https://coderoad.ru/8641200/Python-в-чем-разница-между-глобальной-переменной-и-переменной-с-префиксом-self>

# Работа с виджетами

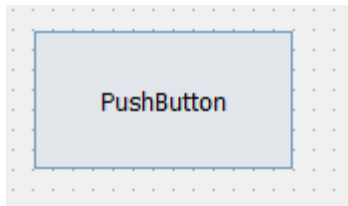
## QLabel – Текстовая метка



<code>self.ui.label.setText("text")</code>	Установить текст в метку <i>В метку можно записывать только текстовую переменную, используйте str() при необходимости</i>
<code>text = self.ui.label.text()</code>	Получить текст из метки
<code>self.ui.label.adjustSize()</code>	Автоматически увеличивать размер метки для передаваемого текста

# Работа с виджетами

## QPushButton – Кнопка



<pre>self.ui.button.setText("text")</pre>	<p>Установить текст на кнопку</p> <p><i>Можно записывать только текстовую переменную, используйте <code>str()</code> при необходимости</i></p>
<pre>self.ui.button.clicked.connect(self.function)</pre>	<p>Выполнение метода класса при нажатии на кнопку</p> <p>Команда пишется в функции <b>init</b></p> <p>Скобки для метода класса не указываются</p>



# Работа с виджетами

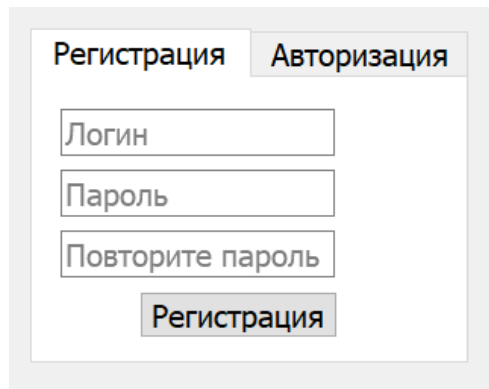
## QLineEdit – Однострочное поле для ввода



<code>text = self.ui.lineEdit.text()</code>	Получить введенный в поле текст, переменная будет строкового типа str
<code>self.ui.lineEdit.clear()</code>	Очистить поле для ввода
<div><div>&gt; <b>placeholderText</b> Введите текст</div><div><div>Введите текст</div></div></div>	Задать подсказку для ввода, пропадет при вводе текста
<div><div><div><b>echoMode</b> Password</div><div>cursorPosition Normal</div><div>alignment NoEcho</div><div>dragEnabled Password</div><div>... PasswordEchoOnEdit</div></div><div><div>••••••••••</div></div></div>	Режим скрытия вводимых символов (ввод пароля)

# Работа с виджетами

## QTabWidget – Вкладки



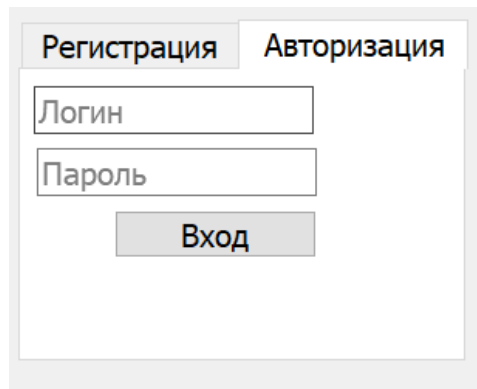
Регистрация   Авторизация

Логин

Пароль

Повторите пароль

Регистрация



Регистрация   Авторизация

Логин

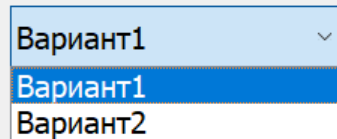
Пароль

Вход

<code>&gt; currentTabText</code> Регистрация	Задать текст на вкладке
<code>self.ui.tab1.deleteLater()</code>	Скрыть вкладку
	<i>*на какой вкладке сохраняешь – та по дефолту и открывается</i>

# Работа с виджетами

## QComboBox – Список



<code>self.ui.comboBox.addItem ("AAA")</code>	Добавить новый вариант в список, принимает строку
<code>self.ui.comboBox.addItems(["AAA", "BBB", "CCC"])</code>	Добавить новые варианты в список, принимает список строк
<code>self.ui.comboBox.xz()</code>	Стартовое значение, подсказка
<code>self.ui.comboBox.activated.connect(self.function)</code>	При выборе одного из элементов в списке вызывается метод класса, указывается в функции <b>init</b> , <b>скобки не указываются</b>
<code>item = self.ui.comboBox.currentText()</code>	Считать выбранный элемент списка

# Работа с виджетами

## QTableWidget – Таблицы

<code>self.ui. table.clear()</code>	Очистить таблицу
<code>self.ui. table.setHorizontalHeaderLabels(["AAA", "BBB", "CCC"])</code>	Называет заголовки колонок, должны быть заранее созданы в Designer
<code>self.ui. table.setVerticalHeaderLabels(["Carry", "on", "my", "wayward", "son"])</code>	Называет заголовки столбцов, должны быть заранее созданы в Designer (редко используется, по умолчанию 0, 1, 2 ...)
<code>self.ui.table.setRowCount(m)</code> <code>self.ui. table.setColumnCount(3)</code>	Установить количество строк и столбцов в таблице, кол-во колонок должно совпадать с предыдущим шагом
<code>self.ui. table.setItem(i, j,</code> <code>QtWidgets.QTableWidgetItem("text")</code>	Установить в ячейку (строку i колонку j) значение строковой переменной

# Работа с виджетами

## QTableWidget – Таблицы

	AAA	BBB	CCC
Carry			
on			
my			
wayward			
son			

horizontalHeaderVisible	<input checked="" type="checkbox"/>	←
horizontalHeaderCascadingSectionResiz...	<input type="checkbox"/>	
horizontalHeaderDefaultSectionSize	125	
horizontalHeaderHighlightSections	<input checked="" type="checkbox"/>	
horizontalHeaderMinimumSectionSize	49	
horizontalHeaderShowSortIndicator	<input type="checkbox"/>	
horizontalHeaderStretchLastSection	<input type="checkbox"/>	
verticalHeaderVisible	<input checked="" type="checkbox"/>	←
verticalHeaderCascadingSectionResizes	<input type="checkbox"/>	
verticalHeaderDefaultSectionSize	37	
verticalHeaderHighlightSections	<input checked="" type="checkbox"/>	

Скрыть названия строк

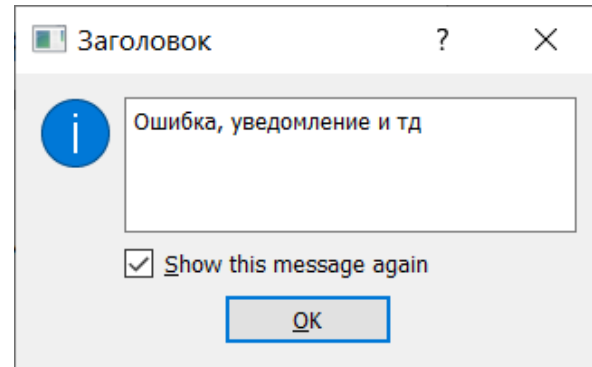
```
for i in range(m):  
    for j in range(3):  
        self.ui.T_my_projects.setItem(i, j,  
QtWidgets.QTableWidgetItem(str(i*10+j)))
```

Пример заполнения таблицы

# Работа с виджетами

## QErrorMessage – Окно ошибки

*Может использоваться для вывода сообщений об ошибке и других уведомлений*



```
self.eBar = QtWidgets.QErrorMessage()
```

Объявляется объект класса  
Окна ошибок, в функции **init**

```
self.eBar.setWindowTitle('Заголовок')  
self.eBar.showMessage('Ошибка, уведомление и тд')
```

Окну присваивается заголовок  
В окне выводится текст

# Работа с виджетами

## QTimer – Таймер интерфейса

*Может использоваться для обновления данных в интерфейсе*

```
self.timer = QtCore.QTimer()
```

```
self.timer.timeout.connect(self.function)
```

```
self.timer.start(5000)
```

Объявляется в функции **init**

Привязывается метод который будет вызываться с периодичностью таймера, *указывается без скобок*

Значение таймера задается в миллисекундах (1сек = 1000 мсек)

# Работа с виджетами

## Работа с окнами

<code>import LK</code>	Импортируется файл с классом нового окна
<code>self.open = LK.ClassLK() self.open.show()</code>	В нужном месте создается объект класса нового окна, отображается на экране
<code>self.close()</code>	Текущее окно при необходимости может быть закрыто



# Работа с виджетами

## QRadioButton – переключатель

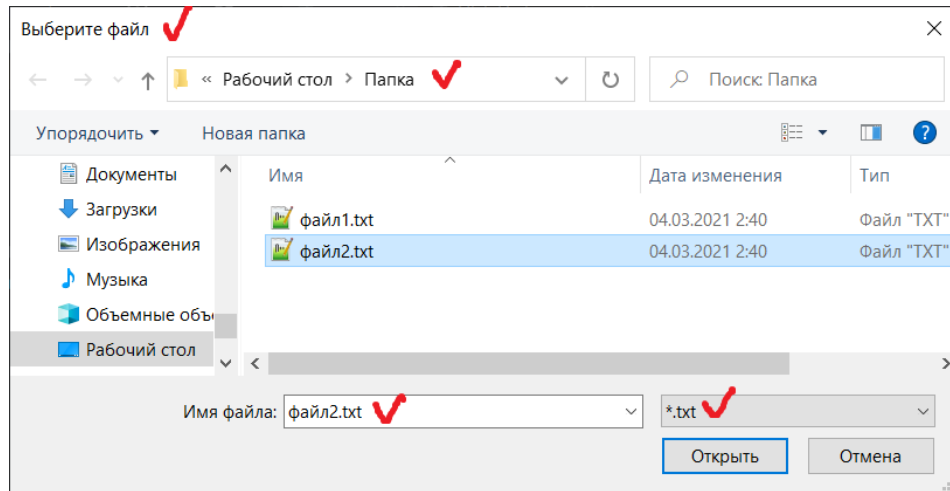
☐ Вариант 1

☒ Вариант 2

☐ Вариант 3


# Работа с виджетами

## Работа с проводником



```
file_path =  
QtWidgets.QFileDialog.getOpenFileName(self,  
'Выберите файл', 'Старт', '*.txt')
```

Открывается окно Проводника в каталоге **Старт** с заголовком **«Выберите файл»**, отобразит файлы подходящие под маску **\*.txt**, возвращает путь к выбранному файлу