**Программы учебного назначения**

Интерфейс программы мы сделаем в Qt Designer, и состоять он будет из большого количества виджетов, вложенных друг в друга. Инспектор объектов Qt Designer примет вид (см. рис. 1).

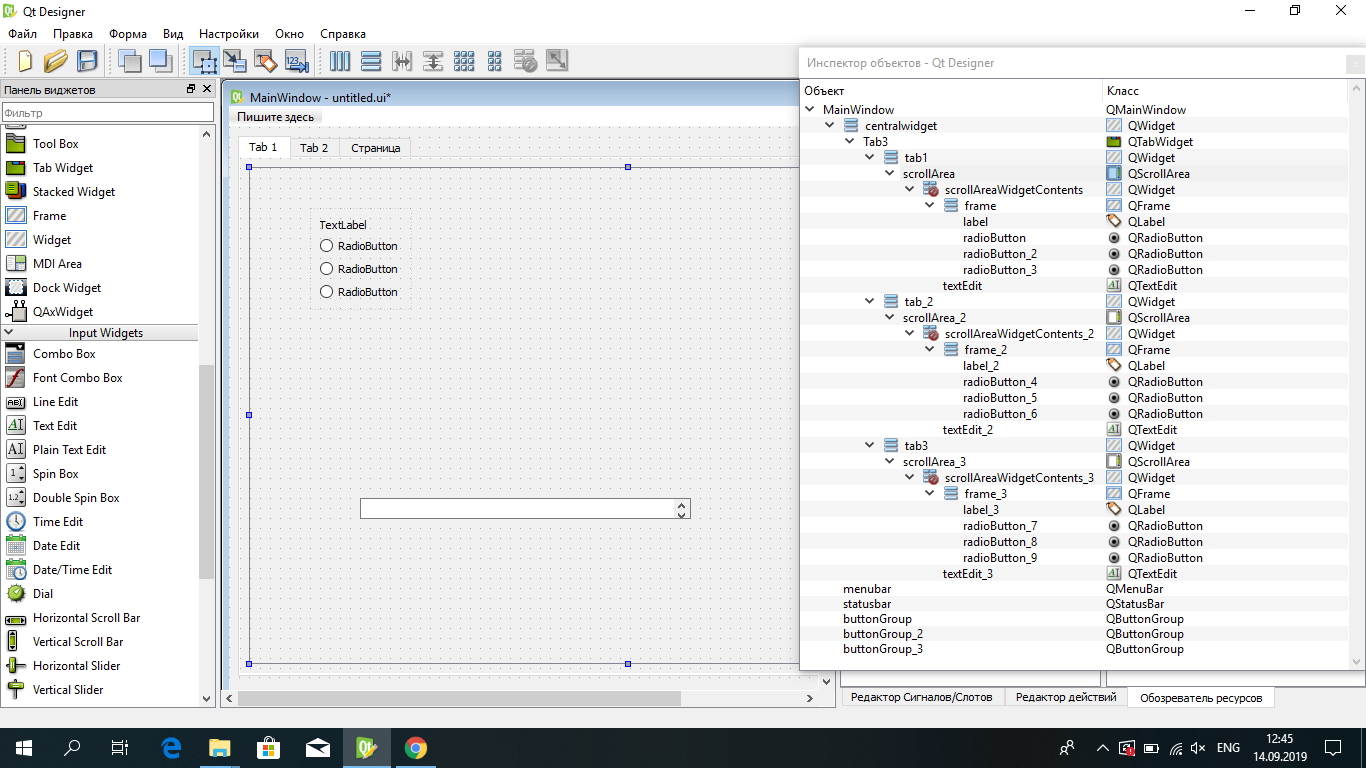


Рисунок 1. Инспектор объектов созданной программы

Обратите внимание на новые виджеты, которые используются в интерфейсе и на их иерархическую структуру. Внутри главного окна все виджеты компонуются по горизонтали. И так происходит со всеми вложенными виджетами, имеющими компоновку. В главное окно помещается **QTabWidget - панель с вкладками**.

Внутрь вкладки помещен виджет **QScrollArea. Его предназначение - быть контейнером для какого-то числа других виджетов.** Даже если в него поместить сотни и тысячи виджетов, то все они будут доступны для пользователя через полосу прокрутки.

Внутри контейнера QScrollArea находятся два виджета: поле ввода QTextArea и еще один контейнер QFrame, в котором находятся четыре виджета: одна надпись и **три кнопки радио**. Для того, чтобы объединить несколько кнопок радио в группу, нужно выделить их и нажать правую кнопку мыши, в появившемся контекстном меню выбрать «Назначить группу кнопок». Разместите 3 любых вопроса (см. рис.2).

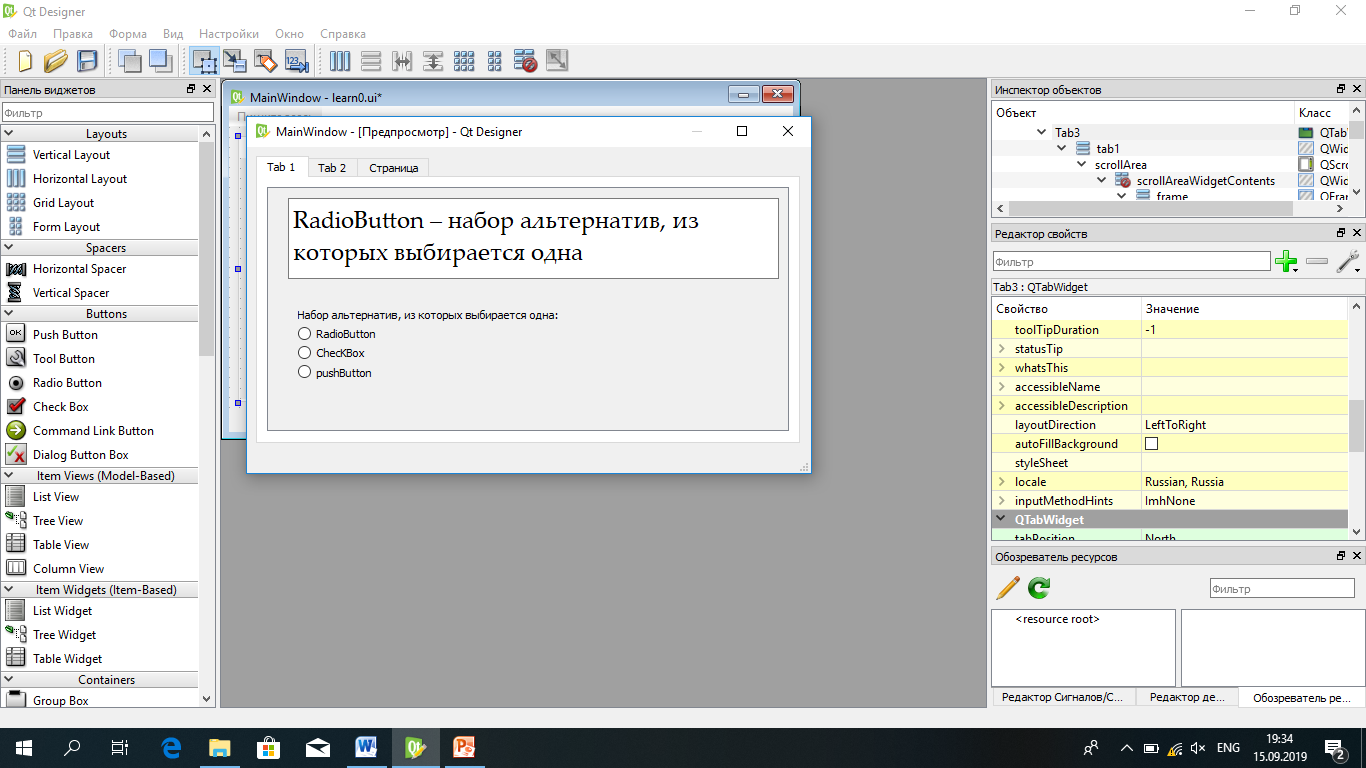


Рисунок 2. Интерфейс программы.

Сохраните интерфейс как learn1.ui и сконвертируйте в learnl.py.

Главным файлом будет learnmainl.py (воспользуйтесь шаблоном). В нем еще нет ни одной функции. , созданием которых нам и предстоит сейчас заняться.

Разумно начинать с первой карточки, затем переходить ко второй, а завершать третьей. Нельзя будет переходить к последующей карточке, если не сделано задание предыдущей. Сделанным оно будет считаться только в случае правильного ответа на задание.

В программе learnmainl.py мы закроем вторую и третью вкладки. Для этого в функцию init () нужно добавить две строки:

**self.ui.ТаbЗ.setTabEnabled(1, False)**

**self.ui.ТаbЗ.setTabEnabled(2, False)**

Метод setTabEnabled() имеет два аргумента: номер вкладки и переменную типа bool, которая включает или выключает доступность вкладки.

При правильном ответе на вопрос первой вкладки станет доступна вторая вкладка.

Напишите функцию (см. рис. 3)

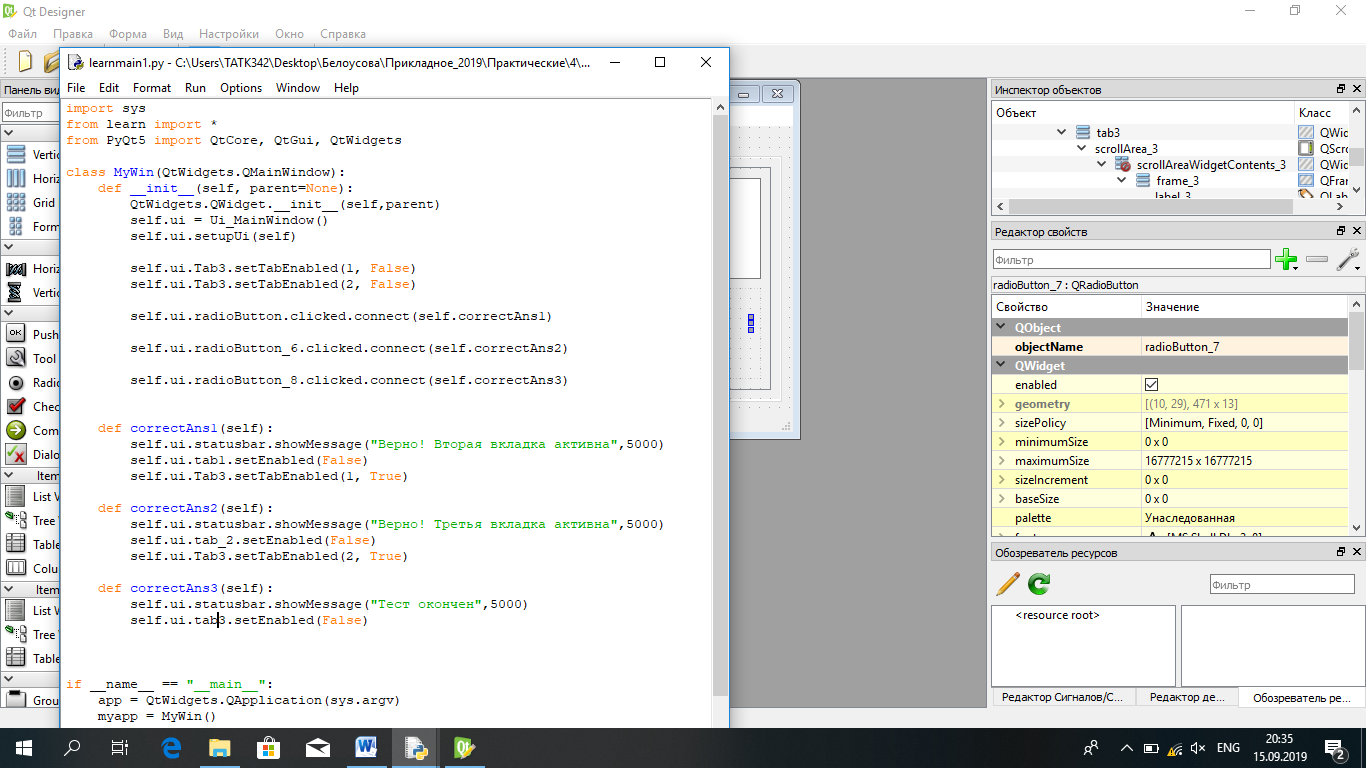


Рисунок 3. Функция активации второй вкладки

Свяжите ее с кнопкой радио radioButton с помощью добавления к функции init () строки: **self.ui.radioButton\_2.clicked.connect(self.correctAnsl).**

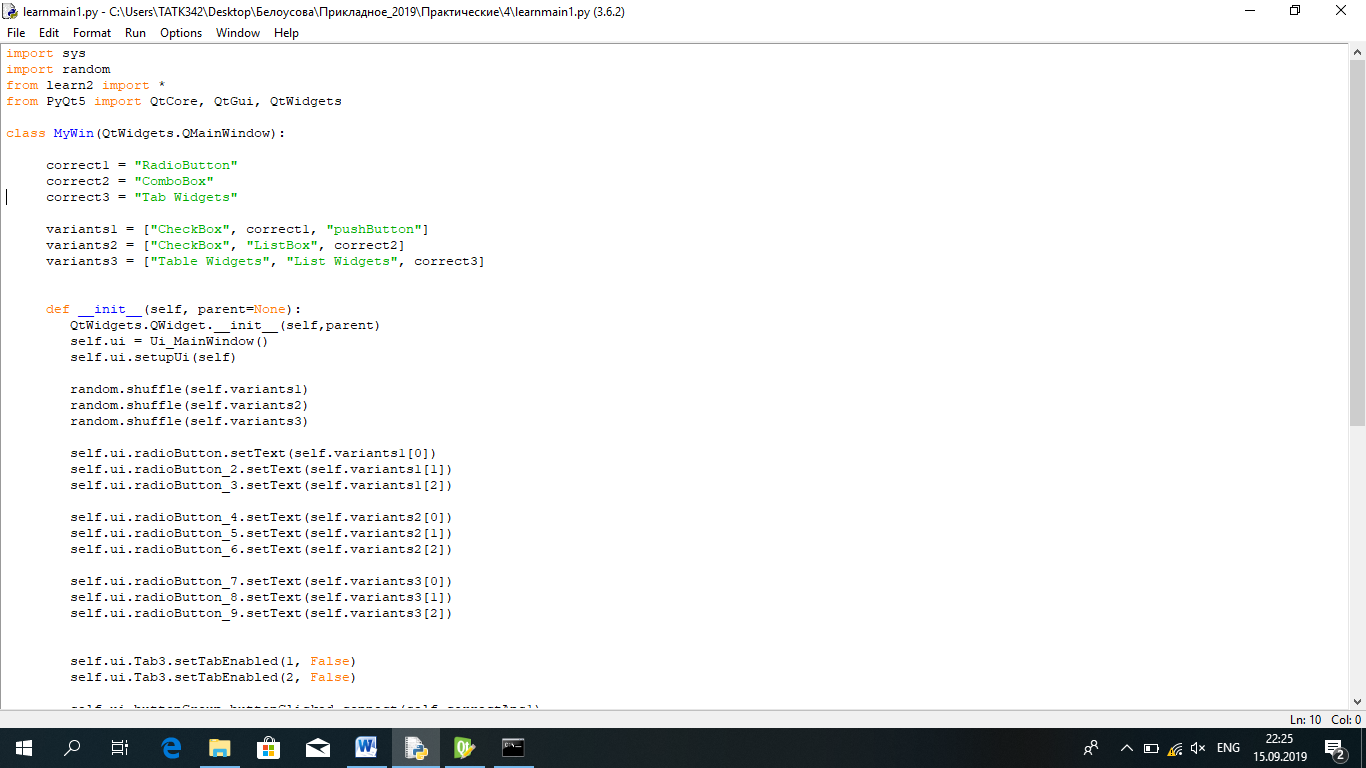
То же самое проделайте и для двух остальных вкладок.

Модифицируем программу learnmainl.py так, чтобы при сохранении функциональности варианты ответов всякий раз появлялись бы в другой последовательности, но программа бы все равно при проверке реагировала бы только на правильный ответ.

Получается, что при каждом запуске программы интерфейс (текст кнопок радио) будет выглядеть по-разному. Поэтому в самом файле интерфейса этот текст можно просто удалить (файл learn2.ру). В исполняющей программе изменяться функции, а также добавятся некоторые переменные. Последовательность действий программы изложена в шести пунктах:

1. Правильные ответы помещаются в три переменные типа String.
2. Варианты ответов помещаются в списки (в каждом по три переменные типа String).
3. Перед построением интерфейса списки перемешиваются в случайном порядке при помощи метода shuffle () модуля random.
4. К радио кнопкам добавляется текст.
5. К группам кнопок привязывается сигнал buttonClicked, по наступлении которого вызывается соответствующая функция.
6. Внутри каждой функции происходит проверка, которая - перебирая все кнопки радио группы кнопок - устанавливает, какая именно кнопка нажата.
7. Если текст этой кнопки соответствует тексту правильного ответа (он хранится в переменной, см. п. 1), то выполняются действия по выводу надписи в строке состояния и т.д.

Получается следующий код (см. рис.4).



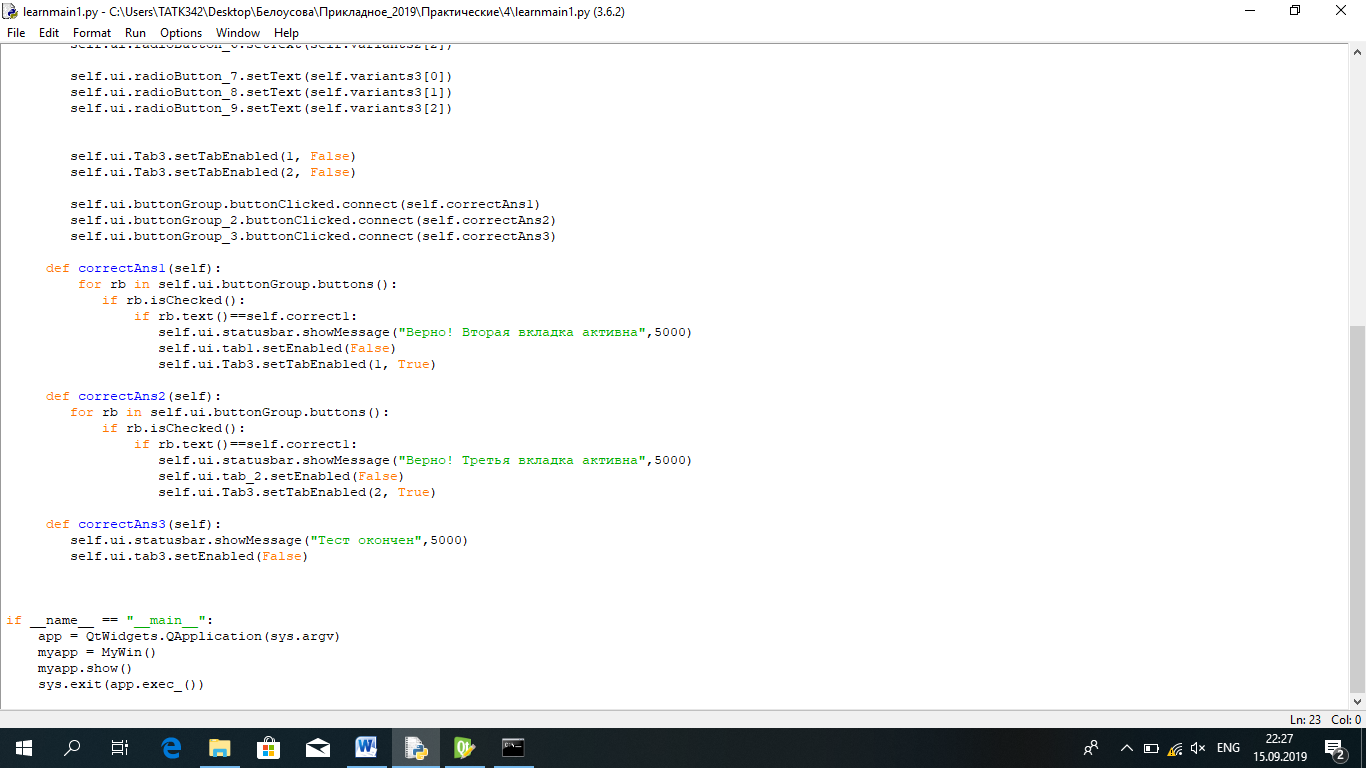


Рисунок 4. Код программы теста

Проведем еще одну важную модификацию. Вынесем все учебное содержание (текст, формулировку вопроса и варианты ответов) в отдельный файл. Таким образом, сама программа будет состоять только из пустого интерфейса и исполняющей программы, которая сразу после запуска будет считывать файл и работать дальше в соответствии с его содержимым. Получится, что меняя файлы содержания, мы каждый раз будем получать новые учебные материалы, не меняя ничего в программе.

Важно выбрать тип файла для хранения учебного наполнения. Мы сделаем выбор в пользу файла типа XML. В файле с XML важно знать два правила:

1. ничего не должно быть вне тегов;
2. теги могут иметь атрибуты.

Рассмотрите код и создайте файл learn.xml (см.рис.5) .

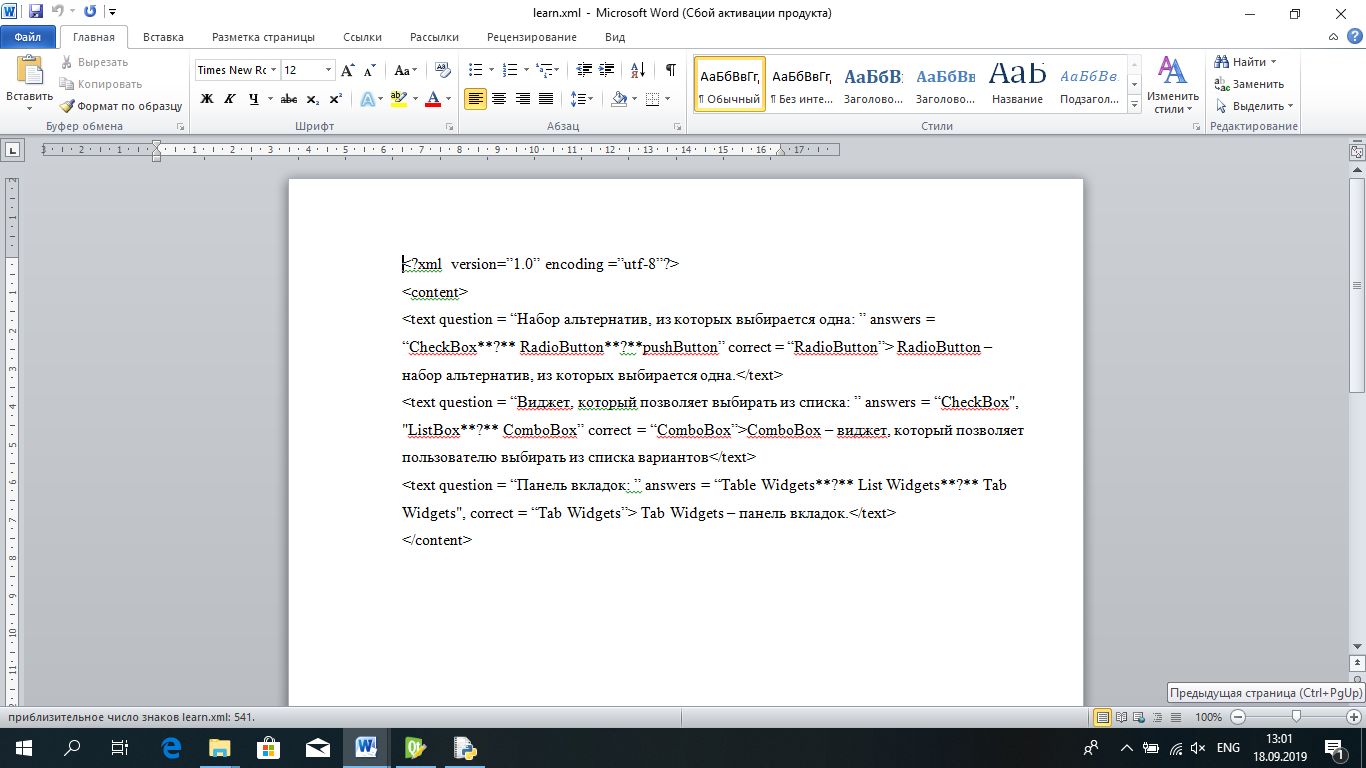
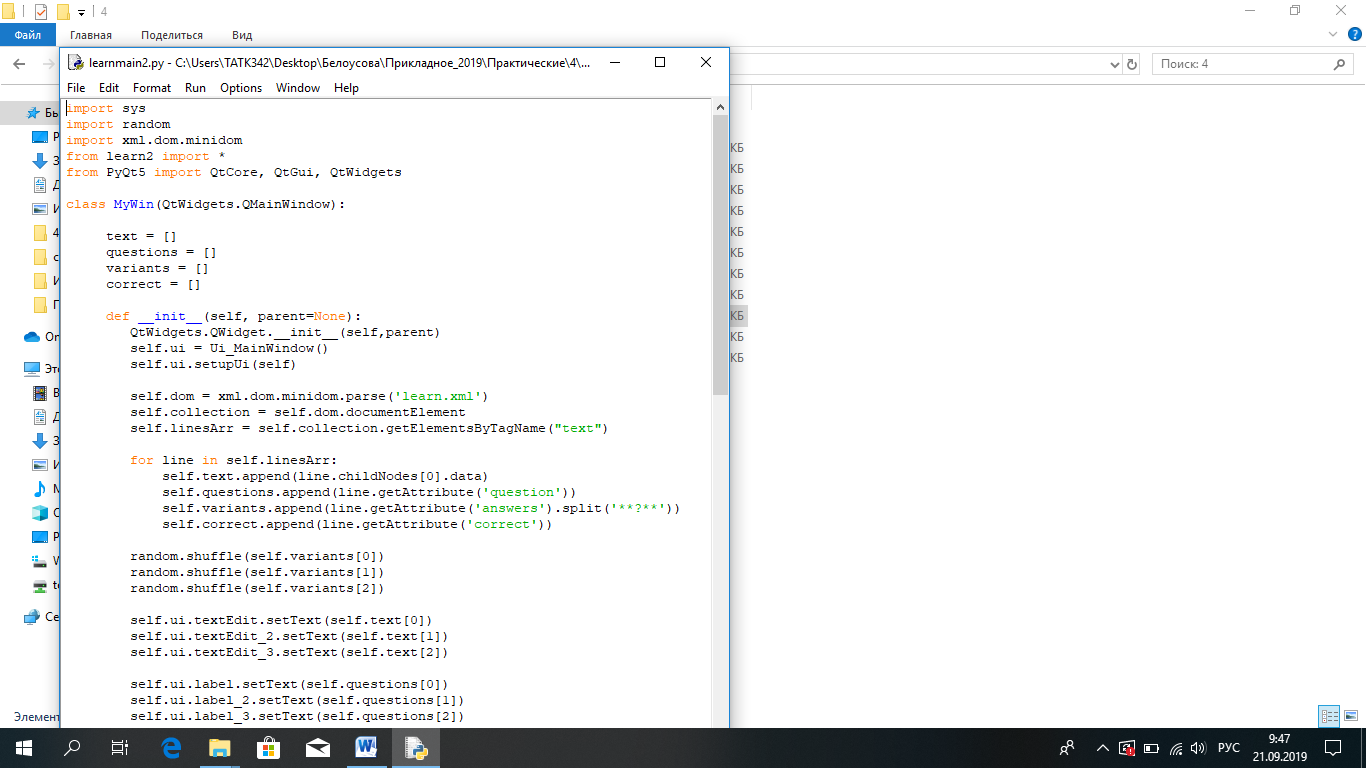
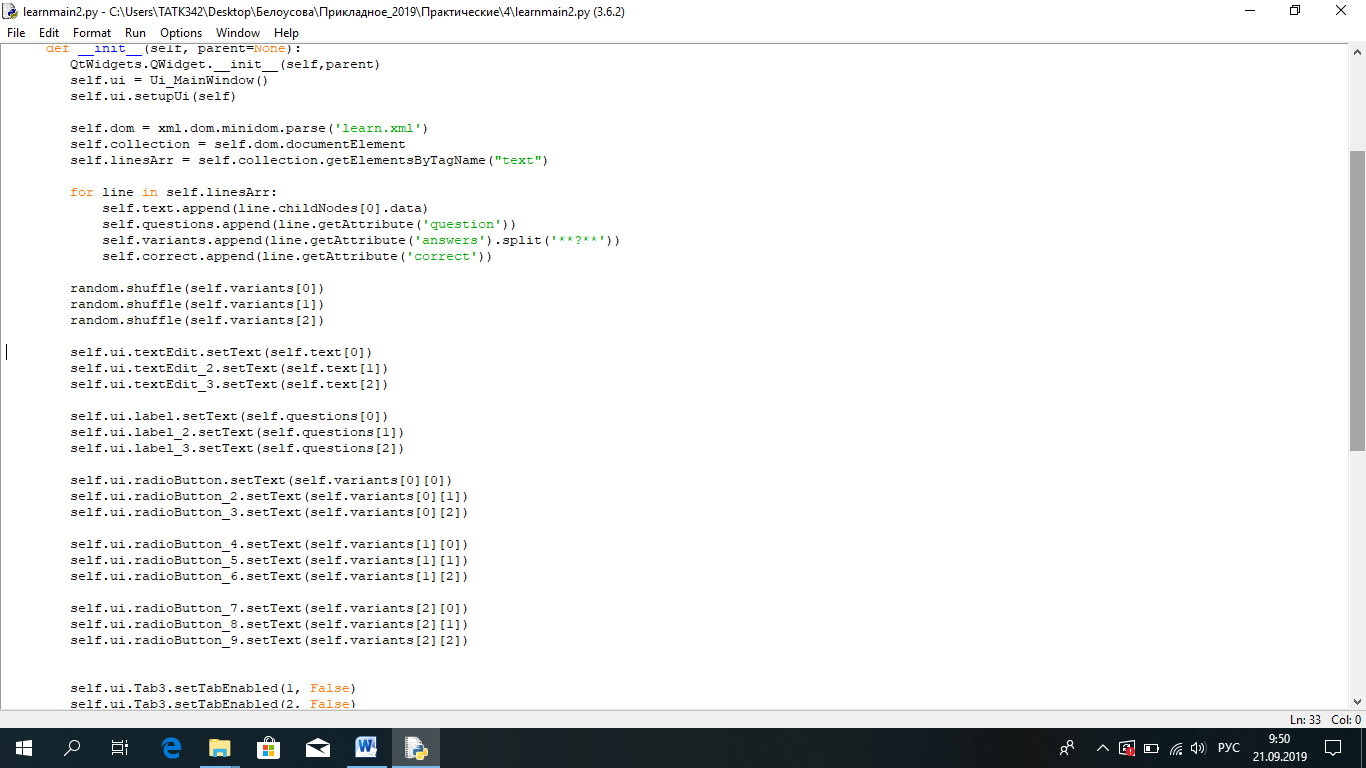


Рисунок 5. Код файла learn.xml

Строка 1 является информационной, из нее компьютер понимает, что перед ним файл XML, имеющий кодировку utf-8. Строки 2 и 6 содержат соответственно открывающий и закрывающий тег content. В этом теге находится собственно содержание базы данных, которое распределено между трех тегов text. Содержанием тега является текст для текстового поля (теоретическая информация), а в атрибутах находятся вопрос (question), варианты ответов (answers) и правильный ответ (correct). Обратите особое внимание на то, как записаны варианты ответов. Они являются одним атрибутом, но разделены странной последовательностью символов \*\*?\*\*. Странной она сделана намеренно, чтобы программа потом легко смогла отделить один вариант ответа от другого. Для того, чтобы разделение было всегда корректным, выбираются такие сочетания символов, которые не могут встретиться в естественном языке.

Теперь всю информацию (текст, вопросы, варианты ответов) из файла интерфейса можно убрать (см.рис.6).





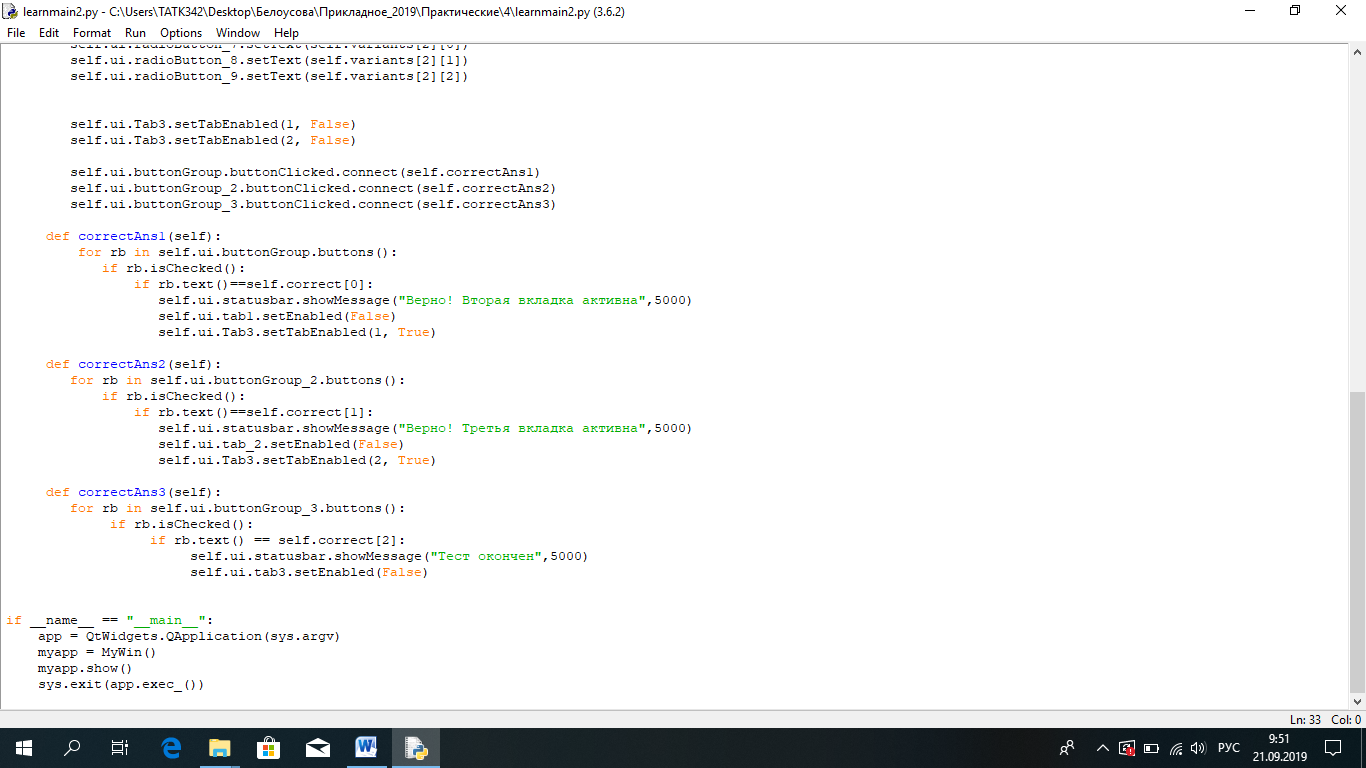


Рисунок 5. Код программы learn2.py

Теперь, изменяя файл XML, мы будем получать разные учебные материалы, впрочем строго ограниченные по структуре. При таком алгоритме мы не можем поменять число вариантов ответов или количество вопросов внутри вкладки.

**Самостоятельно:**

Создайте свой тест по вариантам состоящий из 10 вопросов (можно брать данные из файла XML, можно без создания файла.