**Меню и диалоговые окна**

Для начала создадим в Qt Designer простой интерфейс, состоящий из главного окна размером 320x240. Внутрь поместите многострочное текстовое поле (QPlainTextEdit) и скомпонуйте содержимое главного окна по сетке.

Все управление будет происходить из меню. Дважды нажмите левой кнопкой мыши на надпись «Пишите здесь». Напишите имя группы меню. Пусть она называется File. Нажмите **Enter.** Qt Designer построит группу меню под названием File. Теперь таким же образом, двойным нажатием мыши, создайте пункт меню Save, добавьте разделитель, а под ним - пунк меню Exit (см. рис. 1).

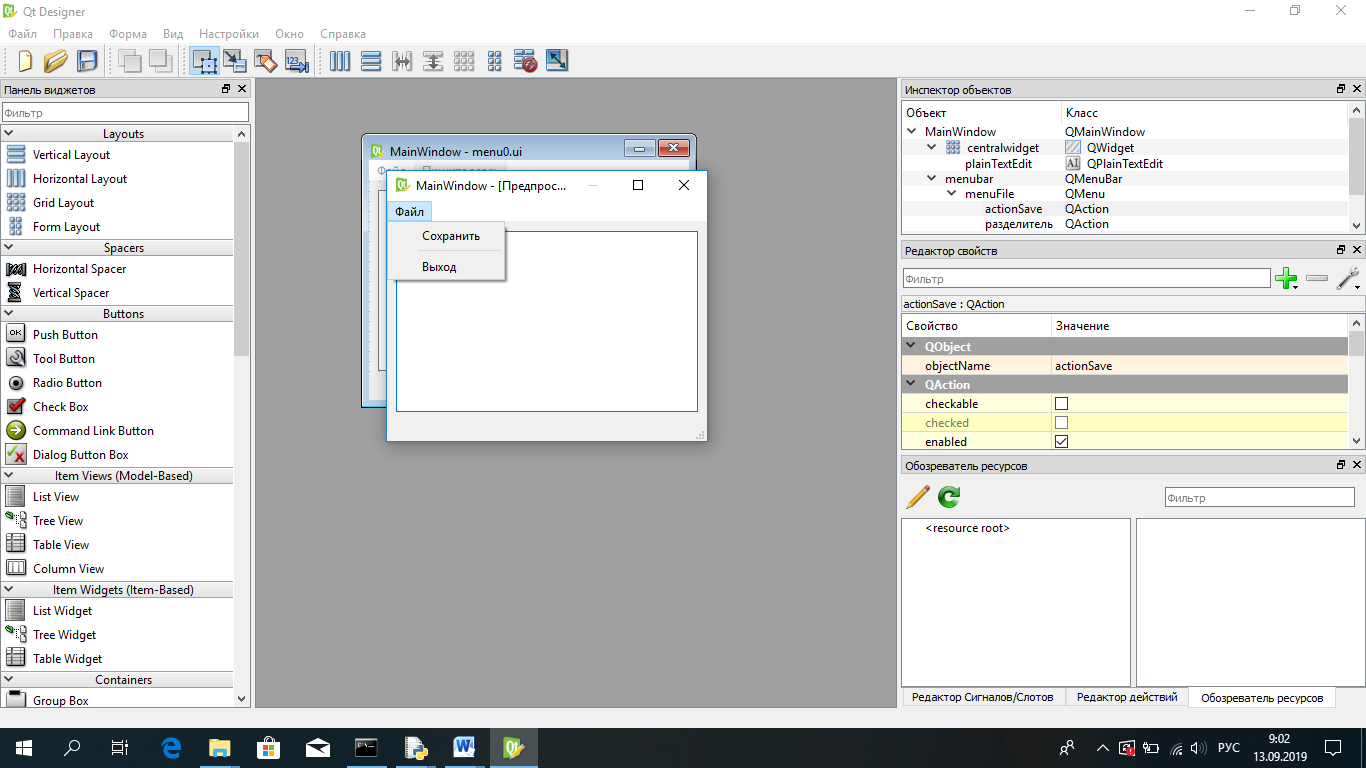


Рисунок 1. Интерфейс главного окна

Теперь можно конвертировать файл manuO.ui в menuO.py. Должен получиться следующий код (см. рис. 2).

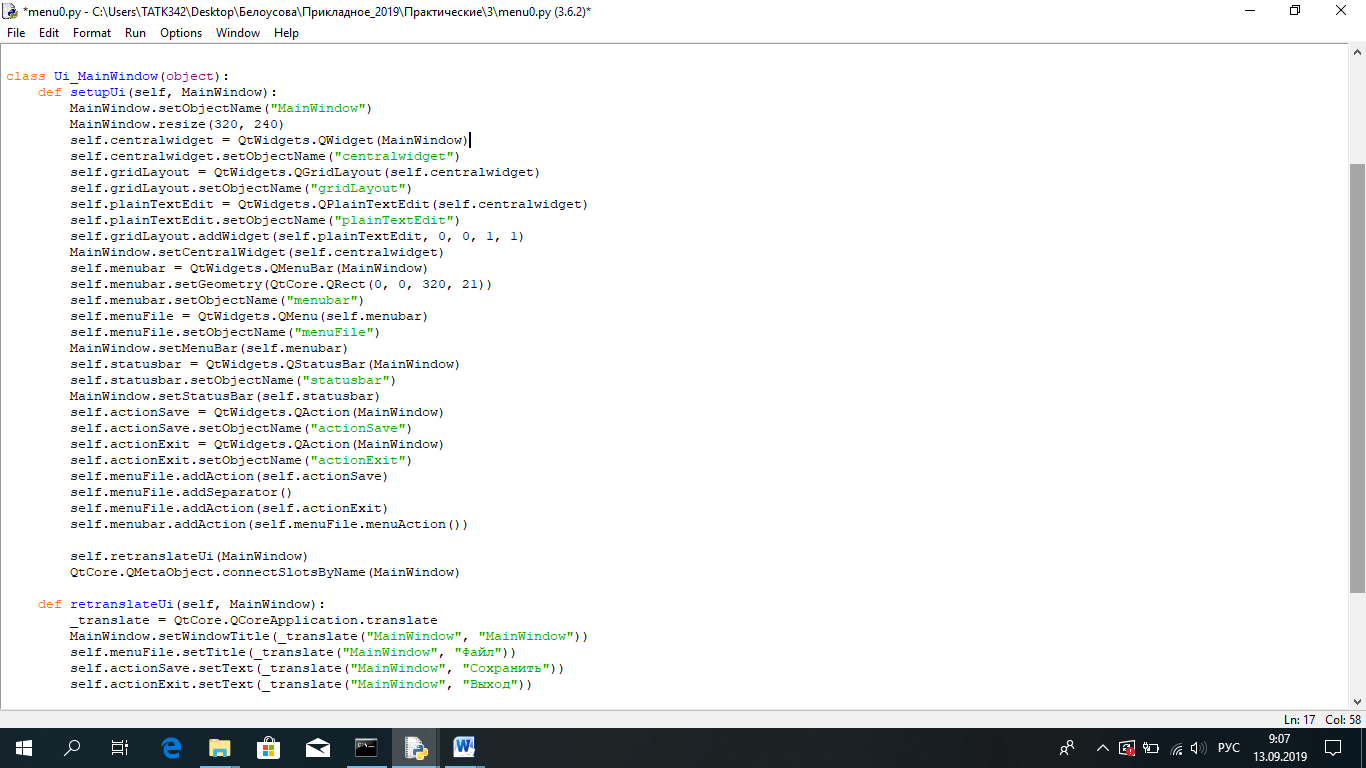


Рисунок 2. Конвертированный файл

Постройте код исполняемой программы. Для этого, как обычно, воспользуйтесь построенным шаблоном.

Запустите программу menumain0.py. Меню нажимается и открывается, в поле можно что-то написать.

Теперь можно переходить к настройкам меню. Найдите в нем переменные пунктов меню - actionSave и actionExit.

Начнем с пункта меню Exit. Чтобы привязать функцию к пункту меню существует сигнал triggered. Чтобы программа просто закрывалась при нажатии пункта меню Exit, то в конец функции init () достаточно добавить строку: **self.ui.actionExit.triggered.connect(self.close) (**метод connect () в качестве аргумента может содержать стандартную команду. Поскольку через self мы вызываем главное окно, self, close означает «Закрой главное окно»).

Пусть при нажатии на крестик и при выборе пункта меню Exit программа выводит диалоговое окно для подтверждения выхода из программы. При нажатии на крестик происходит событие closeEvent. Можно перехватить это событие и при его наступлении проигнорировать. Изучите функцию (см. рис.3):

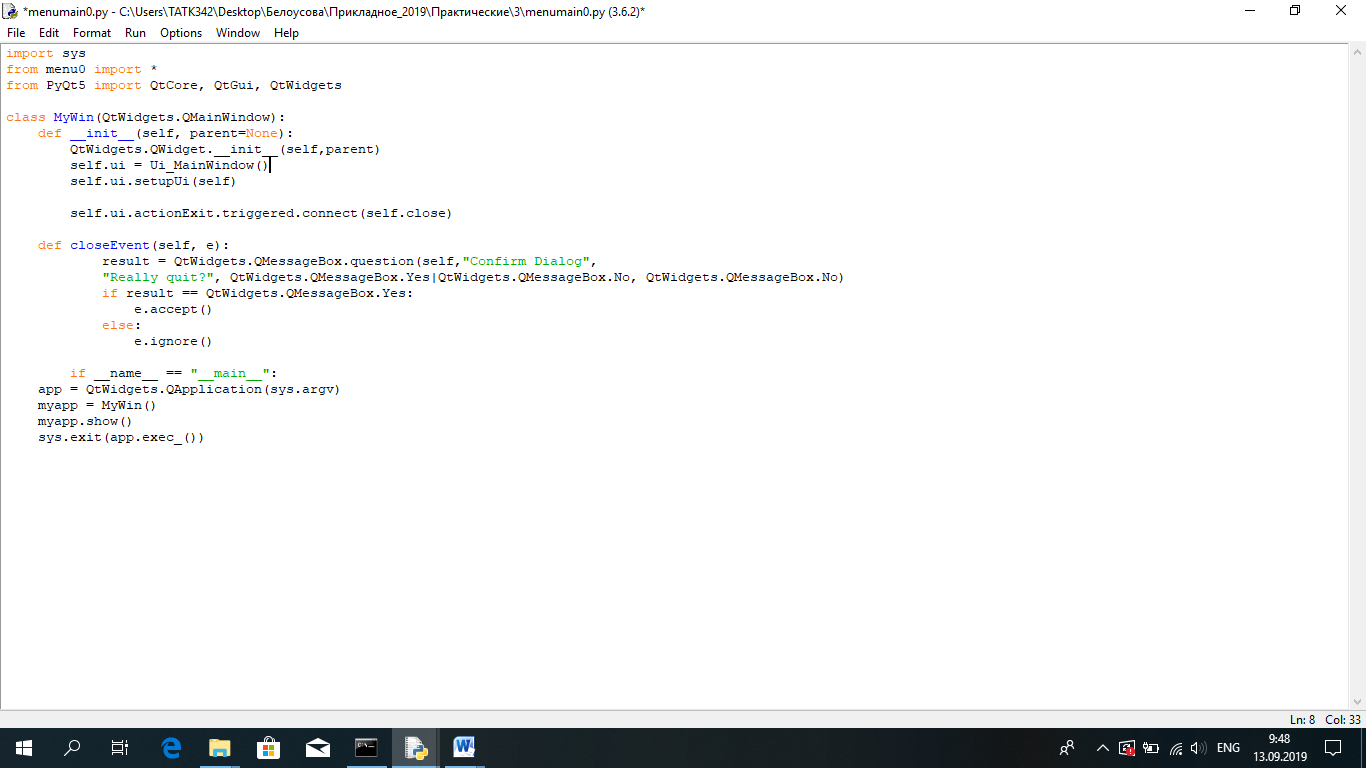


Рисунок 3. Функция closeEvent

Здесь е - переменная события closeEvent. Методы accept () и ignore () соответственно принимают событие или игнорируют его. При наступлении события выводится диалоговое окно класса QMessageBox (его вариант для вывода вопроса). У этого диалогового окна пять аргументов: родительский виджет (self, т.е. главное окно); заголовок окна; текст внутри окна; через знак | кнопки, которые будут у окна (в нашем окне кнопки Yes и No); кнопка по умолчанию (No). Задача диалогового окна вопроса – отговорить пользователя совершать действие, дать ему время одуматься. Это диалоговое окно является модальным по умолчанию.

Поместите функцию в класс MyWin(), ниже функции init (). Запустите программу (см. рис. 4).

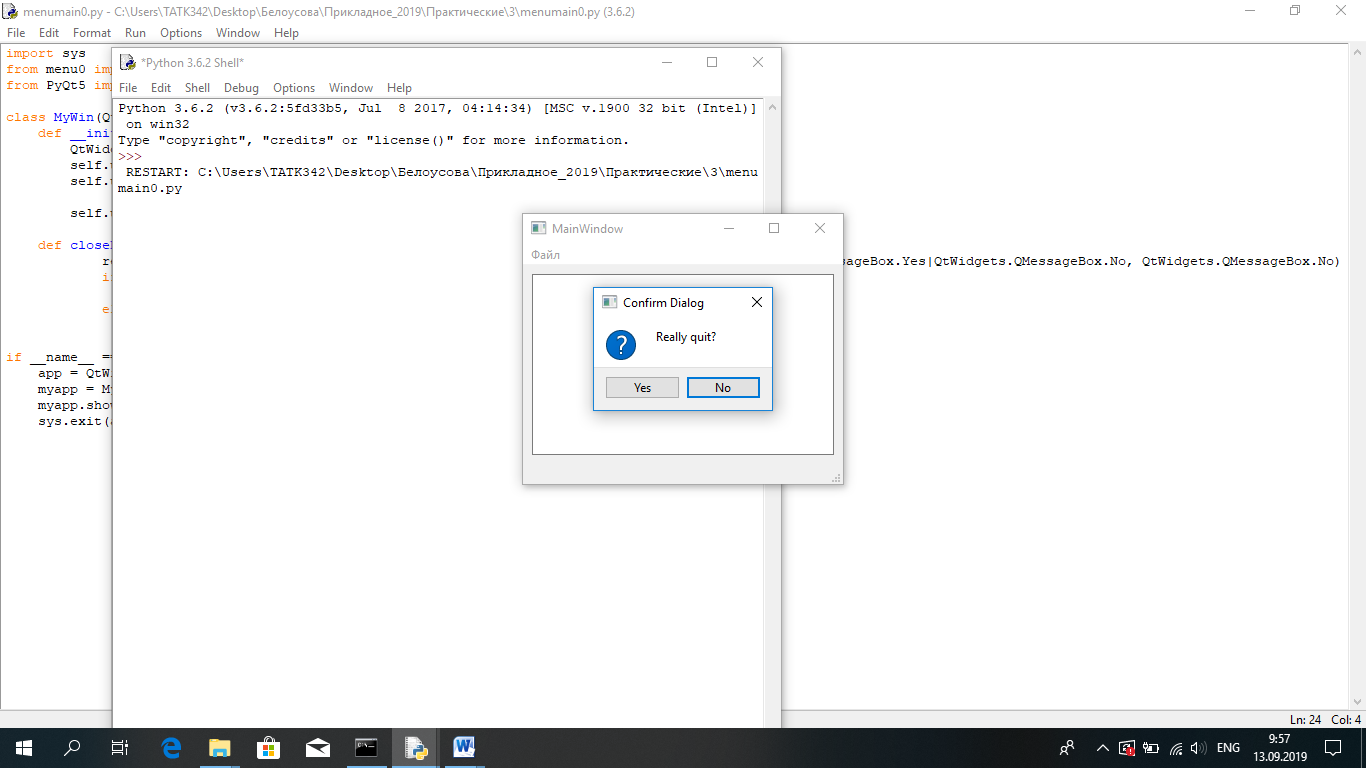


Рисунок 4. Диалоговое окно выхода

Казалось бы, нужно написать отдельную функцию, которая дублировала бы содержание функции closeEvent (). Но секрет в том, что ничего делать не надо. Выбирая пункт меню Exit Вы вызываете то же самое событие close. И следовательно, также вызывается функция closeEvent ().

Самостоятельно:

* Изменить код программы menumain0.py, чтобы при нажатии на крестик не происходило бы ничего, а диалоговое окно вопроса выводилось бы только при выбора пункта меню Exit?
* Добавьте в диалоговое окно третью кнопку со значением Cancel.
* Модифицируйте программу menumain0.py так, чтобы текст внутри диалогового окна был размещен в несколько строк.

Перейдем к пункту меню Save. Для сохранения содержимого многострочного текстового поля в файл нужно написать функцию и привязать ее к пункту меню.

**def saveToFile(self):**

**self.writeFile = open(“text.txt”, 'w', encoding='utf-8') self.writeFile.write(self.ui.plainTextEdit.toPlainText())**

**self.writeFile.close()**

Для того, чтобы привязать функцию к пункту меню, добавьте в конец функции init () строку: **self.ui.actionSave.triggered.connect(self.saveToFile)**

Дополним функцию saveToFile () диалоговым окном сохранения файла. Такое диалоговое окно уже имеется в готовом виде (cм. рис.5)

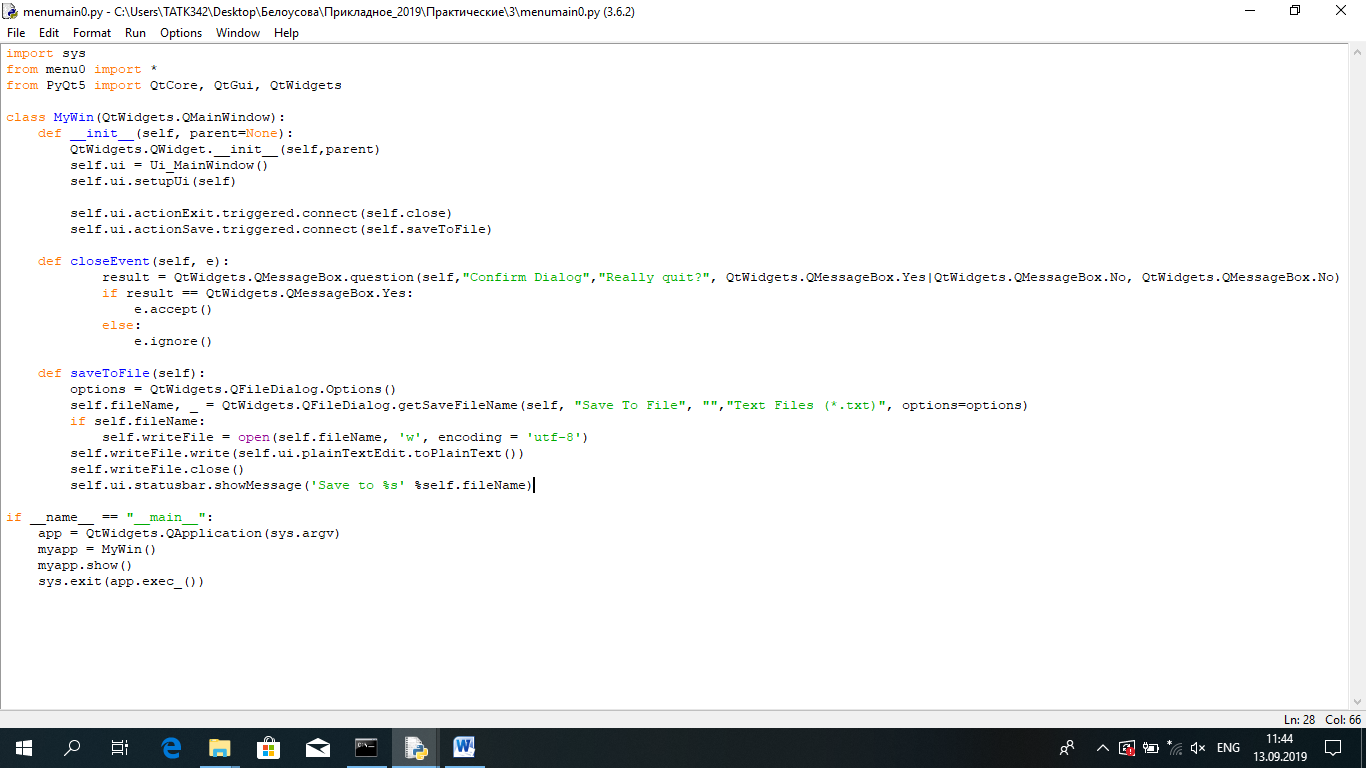


Рисунок 5. Функция сохранения окна

Строка 2 создает переменную, в которой хранятся опции диалогового окна. В таком написании все опции имеют значения по умолчанию. Строка 3 объявляет переменную диалогового окна. В ней хранится название файла, который выберет пользователь. *Пока не обращайте внимания на запятую и подчеркивание*. У диалогового окна пять аргументов: родительский виджет; заголовок (Save То File); имя файла по умолчанию; фильтр (окно будет показывать только файлы с расширением .txt; опции.

После нажатия кнопки «Сохранить» программа переходит к строке 4. Если пользователь ввел какое-то имя файла, то происходит запись в файл, а в строку состояния выводится соответствующая надпись. Если ничего не было введено, то функция saveToFile () завершается.

Пусть программа сохраняет содержимое многострочного текстового поля не в текстовый файл, а как веб­ страницу, т.е. в формате .html.

Итак, мы получали текст из текстового поля с помощью метода toPlainText (). В PyQt5 есть замечательный метод toHtml(), который превращает содержимое текстового поля в код html. Сперва мы должны превратить его в виджет QTextEdit. Для этого в файле menu0.py надо исправить строку, объявляющую переменную textPlainEdit. Строка изменится:

с **self.plainTextEdit = QtWidgets.QPlainTextEdit(self.centralwidget)**

на **self.plainTextEdit = QtWidgets.QTextEdit(self.centralwidget).**

Сохраните файл как menul.py. Внешне ничего не изменится, но теперь мы можем применить к виджету метод toHtml (). Сохраните файл menumain0.py как menumain1.py и проведите соответствующие модификации. *Файл интерфейса изменился на menul.py*. В функции saveToFile () четвертый аргумент изменится на "HTML Files (\* . html) ". Строка, которая производит запись в файл, примет вид: **self.writeFile.write(self.ui.plainTextEdit.toHtml())** (см. рис. 6.)

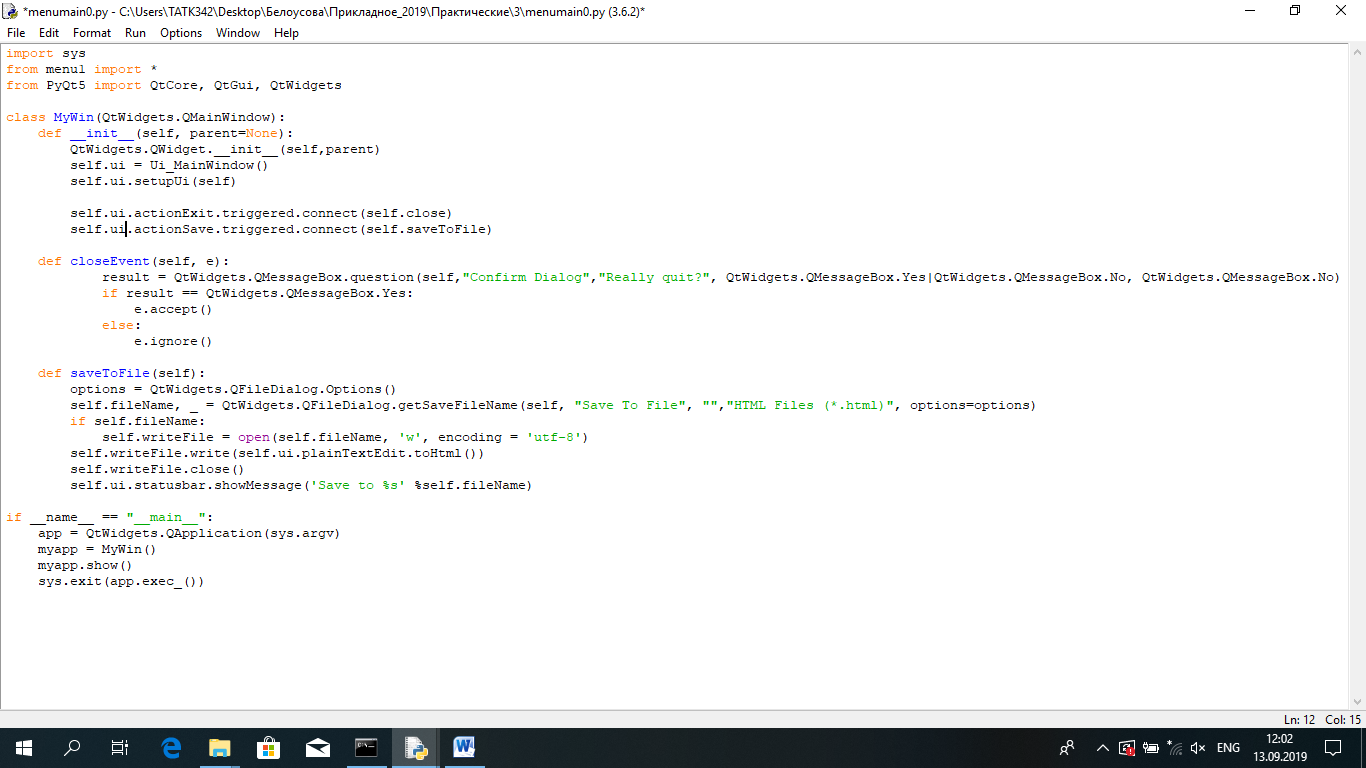


Рисунок 6. Изменение в функции SaveToFile

Запустите программу. Напишите что-нибудь в текстовом поле и сохраните содержимое при помощи пункта меню Save и откройте файл в браузере.

Обратите внимание на то, что PyQt5 преобразовал текст в веб­страницу. В браузере Google Chrome, нажмите правой кнопкой мыши на поле браузера и выберите в появившемся контекстном меню пункт «Просмотр кода страницы». Вы увидите, как PyQt5 преобразовал текст.

**Самостоятельно:**

* Модифицируйте программу menumain1.py так, чтобы вместо диалогового окна, выводилось диалоговое окно с текстом Save text before quit? и с тремя кнопками: Yes, No, Cancel. При нажатии Yes будет выводиться диалоговое окно сохранения в текстовый файл, а затем последует выход из программы. При нажатии No программа просто завершится. Нажатие на Cancel просто вернет пользователя обратно в программу без всяких изменений и сохранений.
* Модифицируйте программу menumain1.py так, чтобы в меню File добавился пункт Open, при выборе которого выводилось бы диалоговое окно открытия текстового файла, а содержимое открываемого файла помещалось бы в текстовое поле для редактирования.