**Программные тренажеры - основы**

Напишем программный тренажер, улучшив программы. Задача будет состоять в том, чтобы разработать графическую оболочку, которая бы заполнялась материалом из базы данных и удовлетворяла бы следующим требованиям:

* Все учебное содержание должно загружаться из файла XML.
* Интерфейс должен иметь поле, с полосой прокрутки, в котором будут располагаться вопросы с вариантами ответов в виде кнопок радио.
* Вопросов может быть сколько угодно, вариантов ответов для каждого вопроса может быть произвольное количество (не одинаковое для всех вопросов).
* Всякий раз при загрузке программа должна перемешивать не только вопросы, но и варианты ответов в них, чтобы у пользователя не происходило запоминания расположения правильных ответов.
* Интерфейс должен иметь кнопку, запускающую процесс проверки, результатом которого должен быть процент правильного выполнения всех заданий. Результат должен высчитываться точно, не зависимо от количества вопросов в тренажере.

Для решения задачи нужно разработать структуру файла XML, интерфейс тренажера и исполняющую программу. Интерфейс будет минималистичен (т.к. все заполнение материалом будет происходить в исполняющем файле), включая несколько виджетов (см. рис.1).

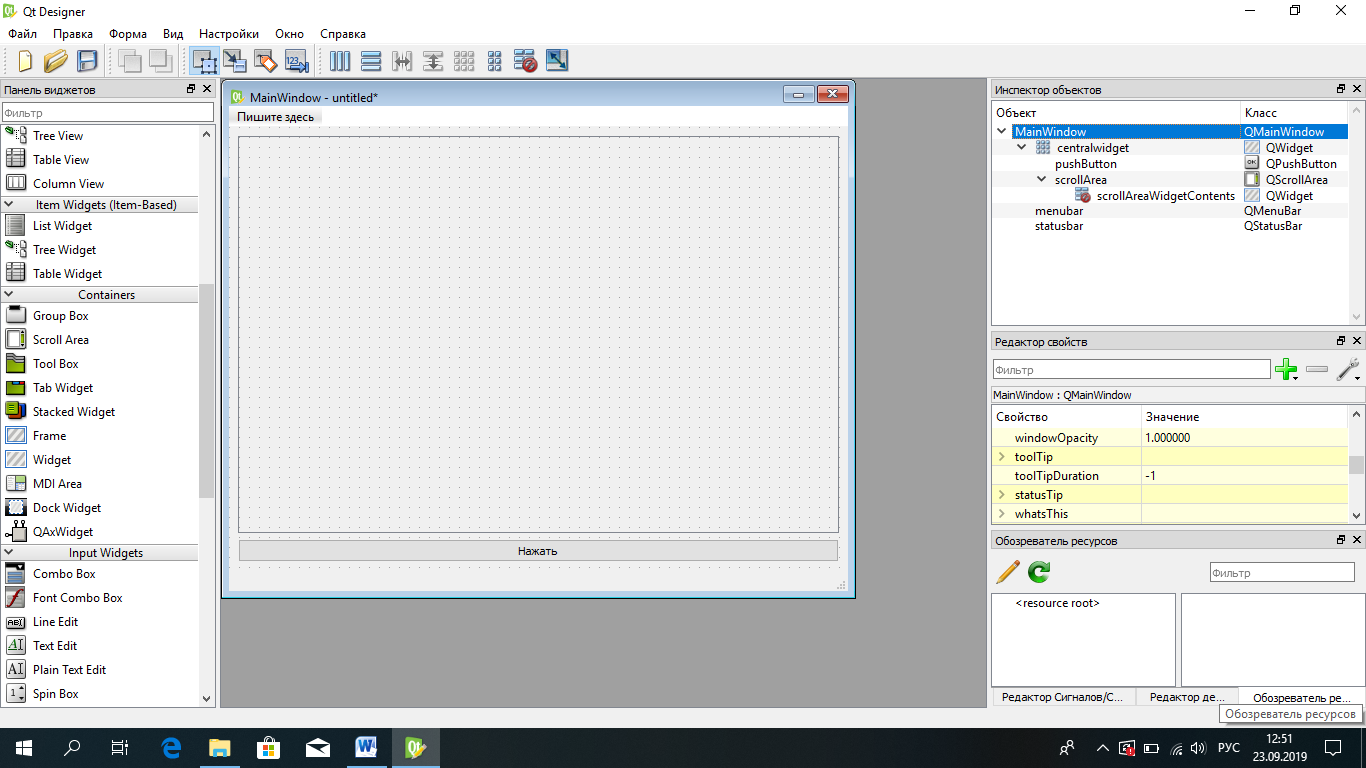


Рисунок 1. Интерфейс программы

Все содержимое файла XML разместится в виджете QScrollArea, который не имеет пока даже никакой компоновки. Кнопка «Нажать» будет запускать процесс проверки, а результат будет выведен в строку состояния.

Теперь создадим файл XML и назовем его bd.xml (см. рис. 2).

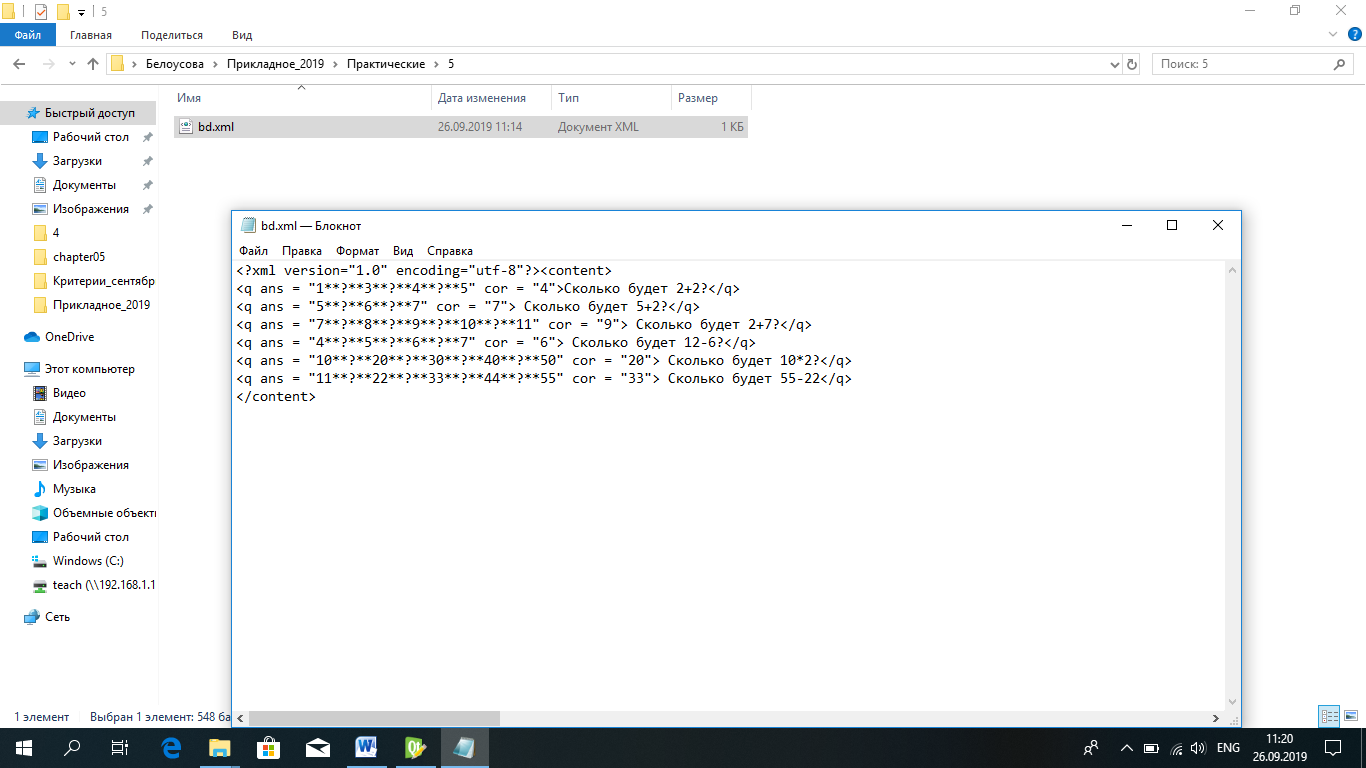
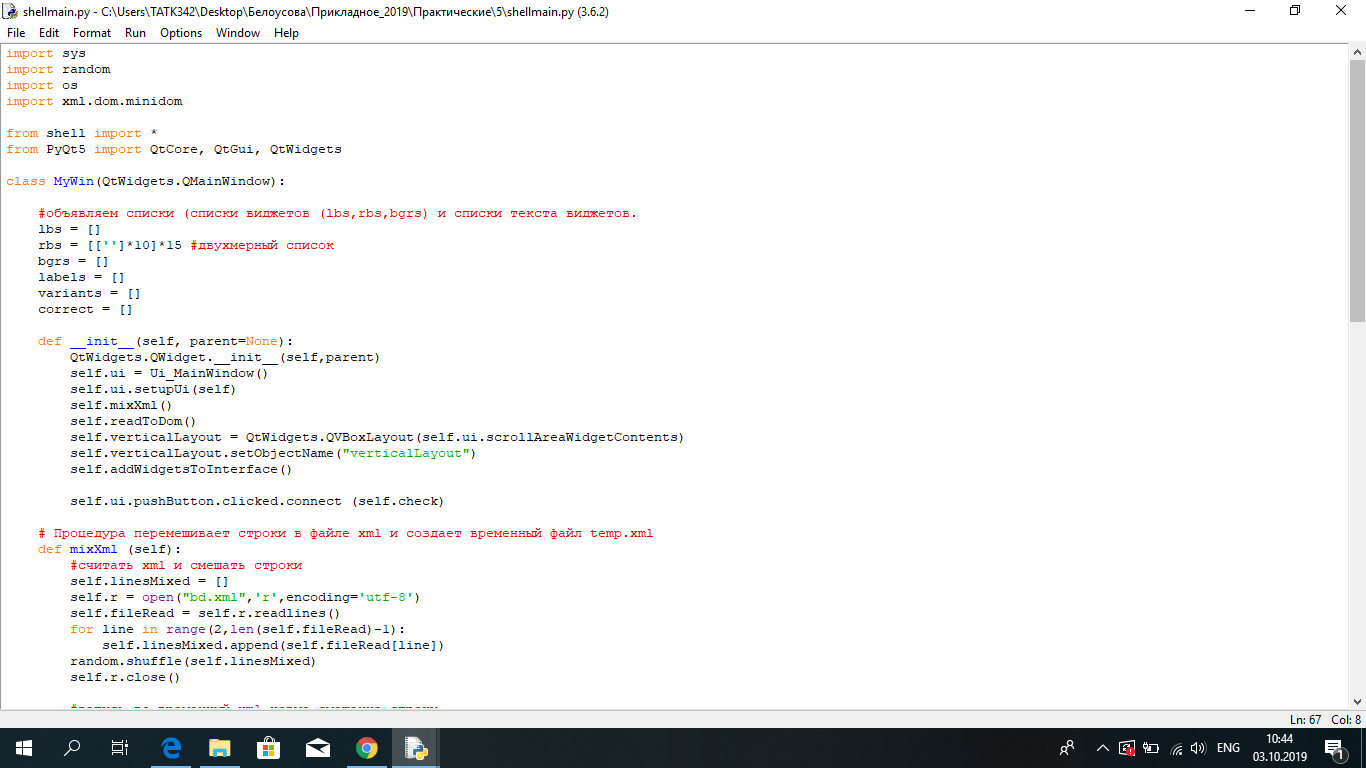
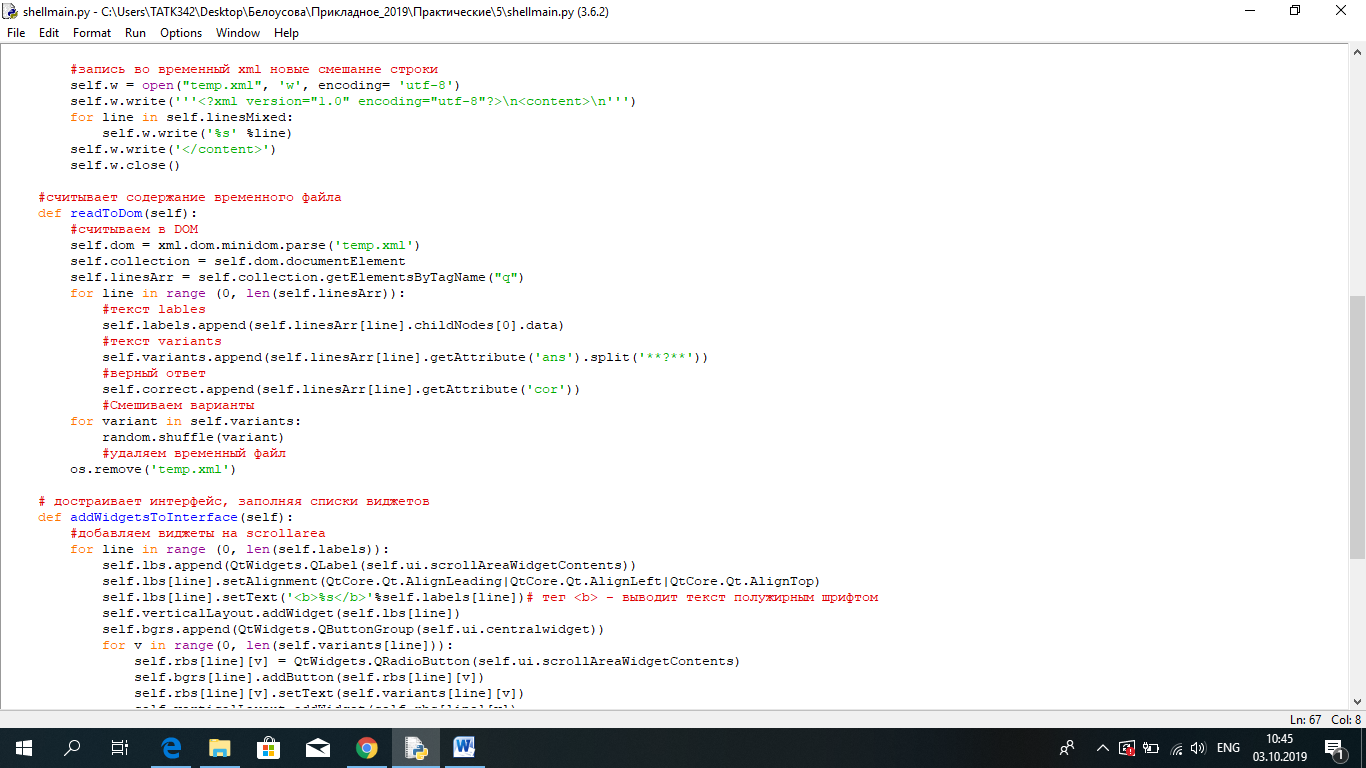


Рисунок 2. Код файла bd.xml

В Python создадим код программы (см. рис. 3).





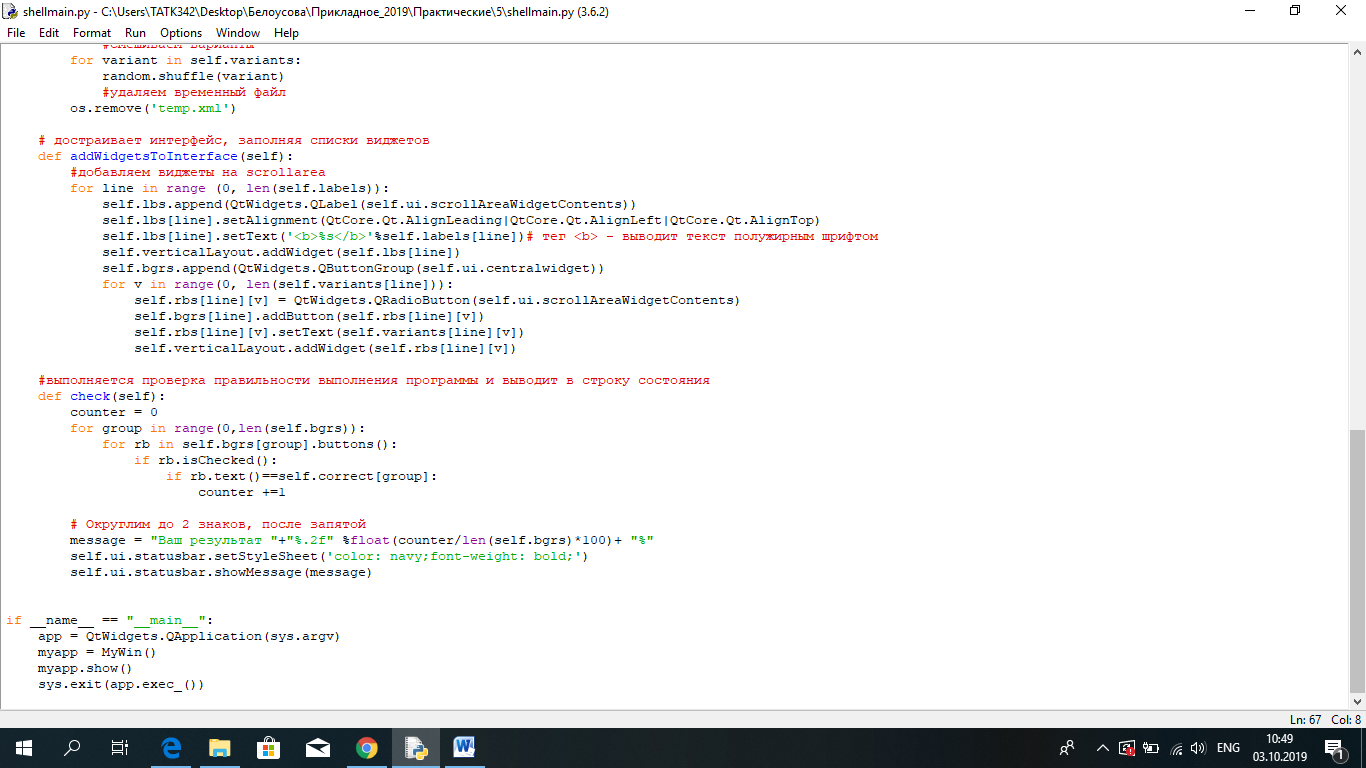


Рисунок 3.Код программы shellmain.py

В процессе работы возникают две сложности. Во-первых, количество вариантов ответов, т.е. кнопок радио, для каждого вопроса разное. Во-вторых, кнопки нельзя просто вывести, их надо организовать в группу кнопок.

Вот алгоритм решения:

1. Запускается цикл for, который переберет все варианты ответов для каждого вопроса.
2. Каждая кнопка радио записывается в двухмерный список, который мы объявили выше. Если представить этот массив как матрицу х\*у, то количество х будет равно количеству вопросов тренажера, а у в каждом х будет разный (равный количеству вариантов ответов для текущего вопроса). Мы заготовили матрицу с запасом, размером 15 на 10.
3. Далее текущая кнопка радио добавляется в группу кнопок, которая была создана выше.
4. Кнопка радио снабжается текстом и выводится на экран.

**Самостоятельно:**

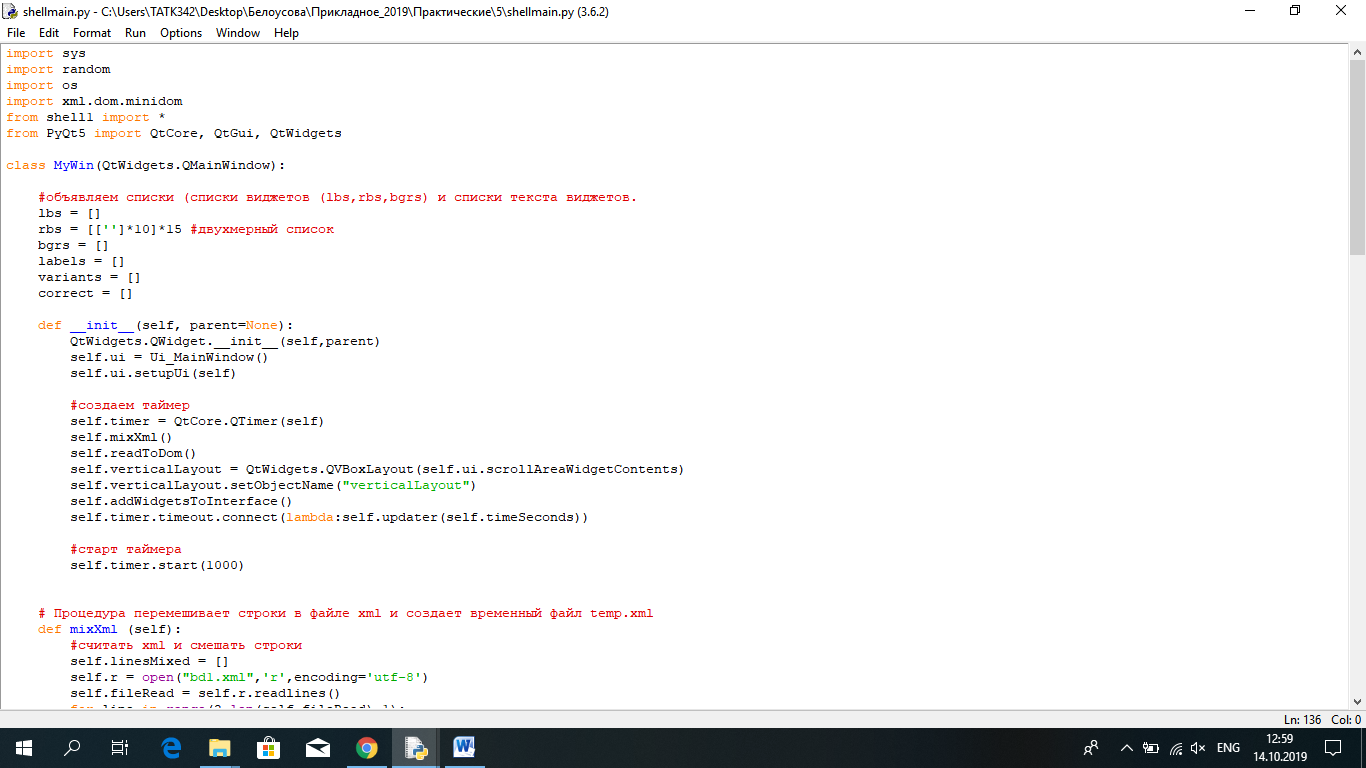
Сделайте так, чтобы цвет строки состояния зависел от количества набранных баллов. Используйте схему: 0-50%: красный цвет, 51-75%: желтый цвет, 76-100%: зеленый цвет.

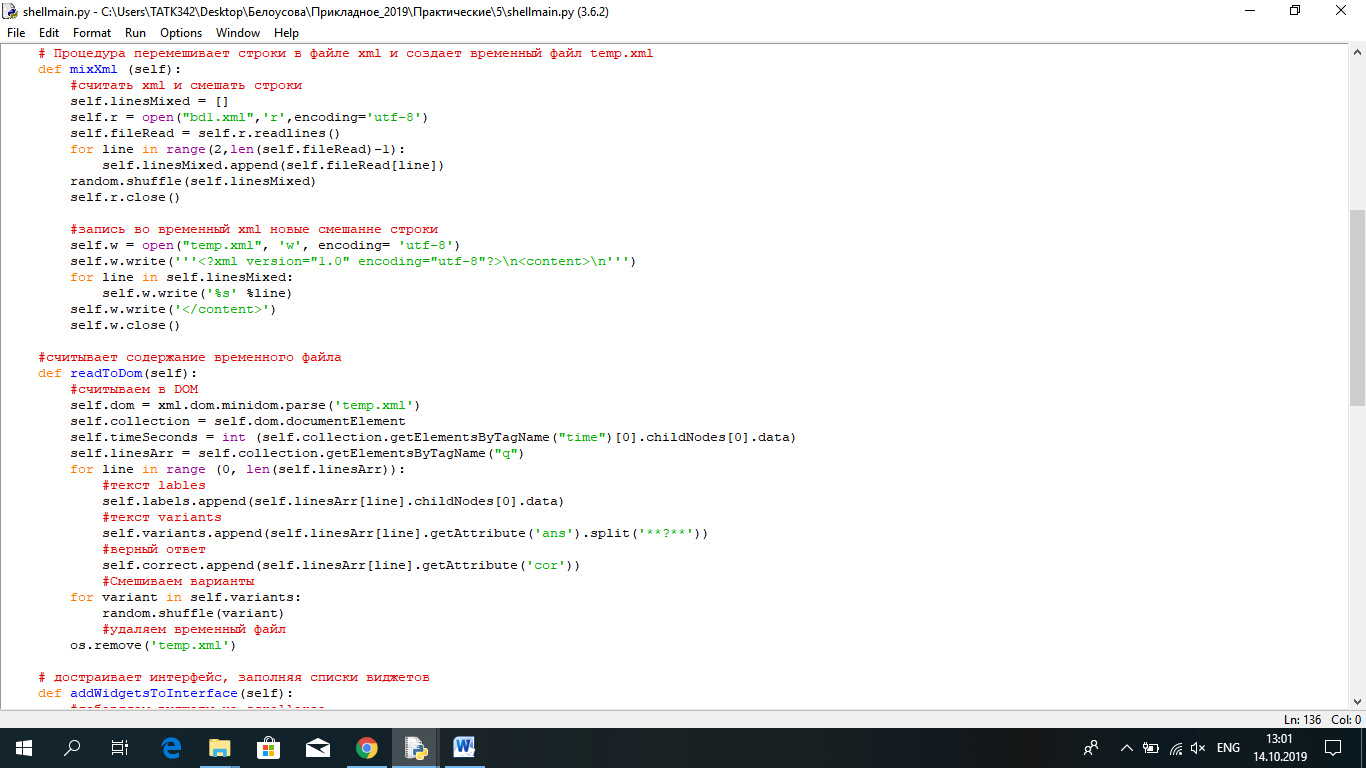
!!! Для хранения кнопок радио был объявлен двухмерный список 15 на 10. Но если пользователь захочет сделать тренажер с 16-ю и более вопросами, то программа выдаст ошибку. Как гибко настроить размер списка, чтобы он всегда соответствовал параметрам базы данных XML?

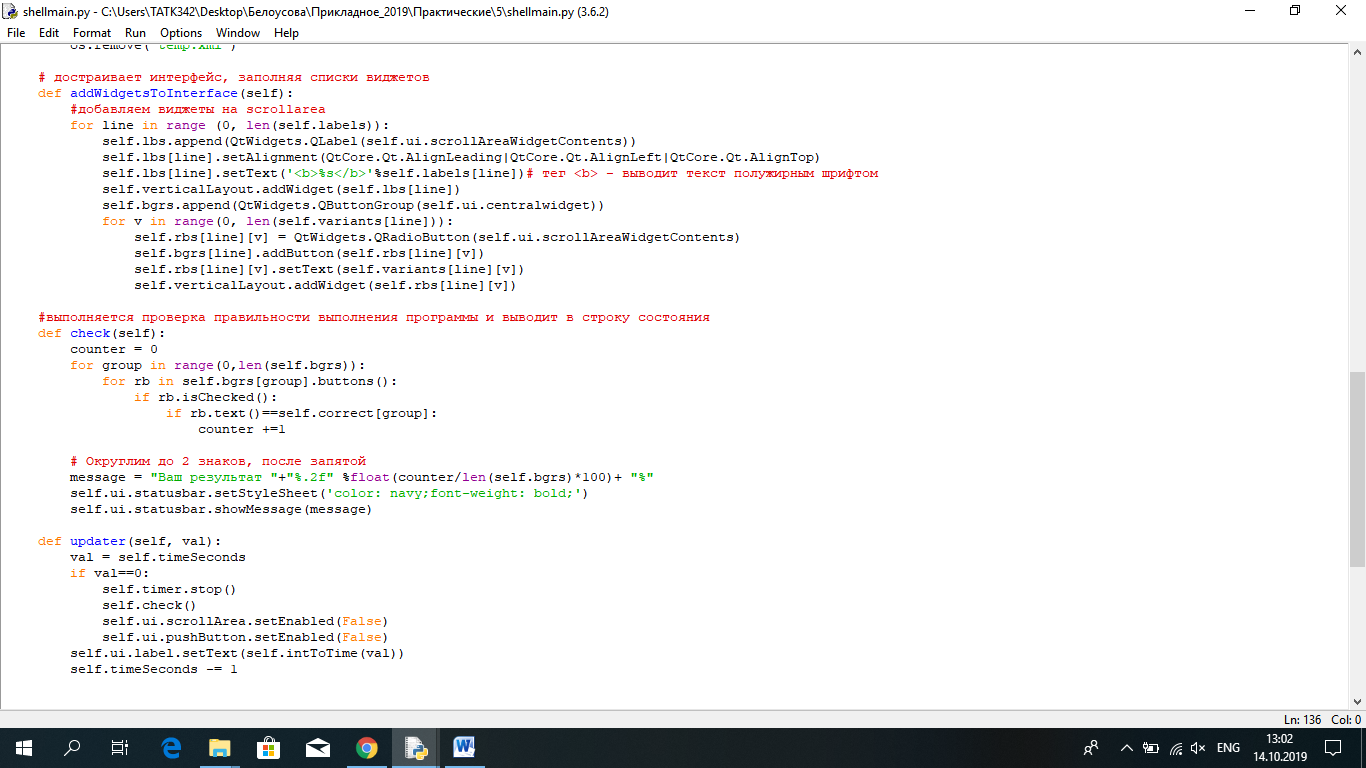
(Решение в том, чтобы перенести объявление списка в конец функции readToDom(), когда точно известно сколько вопросов будет в тренажере).

Важным показателем при проверке работы студента является время выполнения заданий. Данные о времени программа будет получать из базы данных, поэтому нужно ввести еще один дополнительный тег <time>30</time> (после <content>) содержание которого будет означать время на выполнение задания в секундах. Сохраните файл как bd1.xml

В интерфейс добавить надпись (Label), чтобы в ней шел обратный отсчет (файл интерфейса сохранен как shell1.py) (см. рис. 4).







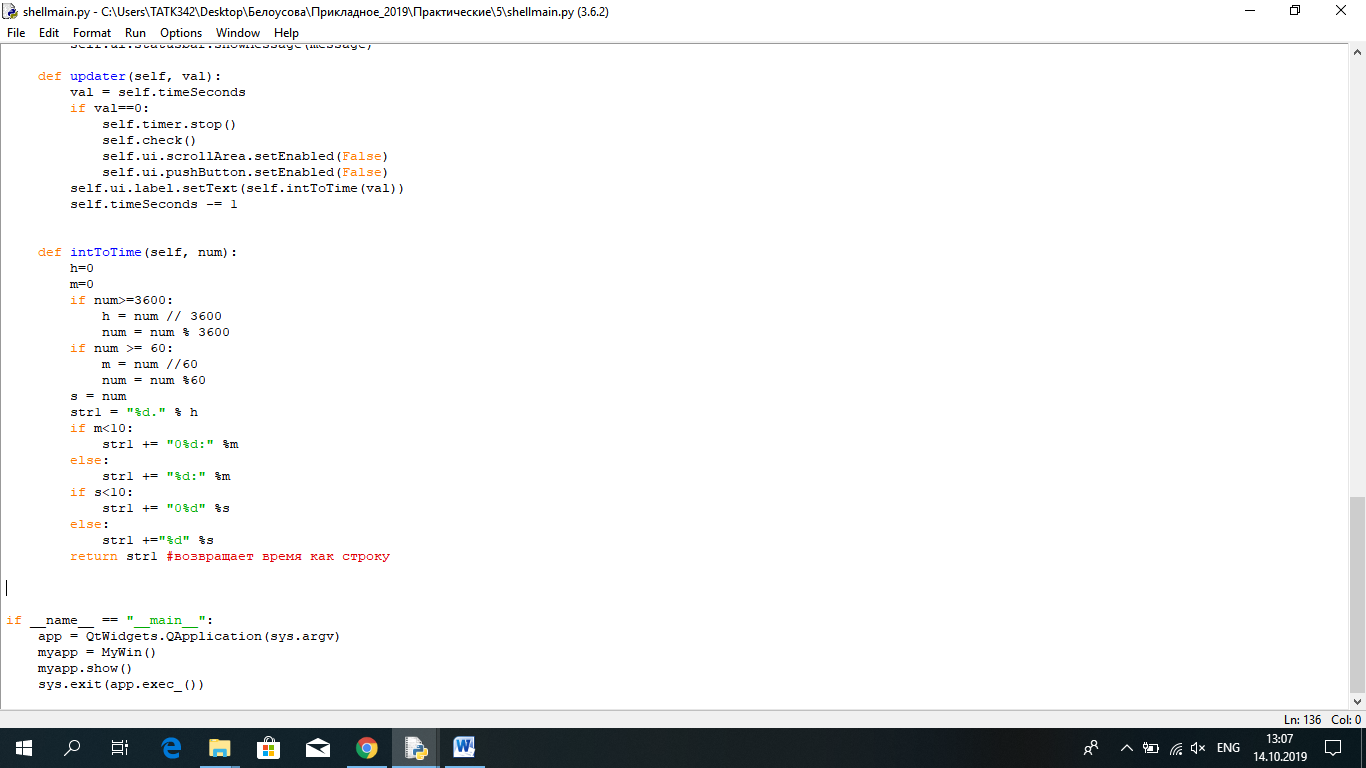


Рисунок 4. Код программы Shellmain.py с функцией time

**Самостоятельно:** В приведенной выше программе таймер обратного отсчета выводится в виде часов, минут и секунд. Модифицируйте программу таким образом, чтобы оставшееся время показывалось в виде уменьшающейся полоски. Используйте виджет QProgressBar. Измените его цвет на серый цвет.

Создается переменная класса QTimer (). С помощью метода timeout () поток таймера связывается с функцией updater (). Метод start() задает интервал в миллисекундах, через который начиная с этого момента будет вызываться функция updater () . Далее при чтении файла XML время таймера считывается в переменную timeSeconds. Некоторым изменениям подверглась функция updater (). При достижении нулевой отметки таймер останавливается, а в конце функции значение переменной timeSeconds уменьшается на единицу.