Практическое задание №4.

Расширение функционала редактора с подсветкой кода до возможности запуска Python-скриптов.

Цель работы: Получить практические навыки работы с классом QProcess в среде Qt, освоить механизм запуска внешних скриптов из приложения.

Задание:

- Добавить кнопку «Выполнить» и виджет для вывода результата запуска скриптов в главное окно и скомпоновать их на форме.
- Реализовать слот для кнопки выполнения скрипта, в результате которого код программы на Python будет запущен.

Порядок выполнения:

1) Добавьте новые элементы в интерфейс по заданию. Например, редактор может выглядеть так:

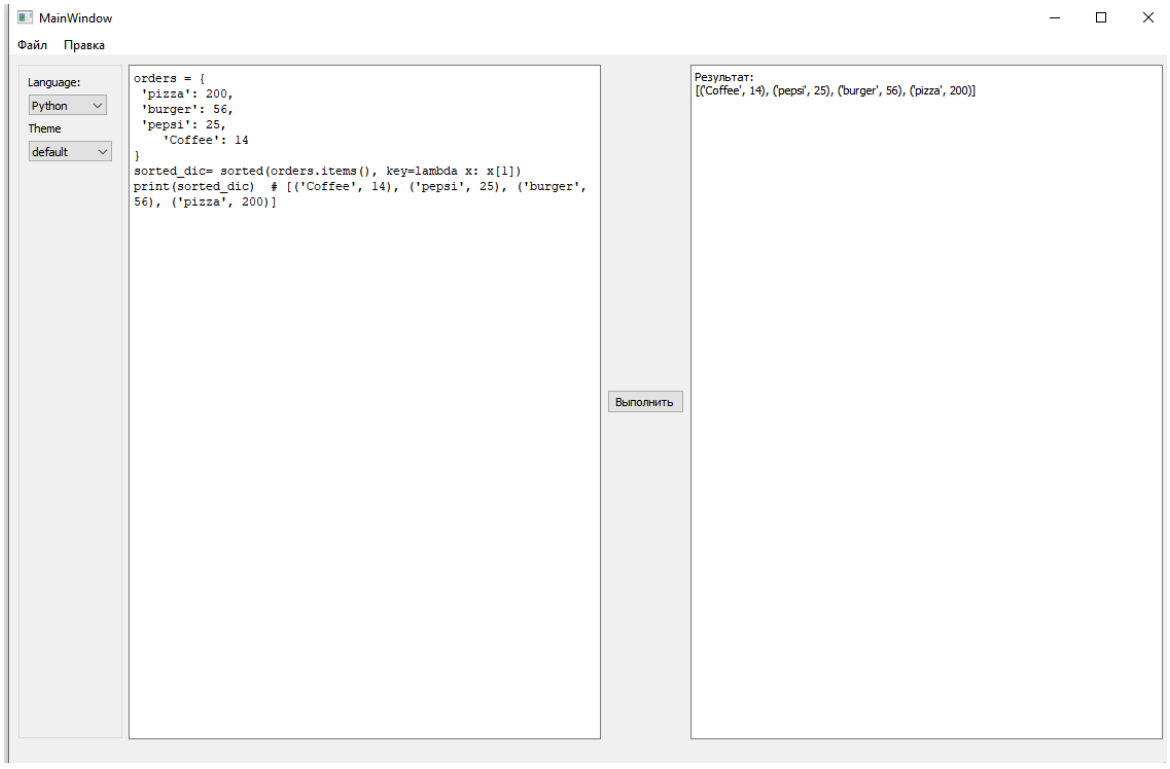

Рисунок 26. Редактор с функцией запуска скриптов

2) Для реализации запуска скриптов используйте QProcess — класс Qt, который позволяет запускать внешние программы и взаимодействовать с ними. Реализуйте слот, в котором текст программы будет сначала

сохранен во временный файл, а затем этот файл будет подан на исполнение следующим образом:

```
QProcess process;
process.start("python", QStringList() << filePath);

if (!process.waitForStarted()) {
    QMessageBox::critical(this, "Ошибка", "Python не установлен или не найден в PATH");
    return;
}

process.waitForFinished(3000); // ждём до 3 секунд

// Считываем вывод и ошибки
QString output = process.readAllStandardOutput();
QString error = process.readAllStandardError();
```

Рисунок 27. Фрагмент кода слота запуска скриптов

Использование в данной реализации метода waitForFinished(int msecs) позволяет ожидать завершения процесса заданное количество миллисекунд. Этот метод возвращает true, если процесс успешно завершился и возвращает false, если истекло время ожидания или произошла ошибка.

3) Выводите результат исполнения кода в интерфейс.
4) Добавьте проверку выбранного языка из выпадающего списка, чтобы реализованный слот выполнялся только при выборе языка Python. При выборе других языков редактор должен информировать об ограниченном функционале.
5) Реализовать обработку ситуации, в которой Python-скрипт завис или выполняется слишком долго. Для этого можно использовать process.kill() с кнопкой «Остановить», а также реализовать проверку времени ожидания.
6) Протестируйте работу полученного приложения, проверив следующие случаи:
    - Пользователь нажимает «Выполнить» с пустым редактором → приложение не должно аварийно завершаться.
    - Проверить, отображается ли правильный результат исполнения простого Python-скрипта в виджете вывода.
    - Если в коде есть ошибка, то она должна корректно отображаться в том же выводе.

Дополнительное задание: реализуйте возможность запуска Python-скриптов в отдельном потоке с помощью QThread или QtConcurrent, создав отдельный класс для работы с процессом.

Контрольные вопросы:

1) Что такое QProcess и зачем он используется в Qt?
2) Что означают параметры при вызове функции process.start()?
3) Как проверить, успешно ли запущен процесс?
4) Как получить вывод программы после её выполнения?
5) Почему важно использовать временные файлы при запуске пользовательского кода?
6) Как работает метод waitForFinished() и зачем он нужен?
7) Как очистить временный файл после выполнения скрипта и как можно использовать при этом QTemporaryFile?
8) Как обрабатывать ситуации, когда Python-скрипт завис или выполняется слишком долго?