

# Цель работы

Целью данной работы является разработка приложения для автоматического заполнения docx документов согласно шаблонам для последующей загрузки в редактор контента для создания упражнений на платформу для изучения английского языка.

# Задание на работу

# Изучить библиотеки для работы с docx документами на разных языках и выбрать наиболее удобную.

# Разработать экранное приложение с использованием выбранной библиотеки.

# Данные для разработки приложения

# Инструкция по разметке docx документов и шаблоны docx документов в Google drive <https://drive.google.com/drive/folders/1bQBf_3AttrEC9MQDIvUS0727gvcAb-Kl?usp=sharing>.

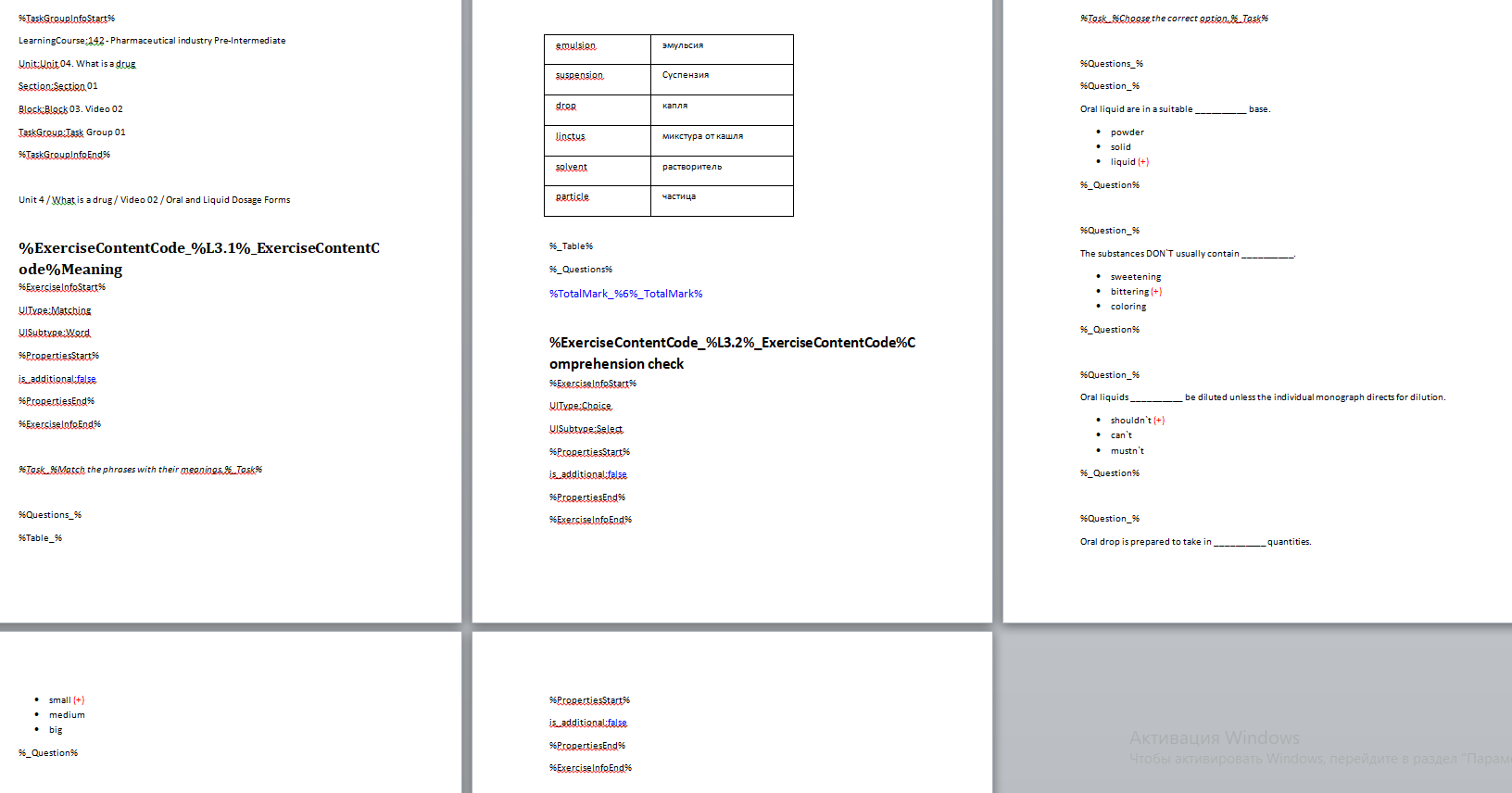
# Исходные данные

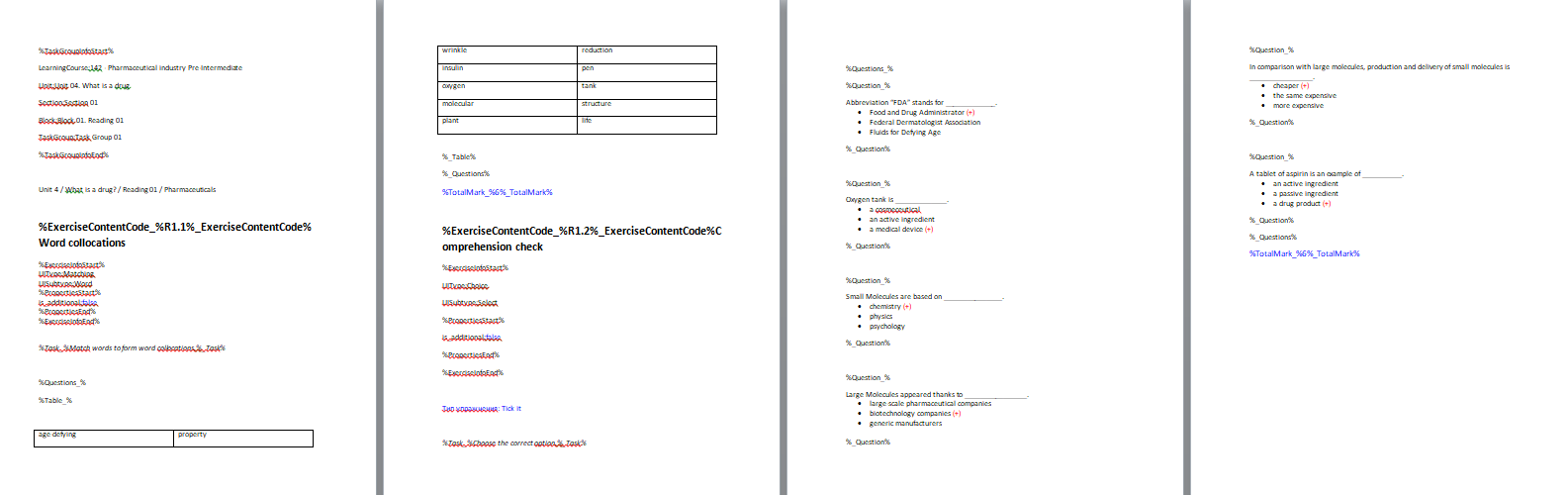
# Exel таблица с описание упражнений и docx документы с текстами упражнений по английскому языку.

# Выходные данные

# Docx документы, пригодные для конвертации в упражнения на платформе JetClass.

# Результат работы приложения





# Код приложения

Файл main.py

import sys

import os

from PyQt6 import QtGui, uic

from PyQt6 import QtWidgets

from docx import Document

from docx.shared import Pt

from docx.shared import RGBColor

import design

class Word(QtWidgets.QMainWindow, design.Ui\_Word):

def *\_\_init\_\_*(self):

super(Word, self).*\_\_init\_\_*()

self.setupUi(self)

self.CreateDoc.clicked.connect(self.Create\_Document)

self.ExerciseType.currentTextChanged.connect(self.Change\_Subtype)

self.Createexec.clicked.connect(self.Create\_exercize)

self.Finishblock.clicked.connect(self.Finish\_block)

self.CreateQuestion.clicked.connect(self.Create\_Question)

self.CreateAnswer.clicked.connect(self.Create\_Answer)

self.FinishQuestion.clicked.connect(self.Finish\_Question)

self.FinishExercize.clicked.connect(self.Finish\_Exercize)

self.doc=Document()

self.exercize\_count=1

self.question\_count=0

self.UnitNum=""

self.BlockNum=""

self.BlockNumTotal=""

self.UIType=""

self.UISubtype=""

self.paragraph=""

self.answertext=""

self.counter=0

self.line=1

# Создаёт шапку документа

def Create\_Document(self):

if(len(self.UnitNumber.text())<2):

self.UnitNum="0"+self.UnitNumber.text()

else:

self.UnitNum=self.UnitNumber.text()

if(len(self.BlockNumber.text())<2):

self.BlockNum="0"+self.BlockNumber.text()

else:

self.BlockNum=self.BlockNumber.text()

if(len(self.BlockNumberTotal.text())<2):

self.BlockNumTotal="0"+self.BlockNumberTotal.text()

else:

self.BlockNumTotal=self.BlockNumberTotal.text()

paragraph = self.doc.add\_paragraph()

paragraph.style.font.size = Pt(12)

paragraph.style.font.name = 'Calibri'

run = paragraph.add\_run("%TaskGroupInfoStart%")

run = self.doc.add\_paragraph().add\_run("LearningCourse:"+self.CourseNumber.text()+" - "+self.CourseName.text())

run = self.doc.add\_paragraph().add\_run("Unit:Unit "+self.UnitNum+". "+self.UnitName.text())

run = self.doc.add\_paragraph().add\_run("Section:Section 01")

run = self.doc.add\_paragraph().add\_run("Block:Block "+self.BlockNumTotal+". "+self.BlockType.currentText()+" "+self.BlockNum)

run = self.doc.add\_paragraph().add\_run("TaskGroup:Task Group 01")

run = self.doc.add\_paragraph().add\_run("%TaskGroupInfoEnd%")

self.doc.add\_paragraph()

self.doc.add\_paragraph("Unit "+self.UnitNumber.text()+" / "+self.UnitName.text()+" / "+self.BlockType.currentText()

+" "+self.BlockNum+" / "+self.BlockName.text())

def Change\_Subtype(self):

while(self.ExerciseSubType.count()>0):

self.ExerciseSubType.removeItem(0)

if(self.ExerciseType.currentText()=='Tick it'):

self.ExerciseSubType.addItem("Правильный ответ один")

self.ExerciseSubType.addItem("Правильных ответов несколько")

elif(self.ExerciseType.currentText()=='Drag and drop'):

self.ExerciseSubType.addItem("Единый текст")

self.ExerciseSubType.addItem("Нумерованные предложения")

self.ExerciseSubType.addItem("Нумерованные предложения, лишние слова")

def Create\_exercize(self):

self.question\_count=0

if (self.BlockType.currentText()=="Reading"):

BlockType="R"

elif (self.BlockType.currentText()=="Video"):

BlockType="L"

elif (self.BlockType.currentText()=="Grammar"):

BlockType="G"

elif (self.BlockType.currentText()=="Vocabulary"):

BlockType="V"

elif (self.BlockType.currentText()=="Writing"):

BlockType="W"

paragraph = self.doc.add\_paragraph()

run = self.doc.add\_paragraph(style="Heading 2").add\_run("%ExerciseContentCode\_%"+BlockType+self.BlockNumberTotal.text()+"."+str(self.exercize\_count)+

"%\_ExerciseContentCode%"+self.ExerciseName.text())

run.font.size = Pt(18)

run.font.name = 'Calibri'

run.font.bold = True

run.font.color.rgb=RGBColor(0x00, 0x00, 0x00)

self.exercize\_count=self.exercize\_count+1

self.CreateQuestion.setEnabled(True)

self.CreateAnswer.setEnabled(True)

self.FinishQuestion.setEnabled(True)

if(self.ExerciseType.currentText()=='Tick it'):

self.UIType="Choice"

self.UISubtype="Radioline"

elif(self.ExerciseType.currentText()=='Writing'):

self.UIType="Writing"

self.UISubtype="Plain"

elif(self.ExerciseType.currentText()=='Speaking'):

self.UIType="Speaking"

self.UISubtype="Plain"

self.CreateQuestion.setEnabled(False)

self.CreateAnswer.setEnabled(False)

self.FinishQuestion.setEnabled(False)

elif(self.ExerciseType.currentText()=='Essay'):

self.UIType="Essay"

self.UISubtype="Essay"

self.CreateQuestion.setEnabled(False)

self.CreateAnswer.setEnabled(False)

self.FinishQuestion.setEnabled(False)

elif(self.ExerciseType.currentText()=='Matching'):

self.UIType="Matching"

self.UISubtype="Word"

self.CreateQuestion.setEnabled(False)

self.CreateAnswer.setEnabled(False)

self.FinishQuestion.setEnabled(False)

elif(self.ExerciseType.currentText()=='Categories'):

self.UIType="Moving"

self.UISubtype="Table"

self.CreateQuestion.setEnabled(False)

self.CreateAnswer.setEnabled(False)

self.FinishQuestion.setEnabled(False)

elif(self.ExerciseType.currentText()=='Drag and drop'):

self.UIType="Moving"

self.UISubtype="Inline"

self.counter=0

if(self.ExerciseSubType.currentText()!="Нумерованные предложения, лишние слова"):

self.FinishQuestion.setEnabled(False)

elif(self.ExerciseType.currentText()=='Multiple choice'):

self.UIType="Choice"

self.UISubtype="Select"

run = self.doc.add\_paragraph().add\_run("%ExerciseInfoStart%")

run = self.doc.add\_paragraph().add\_run("UIType:"+self.UIType)

run = self.doc.add\_paragraph().add\_run("UISubtype:"+self.UISubtype)

run = self.doc.add\_paragraph().add\_run("%PropertiesStart%")

paragraph = self.doc.add\_paragraph()

run = paragraph.add\_run("is\_additional:")

run = paragraph.add\_run(self.Isadditional.currentText())

run.font.color.rgb=RGBColor(0x00, 0x00, 0xff)

if(self.ExerciseType.currentText()=='Tick it'):

if(self.ExerciseSubType.currentText()=='Правильных ответов несколько'):

run = paragraph.add\_run("multiple\_answers:")

run = paragraph.add\_run("true")

run.font.color.rgb=RGBColor(0x00, 0x00, 0xff)

if(self.ExerciseType.currentText()=='Categories'):

run = paragraph.add\_run("dont\_check\_places:")

run = paragraph.add\_run("true")

run.font.color.rgb=RGBColor(0x00, 0x00, 0xff)

if(self.ExerciseType.currentText()=='Drag and drop'):

if(self.ExerciseSubType.currentText()=='Нумерованные предложения, лишние слова'):

run = paragraph.add\_run("show\_possible\_answers\_for\_each\_question:")

run = paragraph.add\_run("true")

run.font.color.rgb=RGBColor(0x00, 0x00, 0xff)

elif(self.ExerciseSubType.currentText()=='Единый текст'):

run = paragraph.add\_run("display\_questions\_as\_plain\_text:")

run = paragraph.add\_run("true")

run.font.color.rgb=RGBColor(0x00, 0x00, 0xff)

run = paragraph.add\_run("show\_answer\_places::")

run = paragraph.add\_run("true")

run.font.color.rgb=RGBColor(0x00, 0x00, 0xff)

run = self.doc.add\_paragraph().add\_run("%PropertiesEnd%")

run = self.doc.add\_paragraph().add\_run("%ExerciseInfoEnd%")

self.doc.add\_paragraph()

paragraph = self.doc.add\_paragraph()

run = paragraph.add\_run("%Task\_%"+self.ExerciseText.toPlainText()+"%\_Task%")

run.font.italic=True

run.font.cs\_italic=True

paragraph = self.doc.add\_paragraph()

run = self.doc.add\_paragraph().add\_run("%Questions\_%")

if(self.ExerciseType.currentText()=='Matching'):

run = self.doc.add\_paragraph().add\_run("%Table\_%")

run = self.doc.add\_paragraph()

elif(self.ExerciseType.currentText()=='Categories'):

run = self.doc.add\_paragraph().add\_run("%Table\_%")

run = self.doc.add\_paragraph()

while(self.AnswerType.count()>0):

self.AnswerType.removeItem(0)

if(self.UIType=='Choice'):

self.AnswerType.addItem("Wrong")

self.AnswerType.addItem("Right")

def Create\_Question(self):

tempstr=self.Question.toPlainText()

self.TotalMark.insert(str(len(tempstr)))

for i in range(len(tempstr)):

if(tempstr[i-1]=="'"):

tempstr=tempstr[:i-1]+"`"+tempstr[i:]

self.Question.setPlainText(tempstr)

if(self.UIType=='Choice'):

if(self.AnswerType.count()==1):

self.AnswerType.addItem("Right")

if(self.UIType=='Choice'):

if(self.question\_count!=0):

self.doc.add\_paragraph()

else:

self.question\_count=self.question\_count+1

run = self.doc.add\_paragraph().add\_run("%Question\_%")

run = self.doc.add\_paragraph().add\_run(self.Question.toPlainText())

elif(self.ExerciseType.currentText()=='Drag and drop'):

if(self.ExerciseSubType.currentText()=='Нумерованные предложения, лишние слова'):

if(self.counter==0):

if(self.question\_count!=0):

self.doc.add\_paragraph()

else:

self.question\_count=self.question\_count+1

run = self.doc.add\_paragraph().add\_run("%Question\_%")

self.counter=self.counter+1

else:

self.counter=0

self.paragraph=self.doc.add\_paragraph(self.Question.toPlainText())

else:

self.paragraph=self.doc.add\_paragraph(self.Question.toPlainText())

elif(self.ExerciseType.currentText()=='Writing'):

if(self.question\_count!=0):

self.doc.add\_paragraph()

else:

self.question\_count=self.question\_count+1

run = self.doc.add\_paragraph().add\_run("%Question\_%")

self.paragraph=self.doc.add\_paragraph(self.Question.toPlainText())

def Create\_Answer(self):

tempstr=self.Answer.text()

self.TotalMark.insert(str(len(tempstr)))

for i in range(len(tempstr)):

if(tempstr[i-1]=="'"):

tempstr=tempstr[:i-1]+"`"+tempstr[i:]

self.Answer.clear()

self.Answer.insert(tempstr)

if(self.UIType=='Choice'):

if(self.AnswerType.currentText()=="Right"):

paragraph = self.doc.add\_paragraph(self.Answer.text()+" ", style='List Bullet')

run=paragraph.add\_run("(+)")

run.font.color.rgb=RGBColor(0xff, 0x00, 0x00)

if(self.ExerciseType.currentText()!='Tick it'):

self.AnswerType.removeItem(1)

elif(self.ExerciseSubType.currentText()=="Правильный ответ один"):

self.AnswerType.removeItem(1)

else:

paragraph = self.doc.add\_paragraph(self.Answer.text(), style='List Bullet')

elif(self.ExerciseType.currentText()=='Drag and drop'):

if(self.ExerciseSubType.currentText()=='Единый текст' or self.ExerciseSubType.currentText()=='Нумерованные предложения'):

run=self.paragraph.add\_run(" ("+self.Answer.text()+")")

run.font.color.rgb=RGBColor(0xff, 0x00, 0x00)

else:

if(self.AnswerType.currentText()=="Wrong"):

run=self.doc.add\_paragraph().add\_run(self.Answer.text())

run.font.bold = True

else:

run=self.paragraph.add\_run(" ("+self.Answer.text()+")")

run.font.color.rgb=RGBColor(0xff, 0x00, 0x00)

elif(self.ExerciseType.currentText()=='Writing'):

run = self.paragraph.add\_run(" ("+self.Answer.text()+")")

run.font.color.rgb=RGBColor(0xff, 0x00, 0x00)

def Finish\_Question(self):

run = self.doc.add\_paragraph().add\_run("%\_Question%")

def Finish\_Exercize(self):

if(self.ExerciseType.currentText()=='Matching'):

self.doc.add\_paragraph()

run = self.doc.add\_paragraph().add\_run("%\_Table%")

elif(self.ExerciseType.currentText()=='Categories'):

self.doc.add\_paragraph()

run = self.doc.add\_paragraph().add\_run("%\_Table%")

run = self.doc.add\_paragraph().add\_run("%\_Questions%")

run = self.doc.add\_paragraph().add\_run("%TotalMark\_%"+self.TotalMark.text()+"%\_TotalMark%")

run.font.color.rgb=RGBColor(0x00, 0x00, 0xff)

run.font.size = Pt(14)

question\_count=0

self.doc.save(self.CourseNumber.text()+"\_"+self.UnitNum+"\_01\_"+self.BlockNumTotal+"\_01"+".docx")

def Finish\_block(self):

self.doc=Document()

def main():

app = QtWidgets.QApplication(sys.argv) # Новый экземпляр QApplication

window = Word() # Создаём объект класса ExampleApp

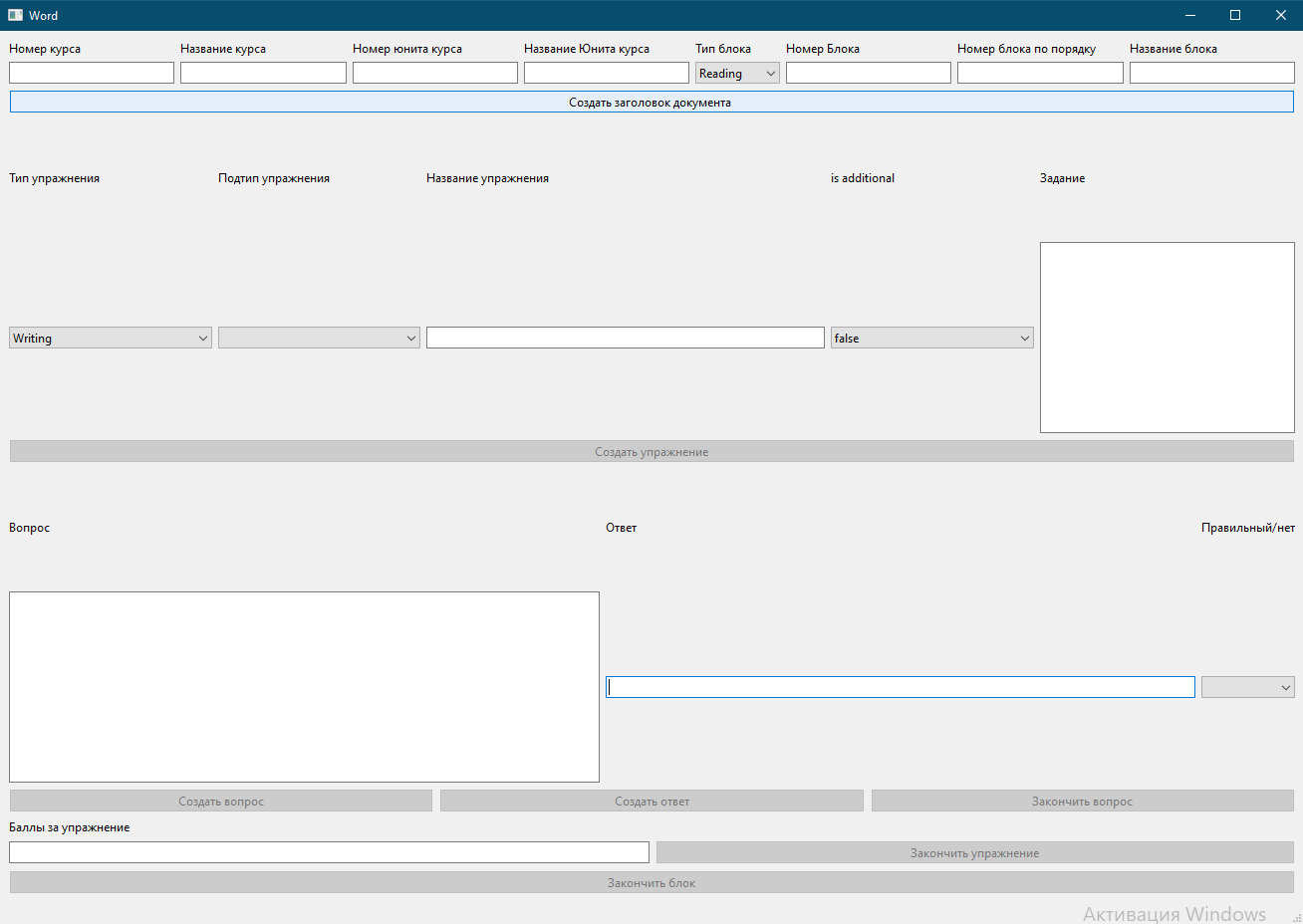
window.show() # Показываем окно

app.exec() # и запускаем приложение

if *\_\_name\_\_* == "\_\_main\_\_":

main()

Qt – форма, конвертируемая в файл design.py



Файл design.py

# Form implementation generated from reading ui file 'form.ui'

#

# Created by: PyQt6 UI code generator 6.1.0

#

# WARNING: Any manual changes made to this file will be lost when pyuic6 is

# run again. Do not edit this file unless you know what you are doing.

from PyQt6 import QtCore, QtGui, QtWidgets

class Ui\_Word(object):

def setupUi(self, Word):

Word.setObjectName("Word")

Word.resize(1309, 897)

self.centralwidget = QtWidgets.QWidget(Word)

self.centralwidget.setObjectName("centralwidget")

self.gridLayout = QtWidgets.QGridLayout(self.centralwidget)

self.gridLayout.setObjectName("gridLayout")

self.gridLayout\_5 = QtWidgets.QGridLayout()

self.gridLayout\_5.setObjectName("gridLayout\_5")

self.Answer = QtWidgets.QLineEdit(self.centralwidget)

self.Answer.setText("")

self.Answer.setObjectName("Answer")

self.gridLayout\_5.addWidget(self.Answer, 1, 1, 1, 1)

self.Question = QtWidgets.QTextEdit(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Expanding, QtWidgets.QSizePolicy.Policy.Preferred)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.Question.sizePolicy().hasHeightForWidth())

self.Question.setSizePolicy(sizePolicy)

self.Question.setPlaceholderText("")

self.Question.setObjectName("Question")

self.gridLayout\_5.addWidget(self.Question, 1, 0, 1, 1)

self.AnswerType = QtWidgets.QComboBox(self.centralwidget)

self.AnswerType.setObjectName("AnswerType")

self.gridLayout\_5.addWidget(self.AnswerType, 1, 2, 1, 1)

self.label\_14 = QtWidgets.QLabel(self.centralwidget)

self.label\_14.setObjectName("label\_14")

self.gridLayout\_5.addWidget(self.label\_14, 0, 0, 1, 1)

self.label\_15 = QtWidgets.QLabel(self.centralwidget)

self.label\_15.setObjectName("label\_15")

self.gridLayout\_5.addWidget(self.label\_15, 0, 1, 1, 1)

self.label\_16 = QtWidgets.QLabel(self.centralwidget)

self.label\_16.setObjectName("label\_16")

self.gridLayout\_5.addWidget(self.label\_16, 0, 2, 1, 1)

self.gridLayout.addLayout(self.gridLayout\_5, 4, 2, 1, 1)

self.Createexec = QtWidgets.QPushButton(self.centralwidget)

self.Createexec.setObjectName("Createexec")

self.gridLayout.addWidget(self.Createexec, 3, 2, 1, 1)

self.Finishblock = QtWidgets.QPushButton(self.centralwidget)

self.Finishblock.setObjectName("Finishblock")

self.gridLayout.addWidget(self.Finishblock, 8, 2, 1, 1)

self.gridLayout\_2 = QtWidgets.QGridLayout()

self.gridLayout\_2.setObjectName("gridLayout\_2")

self.label\_6 = QtWidgets.QLabel(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Minimum, QtWidgets.QSizePolicy.Policy.Fixed)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_6.sizePolicy().hasHeightForWidth())

self.label\_6.setSizePolicy(sizePolicy)

self.label\_6.setObjectName("label\_6")

self.gridLayout\_2.addWidget(self.label\_6, 0, 7, 1, 1)

self.UnitName = QtWidgets.QLineEdit(self.centralwidget)

self.UnitName.setText("")

self.UnitName.setObjectName("UnitName")

self.gridLayout\_2.addWidget(self.UnitName, 1, 4, 1, 1)

self.CourseNumber = QtWidgets.QLineEdit(self.centralwidget)

self.CourseNumber.setText("")

self.CourseNumber.setObjectName("CourseNumber")

self.gridLayout\_2.addWidget(self.CourseNumber, 1, 1, 1, 1)

self.UnitNumber = QtWidgets.QLineEdit(self.centralwidget)

self.UnitNumber.setText("")

self.UnitNumber.setObjectName("UnitNumber")

self.gridLayout\_2.addWidget(self.UnitNumber, 1, 3, 1, 1)

self.label\_7 = QtWidgets.QLabel(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Minimum, QtWidgets.QSizePolicy.Policy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_7.sizePolicy().hasHeightForWidth())

self.label\_7.setSizePolicy(sizePolicy)

self.label\_7.setObjectName("label\_7")

self.gridLayout\_2.addWidget(self.label\_7, 0, 8, 1, 1)

self.BlockNumber = QtWidgets.QLineEdit(self.centralwidget)

self.BlockNumber.setText("")

self.BlockNumber.setObjectName("BlockNumber")

self.gridLayout\_2.addWidget(self.BlockNumber, 1, 7, 1, 1)

self.label\_2 = QtWidgets.QLabel(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Preferred, QtWidgets.QSizePolicy.Policy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_2.sizePolicy().hasHeightForWidth())

self.label\_2.setSizePolicy(sizePolicy)

self.label\_2.setObjectName("label\_2")

self.gridLayout\_2.addWidget(self.label\_2, 0, 2, 1, 1)

self.label\_5 = QtWidgets.QLabel(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Preferred, QtWidgets.QSizePolicy.Policy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_5.sizePolicy().hasHeightForWidth())

self.label\_5.setSizePolicy(sizePolicy)

self.label\_5.setObjectName("label\_5")

self.gridLayout\_2.addWidget(self.label\_5, 0, 6, 1, 1)

self.label = QtWidgets.QLabel(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Preferred, QtWidgets.QSizePolicy.Policy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label.sizePolicy().hasHeightForWidth())

self.label.setSizePolicy(sizePolicy)

self.label.setObjectName("label")

self.gridLayout\_2.addWidget(self.label, 0, 1, 1, 1)

self.CourseName = QtWidgets.QLineEdit(self.centralwidget)

self.CourseName.setText("")

self.CourseName.setObjectName("CourseName")

self.gridLayout\_2.addWidget(self.CourseName, 1, 2, 1, 1)

self.label\_4 = QtWidgets.QLabel(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Preferred, QtWidgets.QSizePolicy.Policy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_4.sizePolicy().hasHeightForWidth())

self.label\_4.setSizePolicy(sizePolicy)

self.label\_4.setObjectName("label\_4")

self.gridLayout\_2.addWidget(self.label\_4, 0, 4, 1, 1)

self.label\_3 = QtWidgets.QLabel(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Preferred, QtWidgets.QSizePolicy.Policy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_3.sizePolicy().hasHeightForWidth())

self.label\_3.setSizePolicy(sizePolicy)

self.label\_3.setObjectName("label\_3")

self.gridLayout\_2.addWidget(self.label\_3, 0, 3, 1, 1)

self.BlockNumberTotal = QtWidgets.QLineEdit(self.centralwidget)

self.BlockNumberTotal.setText("")

self.BlockNumberTotal.setObjectName("BlockNumberTotal")

self.gridLayout\_2.addWidget(self.BlockNumberTotal, 1, 6, 1, 1)

self.BlockName = QtWidgets.QLineEdit(self.centralwidget)

self.BlockName.setText("")

self.BlockName.setObjectName("BlockName")

self.gridLayout\_2.addWidget(self.BlockName, 1, 8, 1, 1)

self.BlockType = QtWidgets.QComboBox(self.centralwidget)

self.BlockType.setObjectName("BlockType")

self.BlockType.addItem("")

self.BlockType.addItem("")

self.BlockType.addItem("")

self.BlockType.addItem("")

self.BlockType.addItem("")

self.BlockType.addItem("")

self.gridLayout\_2.addWidget(self.BlockType, 1, 5, 1, 1)

self.label\_8 = QtWidgets.QLabel(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Minimum, QtWidgets.QSizePolicy.Policy.Minimum)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_8.sizePolicy().hasHeightForWidth())

self.label\_8.setSizePolicy(sizePolicy)

self.label\_8.setObjectName("label\_8")

self.gridLayout\_2.addWidget(self.label\_8, 0, 5, 1, 1)

self.gridLayout.addLayout(self.gridLayout\_2, 0, 2, 1, 1)

self.CreateDoc = QtWidgets.QPushButton(self.centralwidget)

self.CreateDoc.setObjectName("CreateDoc")

self.gridLayout.addWidget(self.CreateDoc, 1, 2, 1, 1)

self.gridLayout\_3 = QtWidgets.QGridLayout()

self.gridLayout\_3.setObjectName("gridLayout\_3")

self.CreateQuestion = QtWidgets.QPushButton(self.centralwidget)

self.CreateQuestion.setObjectName("CreateQuestion")

self.gridLayout\_3.addWidget(self.CreateQuestion, 0, 0, 1, 1)

self.CreateAnswer = QtWidgets.QPushButton(self.centralwidget)

self.CreateAnswer.setObjectName("CreateAnswer")

self.gridLayout\_3.addWidget(self.CreateAnswer, 0, 1, 1, 1)

self.FinishQuestion = QtWidgets.QPushButton(self.centralwidget)

self.FinishQuestion.setObjectName("FinishQuestion")

self.gridLayout\_3.addWidget(self.FinishQuestion, 0, 2, 1, 1)

self.gridLayout.addLayout(self.gridLayout\_3, 5, 2, 1, 1)

self.gridLayout\_6 = QtWidgets.QGridLayout()

self.gridLayout\_6.setObjectName("gridLayout\_6")

self.FinishExercize = QtWidgets.QPushButton(self.centralwidget)

self.FinishExercize.setObjectName("FinishExercize")

self.gridLayout\_6.addWidget(self.FinishExercize, 1, 1, 1, 1)

self.TotalMark = QtWidgets.QLineEdit(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Preferred, QtWidgets.QSizePolicy.Policy.Fixed)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.TotalMark.sizePolicy().hasHeightForWidth())

self.TotalMark.setSizePolicy(sizePolicy)

self.TotalMark.setText("")

self.TotalMark.setObjectName("TotalMark")

self.gridLayout\_6.addWidget(self.TotalMark, 1, 0, 1, 1)

self.label\_17 = QtWidgets.QLabel(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Preferred, QtWidgets.QSizePolicy.Policy.Fixed)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.label\_17.sizePolicy().hasHeightForWidth())

self.label\_17.setSizePolicy(sizePolicy)

self.label\_17.setObjectName("label\_17")

self.gridLayout\_6.addWidget(self.label\_17, 0, 0, 1, 1)

self.gridLayout.addLayout(self.gridLayout\_6, 6, 2, 1, 1)

self.gridLayout\_4 = QtWidgets.QGridLayout()

self.gridLayout\_4.setObjectName("gridLayout\_4")

self.ExerciseType = QtWidgets.QComboBox(self.centralwidget)

self.ExerciseType.setObjectName("ExerciseType")

self.ExerciseType.addItem("")

self.ExerciseType.addItem("")

self.ExerciseType.addItem("")

self.ExerciseType.addItem("")

self.ExerciseType.addItem("")

self.ExerciseType.addItem("")

self.ExerciseType.addItem("")

self.ExerciseType.addItem("")

self.gridLayout\_4.addWidget(self.ExerciseType, 1, 0, 1, 1)

self.Isadditional = QtWidgets.QComboBox(self.centralwidget)

self.Isadditional.setObjectName("Isadditional")

self.Isadditional.addItem("")

self.Isadditional.addItem("")

self.gridLayout\_4.addWidget(self.Isadditional, 1, 3, 1, 1)

self.label\_10 = QtWidgets.QLabel(self.centralwidget)

self.label\_10.setObjectName("label\_10")

self.gridLayout\_4.addWidget(self.label\_10, 0, 1, 1, 1)

self.ExerciseSubType = QtWidgets.QComboBox(self.centralwidget)

self.ExerciseSubType.setObjectName("ExerciseSubType")

self.gridLayout\_4.addWidget(self.ExerciseSubType, 1, 1, 1, 1)

self.ExerciseText = QtWidgets.QTextEdit(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Preferred, QtWidgets.QSizePolicy.Policy.Preferred)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.ExerciseText.sizePolicy().hasHeightForWidth())

self.ExerciseText.setSizePolicy(sizePolicy)

self.ExerciseText.setMaximumSize(QtCore.QSize(400, 400))

self.ExerciseText.setPlaceholderText("")

self.ExerciseText.setObjectName("ExerciseText")

self.gridLayout\_4.addWidget(self.ExerciseText, 1, 4, 1, 1)

self.label\_9 = QtWidgets.QLabel(self.centralwidget)

self.label\_9.setObjectName("label\_9")

self.gridLayout\_4.addWidget(self.label\_9, 0, 0, 1, 1)

self.label\_12 = QtWidgets.QLabel(self.centralwidget)

self.label\_12.setObjectName("label\_12")

self.gridLayout\_4.addWidget(self.label\_12, 0, 3, 1, 1)

self.label\_13 = QtWidgets.QLabel(self.centralwidget)

self.label\_13.setObjectName("label\_13")

self.gridLayout\_4.addWidget(self.label\_13, 0, 4, 1, 1)

self.ExerciseName = QtWidgets.QLineEdit(self.centralwidget)

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.MinimumExpanding, QtWidgets.QSizePolicy.Policy.Fixed)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.ExerciseName.sizePolicy().hasHeightForWidth())

self.ExerciseName.setSizePolicy(sizePolicy)

self.ExerciseName.setMaximumSize(QtCore.QSize(400, 16777215))

self.ExerciseName.setText("")

self.ExerciseName.setObjectName("ExerciseName")

self.gridLayout\_4.addWidget(self.ExerciseName, 1, 2, 1, 1)

self.label\_11 = QtWidgets.QLabel(self.centralwidget)

self.label\_11.setObjectName("label\_11")

self.gridLayout\_4.addWidget(self.label\_11, 0, 2, 1, 1)

self.gridLayout.addLayout(self.gridLayout\_4, 2, 2, 1, 1)

Word.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(Word)

self.menubar.setGeometry(QtCore.QRect(0, 0, 1309, 21))

self.menubar.setObjectName("menubar")

Word.setMenuBar(self.menubar)

self.statusbar = QtWidgets.QStatusBar(Word)

self.statusbar.setObjectName("statusbar")

Word.setStatusBar(self.statusbar)

self.retranslateUi(Word)

QtCore.QMetaObject.connectSlotsByName(Word)

def retranslateUi(self, Word):

\_translate = QtCore.QCoreApplication.translate

Word.setWindowTitle(\_translate("Word", "Word"))

self.label\_14.setText(\_translate("Word", "Вопрос"))

self.label\_15.setText(\_translate("Word", "Ответ"))

self.label\_16.setText(\_translate("Word", "Правильный/нет"))

self.Createexec.setText(\_translate("Word", "Создать упражнение"))

self.Finishblock.setText(\_translate("Word", "Закончить блок"))

self.label\_6.setText(\_translate("Word", "Номер блока по порядку"))

self.label\_7.setText(\_translate("Word", "Название блока"))

self.label\_2.setText(\_translate("Word", "Название курса"))

self.label\_5.setText(\_translate("Word", "Номер Блока"))

self.label.setText(\_translate("Word", "Номер курса"))

self.label\_4.setText(\_translate("Word", "Название Юнита курса"))

self.label\_3.setText(\_translate("Word", "Номер юнита курса"))

self.BlockType.setItemText(0, \_translate("Word", "Reading"))

self.BlockType.setItemText(1, \_translate("Word", "Writing"))

self.BlockType.setItemText(2, \_translate("Word", "Video"))

self.BlockType.setItemText(3, \_translate("Word", "Listening"))

self.BlockType.setItemText(4, \_translate("Word", "Grammar"))

self.BlockType.setItemText(5, \_translate("Word", "Vocabulary"))

self.label\_8.setText(\_translate("Word", "Тип блока"))

self.CreateDoc.setText(\_translate("Word", "Создать заголовок документа "))

self.CreateQuestion.setText(\_translate("Word", "Создать вопрос"))

self.CreateAnswer.setText(\_translate("Word", "Создать ответ"))

self.FinishQuestion.setText(\_translate("Word", "Закончить вопрос"))

self.FinishExercize.setText(\_translate("Word", "Закончить упражнение"))

self.label\_17.setText(\_translate("Word", "Баллы за упражнение"))

self.ExerciseType.setItemText(0, \_translate("Word", "Writing"))

self.ExerciseType.setItemText(1, \_translate("Word", "Tick it"))

self.ExerciseType.setItemText(2, \_translate("Word", "Speaking"))

self.ExerciseType.setItemText(3, \_translate("Word", "Multiple choice"))

self.ExerciseType.setItemText(4, \_translate("Word", "Essay"))

self.ExerciseType.setItemText(5, \_translate("Word", "Matching"))

self.ExerciseType.setItemText(6, \_translate("Word", "Categories"))

self.ExerciseType.setItemText(7, \_translate("Word", "Drag and drop"))

self.Isadditional.setItemText(0, \_translate("Word", "false"))

self.Isadditional.setItemText(1, \_translate("Word", "true"))

self.label\_10.setText(\_translate("Word", "Подтип упражнения"))

self.label\_9.setText(\_translate("Word", "Тип упражнения"))

self.label\_12.setText(\_translate("Word", "is additional"))

self.label\_13.setText(\_translate("Word", "Задание"))

self.label\_11.setText(\_translate("Word", "Название упражнения"))

# Выводы

В ходе выполнения данной работы были получены навыки разработки на языке Python, изучены принципы работы библиотеки pytron-docx и реализация QT-форм на языке python. Разработано приложения для автоматического заполнения docx документов согласно шаблонам.

Код приложения и сопутствующие файлы загружен в репозиторий github: https://github.com/mishakras/Practice