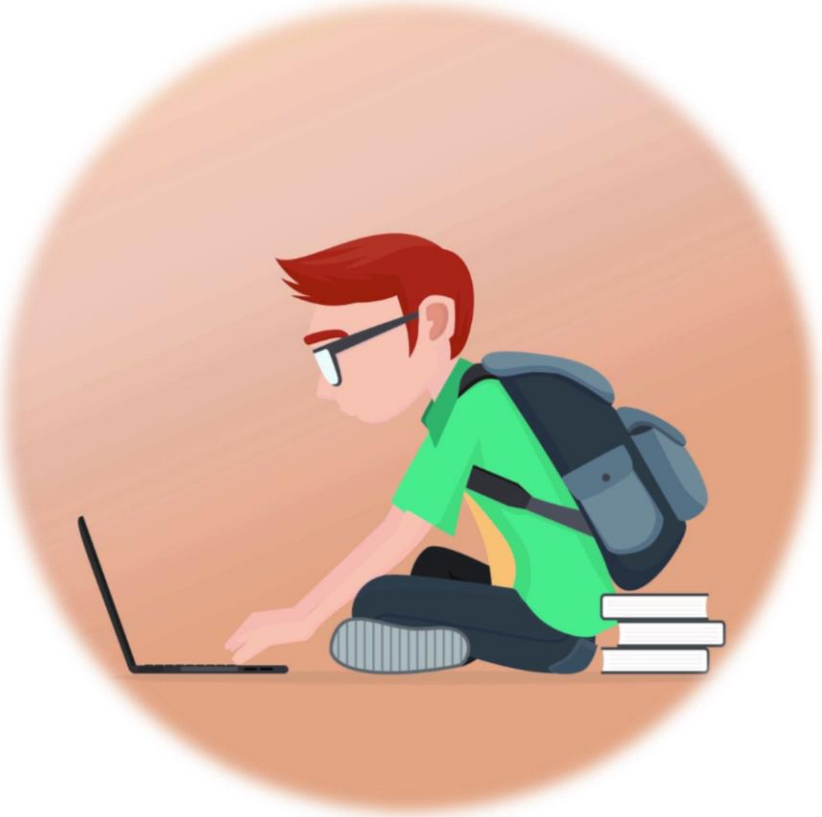
ЯЗЫК ПРОГРАММИРОВАНИЯ PYTHON

Особенности создания оконных приложений в Python (ч.2)

Крашенинников Роман Сергеевич
Главный специалист отдела системного администрирования РХТУ им. Д.И. Менделеева, ведущий программист кафедры информационных компьютерных технологий

ТЕМЫ

- Библиотека **PyQt**
- Создание окна
- Обработка событий
- Основные виджеты
- Диалоговые окна



ПОЛЕЗНЫЕ РЕСУРСЫ

- Краткий курс **PyQt5**
- Полное руководство для новичков
- Простое приложение на **PyQt**

БИБЛИОТЕКА PYQT6

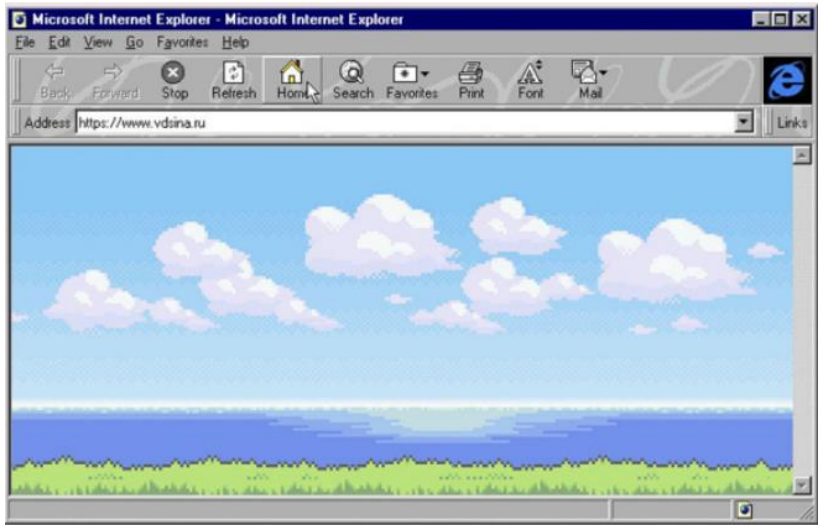
PyQt — набор расширений графического фреймворка Qt для языка программирования Python, выполненный в виде расширения Python.

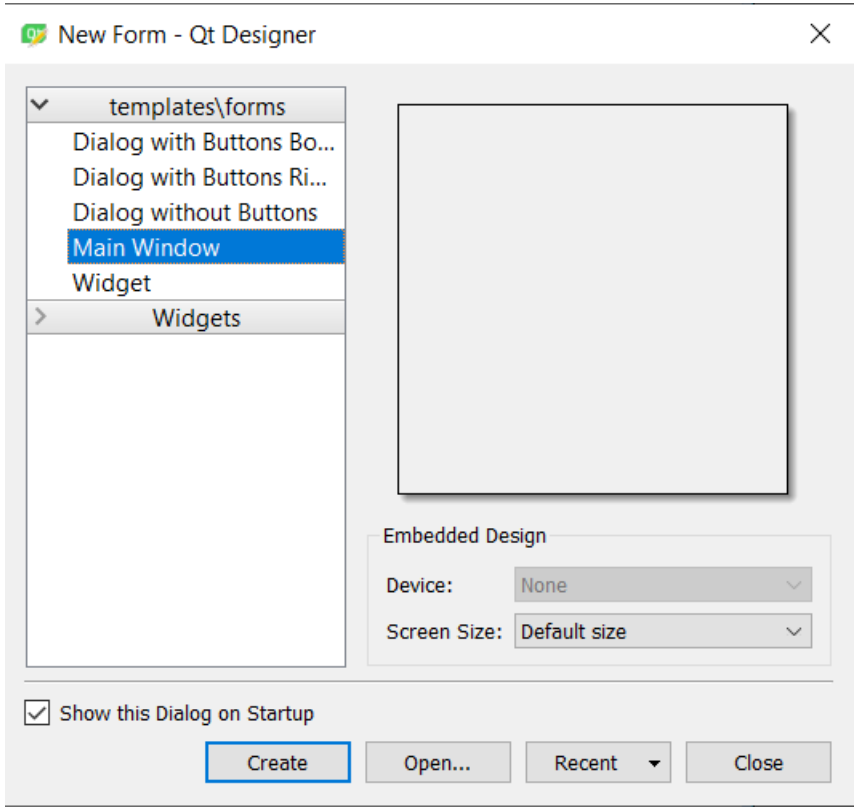
Qt полностью объектно-ориентированная, кросс-платформенная. Дает возможность разрабатывать платформо-независимое ПО, написанный код можно компилировать для Linux, Windows, Mac OS X и других операционных систем. Включает в себя множество классов для работы с сетью, базами данных, классы-контейнеры, а также для создания графического интерфейса и множество других.

С использованием этого фреймворка написано множество популярных программ: 2ГИС для Android, Kaspersky Internet Security, Virtual Box, Skype, VLC Media Player, Opera и другие.

PyQt создание окна

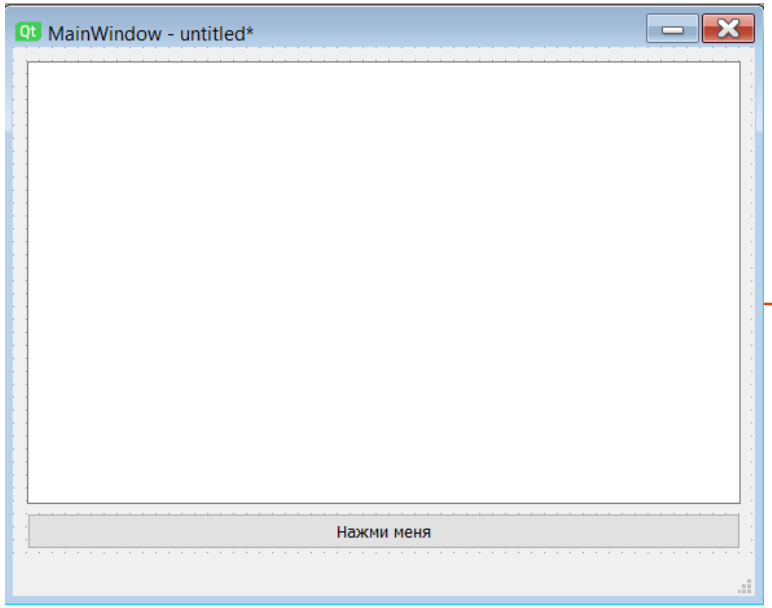
По аналогии с **Tkinter**, главным элементом в **PyQt** является окно. Однако, для создания окна и размещения виджетов на нём, вам не обязательно использовать код, для этого есть специальное приложение **Qt Designer.** В нем, при запуске, для создания оконного приложения, вам необходимо выбрать **Main Window.**





PYQT СОЗДАНИЕ ОКНА

Окно, созданное в **Qt Designer** сохраняется в формате **.ui**, для преобразования его в питоновский код, необходимо воспользоваться утилитой **pyuic6**.




```
In [11]: !pyuic6 my_window.ui -o my_window.py
```

```python
from PyQt6 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(573, 424)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.verticalLayout = QtWidgets.QVBoxLayout(self.centralwidget)
        self.verticalLayout.setObjectName("verticalLayout")
        self.textEdit = QtWidgets.QTextEdit(self.centralwidget)
        self.textEdit.setObjectName("textEdit")
        self.verticalLayout.addWidget(self.textEdit)
        self.pushButton = QtWidgets.QPushButton(self.centralwidget)
        self.pushButton.setObjectName("pushButton")
        self.verticalLayout.addWidget(self.pushButton)
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.pushButton.setText(_translate("MainWindow", "Нажми меня"))
```

PyQt создание окна

Далее, вам нужно просто импортировать созданный класс в вашей программе и создать дочерний класс, отнаследованный от него и класса **QtWidgets.QMainWindow.** Эти действия помогут создать вам скелет вашего окна и вам останется только описать логику его работы.

Шаблон приложения в **PyQt**

```
from PyQt6 import QtWidgets
from my_window import Ui_MainWindow
import sys

class Window(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)

app = QtWidgets.QApplication(sys.argv)
window = Window()
window.show()
app.exec()
```

Инициализация переменных окна и нашего дизайна

Создание объекта приложения

Создания окна

Отображение окна
и запуск приложения

PyQt события

Все приложения с графическим интерфейсом являются событийно-ориентированными. События вызываются главным образом пользователем приложения. Когда мы вызываем метод exec(), приложение входит в главный цикл. Главный цикл получает события и отправляет их объектам. В модели событий имеются три участника:

- Источник события
- Объект события
- Цель события

Источник события – это объект, состояние которого меняется. Он вызывает событие. Событие инкапсулирует изменение состояния в источнике события. Цель события – это объект, которому требуется уведомление. Объект источника события делегирует задачу обработки события цели события.

PyQt события

Для работы с событиями PyQt6 имеет уникальный механизм сигналов и слотов. Сигналы и слоты используются для связи между объектами. Сигнал срабатывает тогда, когда происходит конкретное событие. Слот может быть любой функцией. Слот вызывается, когда срабатывает его сигнал. Отправитель – объект, который посылает сигнал. Получатель – объект, который получает сигнал. Слот – это метод, который реагирует на сигнал. Сигналы можно обработать через метод connect или путём переопределения обработчиков.

Привязка через **connect**

```
class Window(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.pushButton.clicked.connect(self.buttonClicked)
        self.pushButton_2.clicked.connect(self.buttonClicked)

    def buttonClicked(self):
        sender = self.sender()
        text = sender.text()
        sender.setText(self.label.text())
        self.label.setText(text)
```

Привязка через переопределение

```
from PyQt6 import QtWidgets
from PyQt6.QtCore import Qt
from my_window import Ui_MainWindow
import sys

class Window(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)

    def keyPressEvent(self, e):
        if e.key() == Qt.Key.Key_Escape.value:
            self.close()
```

PYQT QCHECKBOX

QCheckBox – чекбокс (виджет, который имеет два состояния: включен и выключен). Как правило, чекбоксы используют для функций приложения, которые могут быть включены или выключены.

Использование чекбокса

```
class Window(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.pushButton.clicked.connect(self.buttonClicked)
        self.pushButton_2.clicked.connect(self.buttonClicked)
        self.checkBox.stateChanged.connect(self.changeTitle)
        self.addition = ''

    def buttonClicked(self):
        sender = self.sender()
        text = sender.text()
        sender.setText(self.label.text() + self.addition)
        self.label.setText(text)

    def changeTitle(self, state):
        if self.sender().isChecked():
            self.addition = self.checkBox.text()
        else:
            self.addition = ''
```

PYQT QSLIDER

Qslider – ползунок (виджет, который имеет простой регулятор). Этот регулятор может быть утянут назад и вперёд. Таким способом, мы выбираем значение для конкретной задачи. Иногда, использование ползунка более естественно, чем ввод числа или использование переключателя-счётчика.

Использование слайдер

```
class Window(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.pushButton.clicked.connect(self.buttonClicked)
        self.pushButton_2.clicked.connect(self.buttonClicked)
        self.horizontalSlider.valueChanged.connect(self.sliderChange)

    def buttonClicked(self):
        sender = self.sender()
        text = sender.text()
        sender.setText(self.label.text())
        self.label.setText(text)

    def sliderChange(self):
        self.label.setText(str(self.sender().value()))
```

PYQT QPIXMAP

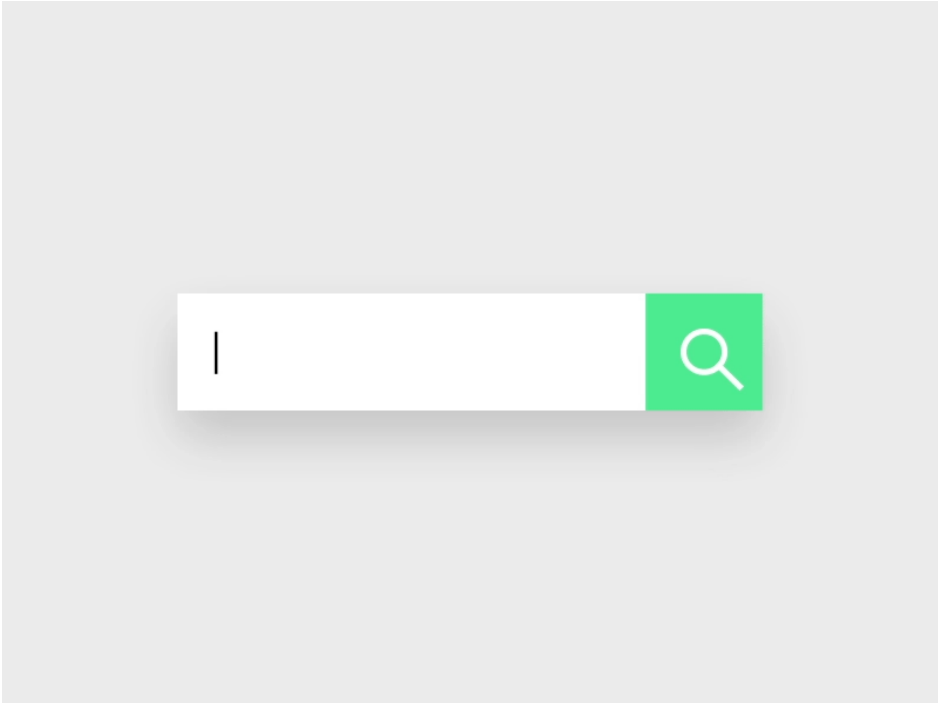
QPixMap – это один из виджетов, использующихся для работы с изображениями. Он оптимизирован для показа изображений на экране.

Использование **QPixMap**

```
from PyQt6 import QtWidgets
from PyQt6.QtCore import Qt
from PyQt6.QtGui import QPixmap
from my_window import Ui_MainWindow
import sys

class Window(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        pixmap = QPixmap("laybByL_mL8.jpg")
        self.label.setPixmap(pixmap)
```

PYQT QLINEEDIT

QLineEdit – это виджет, который разрешает вводить и редактировать одну строку текста.



Использование **QLineEdit**

```
class Window(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.lineEdit.textChanged[str].connect(self.onChanged)

    def onChanged(self, text):
        self.label.setText(text)
```

PYQT QCOMBOBOX

QComboBox – это виджет, который позволяет пользователю выбирать из списка вариантов (выпадающий список).

Использование комбобокса

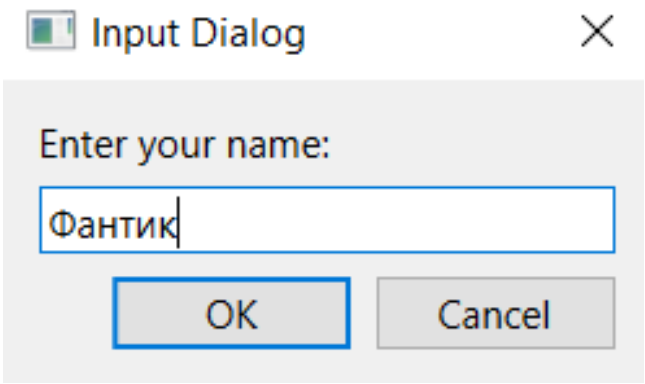
```
class Window(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.comboBox.addItems(["Ubuntu", "Mandriva",
                "Fedora", "Arch", "Gentoo"])
        self.comboBox.activated.connect(self.onActivated)


    def onActivated(self, n):
        self.label.setText(self.sender().itemText(n))
```

PYQT ДИАЛОГОВЫЕ ОКНА

Диалоговые окна (диалоги) являются неотъемлемой частью большинства современных графических приложений. Диалог в обычной жизни - это беседа между двумя и более людьми. В компьютерном приложении, диалог – это окно, которое используется, чтобы «беседовать» с приложением. Диалоги используются для ввода и изменения данных, изменения настроек приложения, и так далее.

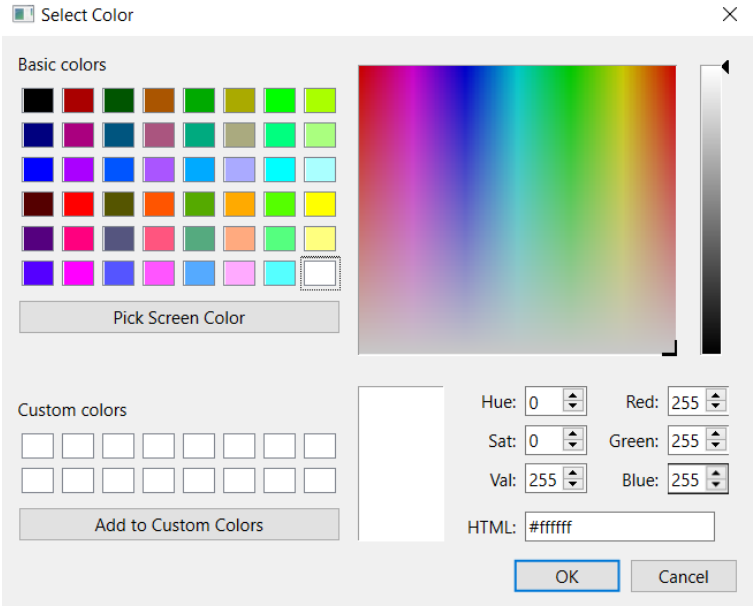
PYQT ДИАЛОГОВЫЕ ОКНА

QInputDialog




```
#text - то, что ввели, ok - нажата ли кнопка ок
text, ok = QInputDialog.getText(self, 'Input Dialog', 'Enter your name:')
```

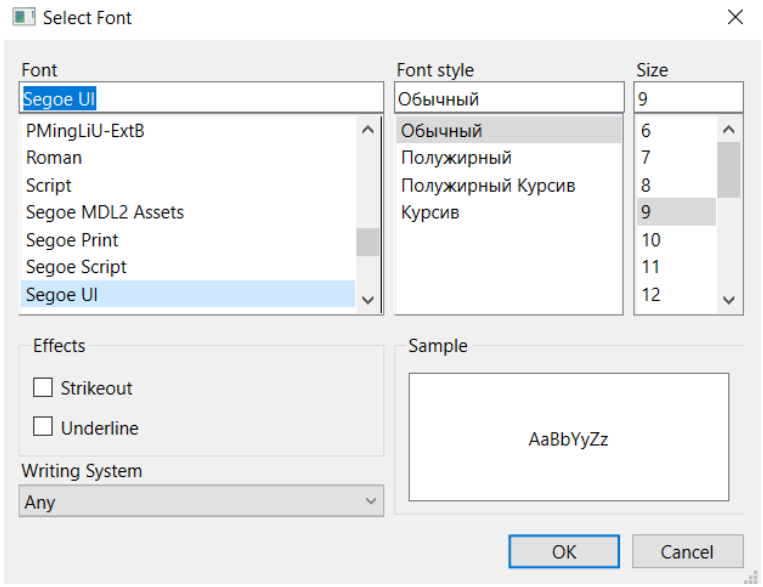
QColorDialog




```
#col - цвет в специализированном Qt формате
col = QColorDialog.getColor()
```

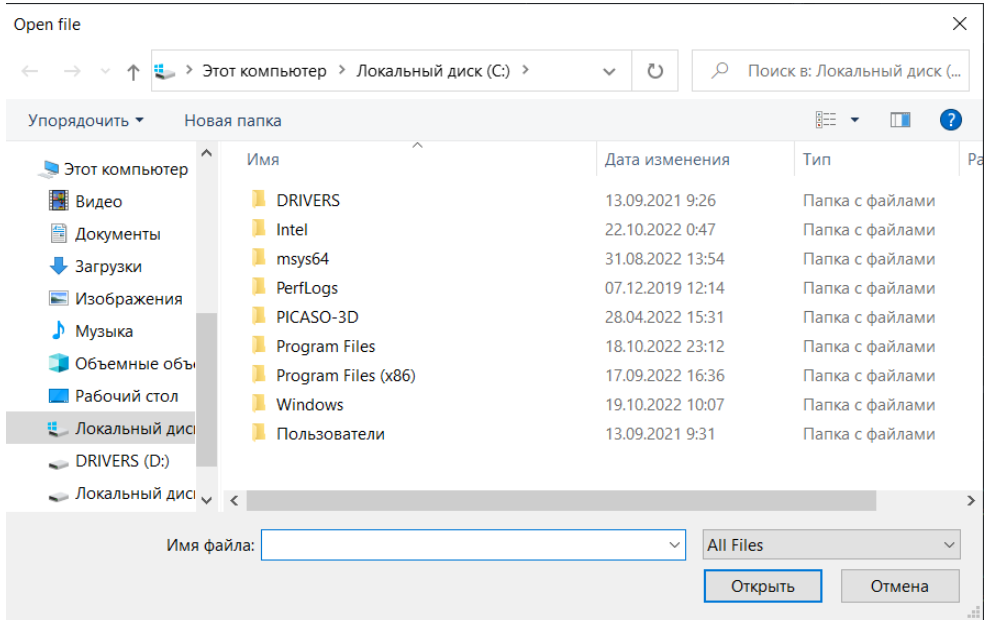
PYQT ДИАЛОГОВЫЕ ОКНА

QFontDialog




```
#font - шрифт, ok - нажата ли кнопка ok
font, ok = QFontDialog.getFont()
```

QFileDialog




```
#fname - путь к выбранному файлу
fname = QFileDialog.getOpenFileName(self, 'Open file')[0]
```

ЗАДАНИЕ ДЛЯ САМОКОНТРОЛЯ

С использованием фреймворка PyQt6 разработать оконное приложение, позволяющее аппроксимировать экспериментальные данные полиномом заданной степени с выводом погрешности аппроксимации. Добавить в программу возможность загрузки данных из файла формата **json,** а так же возможность сохранения графика в выбранный файл, используя **QFileDialog.getSaveFileName()**.

СПАСИБО ЗА ВНИМАНИЕ!