DOI: 10.32517/2221-1993-2024-23-6-64-71

К. С. Поздышева
Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия;
средняя школа № 58 с углубленным изучением предметов естественно-математического цикла, г. Ярославль, Россия

РЕАЛИЗАЦИЯ УЧЕБНЫХ ПРОЕКТОВ НА PYTHON С ПОМОЩЬЮ МОДУЛЯ PY-DOCX*

Аннотация

При изучении программирования целесообразно сочетать систематическую подготовку и проектную деятельность учащихся. На уроках информатики школьники знакомятся с основными алгоритмическими конструкциями и структурами данных. В рамках работы над индивидуальным проектом интересным представляется опыт организации изучения школьниками возможностей различных библиотек. В статье представлены методические рекомендации по организации трех индивидуальных проектов, результатом которых являются: программа, позволяющая генерировать тексты — наградные документы; программа — генератор самостоятельных работ; расписание для классов по готовому расписанию для учителей. Проекты имеют разную сложность и могут быть предложены учащимся IX—X классов.

В статье предложен справочный материал для знакомства школьников с основными возможностями модуля py-docx, сформулированы цели и критерии оценки проектов, дан примерный план работы над проектом, приведены примеры возможных результатов проектной деятельности.

Ключевые слова: обучение программированию, индивидуальный учебный проект, модуль py-docx, Python.

1. Введение

Вопросы совершенствования методики обучения программированию не теряют своей актуальности на протяжении многих лет. В последние годы в качестве первого языка для изучения программирования все чаще используют язык Python [4—6, 8—10]. Целесообразность сочетания систематического изучения основ программирования и проектной деятельности при обучении студентов не раз отмечалась в современной научно-методической литературе [1—3]. Систематический курс программирования дает обучающимся базу для решения прикладных задач. При работе над проектом у учащихся повышается мотивация к изучению программирования, они более активно демонстрируют самостоятельность, инициативность, вовлеченность в конструирование своих знаний. Подобный подход может быть реализован и в средней школе.

Согласно ФГОС среднего общего образования [11], одним из путей развития познавательных, коммуникативных и регулятивных учебных действий, исследовательских и проектных компетенций, предметных и междисциплинарных знаний является включение школьни-

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_06-2024/

Контактная информация

Поздышева Кристина Сергеевна,
магистрант кафедры теории и методики обучения информатике, Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия; *адрес:* 150000, Россия, г. Ярославль, ул. Республиканская, д. 108/1;
учитель информатики, средняя школа № 58 с углубленным изучением предметов естественно-математического цикла, г. Ярославль, Россия; *адрес:* 150060, Россия, г. Ярославль, ул. Труфанова 21а;
e-mail: kpozdysheva@yandex.ru

K. S. Pozdysheva
Yaroslavl State Pedagogical University named after K. D. Ushinsky, Yaroslavl, Russia;
School 58, Yaroslavl, Russia

REALISATION OF LEARNING PROJECTS IN PYTHON USING THE PY-DOCX MODULE

Abstract

When studying programming, it is advisable to combine systematic training and project activities of students. At informatics lessons, students get acquainted with the basic algorithmic structures and data structures. As part of the work on an individual project, the experience of organizing students to study the possibilities of various libraries is interesting. The article presents methodological recommendations for the organization of three individual projects, the result of which is: a program that allows you to generate texts — award documents; program — generator of independent works; timetable for classes according to a ready-made schedule for teachers. Projects have different complexity and can be offered to students in 9^{th}–10^{th} grades.

The article offers reference material for introducing students to the basic capabilities of the py-docx module, defines the aims and criteria for evaluating projects, gives an approximate work plan for the project, and provides examples of possible results of project activities.

Keywords: training in programming, individual learning project, py-docx module, Python.

ков в проектную деятельность. На уроках информатики школьники знакомятся с основными алгоритмическими конструкциями и структурами данных, а при работе над учебным проектом приобретают опыт использования полученных знаний для решения конкретных практических задач. Однако в вопросах выбора тематики проектов учащимся часто бывает нужна помощь преподавателя, и здесь интересным представляется опыт организации изучения возможностей различных библиотек.

Python обладает мощными инструментами для создания, редактирования и анализа текстовых файлов. При выполнении учебного проекта, посвященного генерации и обработке текстовых документов, учащиеся могут расширить свои знания и умения в области программирования, научиться использовать библиотеки Python.

В статье предложены справочный материал для знакомства школьников с основными возможностями модуля py-docx и описание трех учебных проектов разного уровня сложности.

2. Общие сведения о модуле py-docx

Модуль py-docx содержит классы, методы и функции, позволяющие реализовать работу с элементами текстового документа [12].

Класс `docx.Document()` представляет собой загруженный документ.

Открытие документа и его связь с переменной осуществляются с помощью команды следующего формата:

```
переменная =
  docx.Document('название_документа.docx')
```

Команда `переменная = docx.Document()` создает объект «текстовый документ», для сохранения документа в файл необходимо использовать команду:

```
переменная.save('название_документа.docx')
```

В листинге 1 приведен пример программы, генерирующей документ, содержащий текст, списки, таблицу и изображение.

```
import docx

# Создать новый документ
doc = docx.Document()
# Добавить заголовок
title = doc.add_heading('Заголовок 1 уровня', level=1)
# Добавить абзац
p = doc.add_paragraph('Текст с разными начертаниями шрифта: ')
# Добавить в абзац "p" текст и оформить его полужирным шрифтом
run1 = p.add_run('полужирный ')
run1.bold = True
# Добавить в абзац "p" текст и оформить его курсивом
run2 = p.add_run('курсив ')
run2.italic = True
# Добавить в абзац "p" текст и оформить его подчеркнутым шрифтом
run3 = p.add_run('подчеркнутый ')
run3.underline = True
# Добавить в абзац "p" текст
run4 = p.add_run('и обычный.')

#Создать неупорядоченный список
doc.add_paragraph('Первый пункт', style='List Bullet')
doc.add_paragraph('Второй пункт', style='List Bullet')
#Создать упорядоченный список
doc.add_paragraph('Первый пункт', style='List Number')
doc.add_paragraph('Второй пункт', style='List Number')

# Создать таблицу из 2 строк, 3 столбцов
table = doc.add_table(rows=2, cols=3)
table.style= 'Table Grid'

# Заполнить таблицу (table.rows - список строк таблицы table, row.cells -список ячеек
строки row)
for row in table.rows:
  for cell in row.cells:
     cell.text = 'ячейка'

# Добавить изображение
doc.add_picture('1.jpg')

# Сохранить документ
doc.save('example.docx')
```

Листинг 1

Описанные ниже проекты предполагают необходимость обработки информации, расположенной в таблицах исходного текстового документа. Для обращения к списку таблиц, расположенных в документе, необходимо связать его с некоторой переменной с помощью команды следующего формата:

```
переменная = doc.tables
```

В листинге 2 приведен пример программы, анализирующей документ, содержащий текст и таблицы.

```
import docx

doc = docx.Document('example1.docx')

# doc.paragraphs - список абзацев
  документа
print(len(doc.paragraphs))# вывести
  количество абзацев в документе
print(doc.paragraphs[0].text)#вывести
  текст второго абзаца
# вывести значение параметра "выравнивание"
  для каждого абзаца документа
for par in doc.paragraphs:
   print(par.alignment)

# doc.tables - список таблиц документа
print(len(doc.tables))# вывести
  количество таблиц в документе
table1 = doc.tables[0]# table1 - первая
  таблица документа
# вывести текст второй строки первого
  столбца
print(table1.cell(1,0).text)
```

Листинг 2

Представленные программы могут быть использованы в качестве справочной информации, необходимой для выполнения описанных ниже проектов. Также школьникам можно рекомендовать познакомиться с ресурсами, представленными в [7, 12, 13], и со справочной информацией в дополнительных материалах к статье на сайте издательства:

http://infojournal.ru/journals/school/school_06-2024/

3. Проект «Генерация грамот»

3.1. Описание проектной деятельности

Цель проекта заключается в разработке программы, которая заполняет наградные документы по заранее заготовленному шаблону в формате docx. Данные для заполнения должны быть представлены в отдельном docx-файле с таблицей. Программа может использоваться учителями и администрацией школы и должна обеспечить возможность автоматизированной генерации большого количества однотипных документов: грамот, благодарственных писем и т. п.

Исполнителями проекта могут быть учащиеся средних и старших классов, владеющие основами программирования на Python: работа с переменными, условными операторами, циклами, структурами данных (строка, список). Исполнитель должен понимать структуру документов, созданных в текстовом редакторе, способы создания и форматирования текста, вставки изображений и таблиц.

3.2. Этапы работы над проектом

Примерные этапы работы над проектом представлены в таблице 1.

Таблица 1

Этапы работы над проектом «Генератор наградных документов»

Месяц	Этапы работы
Сентябрь	Повторение основ Python, изучение основ работы с файлами
Октябрь	Изучение объектов и методов модуля py-docx: • изучение документации и примеров работы с модулем py-docx; • создание простых программ для работы с документами через py-docx
Ноябрь	Разработка шаблонов грамот: • создание нескольких простых шаблонов грамот в формате docx; • изучение методов вставки данных в шаблоны с помощью py-docx
Декабрь-январь	Написание кода: • написание кода для генерации грамот на основе созданных шаблонов; • реализация механизма автоматического заполнения данных в шаблонах
Февраль-март	Тестирование и отладка кода: • тестирование программы на различных данных; • отладка и исправление ошибок
Апрель	Доработка и оптимизация кода: • оптимизация работы программы и улучшение пользовательского интерфейса (если требуется); • доработка функций на основе обратной связи
Май	Подготовка к защите проекта: • написание технической документации; • подготовка и защита проекта перед школьным жюри

3.3. Пример результата

Результатом работы над проектом может быть код программы, представленной в листинге 3.

В дополнительных материалах к статье на сайте издательства приведены файл с программой, примеры заготовки грамот и документа с информацией об учащихся.

4. Проект «Генератор самостоятельных работ»

4.1. Описание проектной деятельности

Цель проекта: разработать программу для создания самостоятельных работ. Программа предназначена для учителей и должна обеспечить возможность автоматизированной генерации большого количества вариантов самостоятельных работ из подготовленных заранее документов с базой задач.

Исполнителями проекта могут быть учащиеся средних и старших классов, владеющие основами программирования на Python: работа с переменными, условными операторами, циклами, структурами данных (строки, списки). Исполнитель должен уметь работать со случайными числами, разбивать программу на подпрограммы, использовать функции.

4.2. Этапы работы над проектом

Примерные этапы работы над проектом представлены в таблице 2.

4.3. Пример результата

Результатом работы над проектом может быть код программы, представленный в листинге 4.

В дополнительных материалах к статье на сайте издательства представлены файл с программой, примеры исходных файлов с заданиями и ответами.

5. Проект «Расписание»

5.1. Описание проектной деятельности

Целью проекта является разработка программы для генерации документа с расписанием учебных занятий каждого из классов школы на основании информации, содержащейся в документе с таблицей расписания

```
import docx

# Открыть исходный документ с данными об учащихся
doc = docx.Document('children_list.docx')
# Считать таблицу из документа
table = doc.tables[0]
# Открыть заготовку грамоты
template = docx.Document('template.docx')

# Для каждой строки таблицы сохранить информацию из ячеек в отдельных переменных
for row in table.rows[1:]: # Пропускаем заголовок таблицы
  child_name = row.cells[0].text
  sex = row.cells[1].text
  class_number = row.cells[2].text
  teacher_name = row.cells[3].text

  # Создать копию грамоты
  certificate = docx.Document('template.docx')
  #Задать стиль текста
  style= certificate.styles ['Normal']
  style.font.name = 'Calibri'
  style.font.size= 250000

  # Заменить соответствующие элементы грамоты
  for paragraph in certificate.paragraphs:
    if '[child_name]' in paragraph.text:
      paragraph.text = paragraph.text.replace('[child_name]', child_name)
    if '[class_number]' in paragraph.text:
      paragraph.text = paragraph.text.replace('[class_number]', class_number)
    if '[teacher_name]' in paragraph.text:
      paragraph.text = paragraph.text.replace('[teacher_name]', teacher_name)
    if '[sex]' in paragraph.text:
       if sex == 'ж':
            paragraph.text = paragraph.text.replace('[sex]', 'ученица ')
       else:paragraph.text = paragraph.text.replace('[sex]', 'ученик ')
  # Сохранить грамоты с данными об учащемся
  certificate.save(child_name+'_certificate.docx')

print("Грамоты успешно созданы.")
```

Листинг 2

Таблица 2

Этапы работы над проектом «Генератор самостоятельных работ»

Месяц	Этапы работы
Сентябрь	• Изучение объектов и методов модуля py-docx. • Разработка концепции и архитектуры генератора заданий
Октябрь	• Создание прототипа генератора заданий с использованием py-docx. • Тестирование прототипа, консультации с заказчиком
Ноябрь	• Доработка прототипа на основе полученной обратной связи от заказчика. • Реализация сохранения и экспорта заданий в формате docx
Декабрь-январь	• Расширение функциональности генератора, добавление возможности настройки параметров заданий. • Тестирование и отладка кода
Февраль-март	• Внедрение генератора в учебный процесс. • Сбор отзывов и оценка эффективности использования. • Внесение необходимых дополнений в код
Апрель	• Подготовка документации и инструкции по использованию генератора. • Оценка достижения поставленных целей. • Создание презентации проекта
Май	• Представление результатов проекта перед школьным жюри. • Планирование дальнейшего развития программного кода генератора

```
import docx
import random

# Функция для чтения списков заданий и ответов из файлов и выбора случайного задания
def choose_task(filename_task, filename_answ):
  doc_task = docx.Document(filename_task)
  doc_answ = docx.Document(filename_answ)
  tasks = []
  answers = []
  for para in doc_task.paragraphs:
      tasks.append(para.text)
  for para in doc_answ.paragraphs:
      answers.append(para.text)
  task_choice = random.choice(tasks)
  answer_choice = answers[tasks.index(task_choice)]
  return task_choice, answer_choice

# Основная программа
tasks_file = ["task1.docx", "task2.docx", "task3.docx", "task4.docx", "task5.docx"]
answers_file = ["1answ.docx", "2answ.docx", "3answ.docx", "4answ.docx", "5answ.docx"]
for j in range(10):
  self_work = docx.Document()
  for i in range(5):
     task, answer = choose_task(tasks_file[i], answers_file[i])
     self_work.add_paragraph(f"Задание {i + 1}: {task}")
     self_work.add_paragraph(f"Ответ {i + 1}: {answer}")
     self_work.add_paragraph("----------------------")
     self_work.save("cp"+str(j)+".docx")
  print("Самостоятельная работа создана успешно!",j)
```

Листинг 4

учителей школы. Программа предназначена для администрации школы.

Исполнителями проекта могут быть учащиеся старших классов, владеющие основами программирования на Python, уверенно оперирующие такими структурами данных, как строки и списки, имеющими представление о работе со словарями.

5.2. Этапы работы над проектом

Примерные этапы работы над проектом представлены в таблице 3.

5.3. Пример результата

Возможны разные способы решения поставленной задачи в зависимости от выбранных структур данных.

Таблица 3

Этапы работы над проектом «Расписание»

Месяц	Этапы работы
Сентябрь	• Изучение объектов и методов модуля py-docx. • Реализация сохранения и анализа документов в формате docx
Октябрь	• Изучение примеров использования структуры данных «словарь» для решения задач. • Выбор и описание структур данных для представления информации о расписании
Ноябрь	• Разработка программы для создания расписания одного класса
Декабрь-январь	• Уточнение структур данных. • Создание программы для создания расписания всех классов школы
Февраль-март	• Расширение функциональности программы, добавление возможности выбора параметров исходного и итогового документов. • Тестирование и отладка кода
Апрель	• Подготовка документации и инструкции по использованию программы. • Оценка достижения поставленных целей. • Создание презентации проекта
Май	• Представление результатов проекта перед школьным жюри. • Планирование дальнейшего развития программного кода

Одним из вариантов может быть использование словаря следующего формата:

ключ — наименование класса (например, строка ‘7а’);

элементы — словари, в которых ключами служат названия дней недели, а элементами — списки строк с названиями уроков.

На рисунке приведен пример фрагмента такого словаря для школы, где занятия проходят парами и максимальное количество пар в расписании — четыре.

В листинге 5 приведена программа, в которой задача решается в два этапа: 1) считывание исходной информации в словарь и 2) заполнение документов расписаниями отдельных классов по информации из словаря.

Другой способ организации данных (листинг 6) предполагает, что вся информация хранится в словаре текстовых документов: ключом является номер класса, а элементом — текстовый документ. Таким образом, документы с расписаниями отдельных классов создаются в процессе анализа исходной информации.

6. Критерии оценки проекта

Критерии оценки результатов работы над проектом могут быть следующими:

1. Качество кода: структурированность и читаемость кода, использование подходящих алгоритмов и структур данных, наличие комментариев, описывающих функциональность и логику кода.
2. Функциональность программы. Конкретизация данного критерия зависит от содержания проекта:
   - *для проекта «Генерация грамот»:*
     – возможность автоматического заполнения наградных документов;
     – корректность генерации грамот;
     – возможность сохранения и экспорта созданных грамот в формате *.docx;
   - *для проекта «Генератор самостоятельных работ»:*
     – возможность выборки заданий из документов в соответствии с указаниями заказчика;
     – настройка количества заданий и их формы (несколько абзацев, наличие изображений и т. п.);
   - *для проекта «Расписание»:*
     – возможность управления выбором параметров расписания (один класс или несколько);
     – настройка элементов документа с расписанием (заголовки, форматирование, состав итогового документа).

```
{'5а': {'вторник': ['Русский язык', 'Математика', 'Биология',''],
'понедельник': ['Физическая культура','Математика','Русский язык',''],
'пятница': ['Музыка', 'Математика', 'Русский язык', 'Внеурочная деятельность'],
'среда': ['', 'Математика', 'География', 'Физическая культура'],
'четверг': ['Русский язык', 'Английский язык', 'Технология',''],},
'5б': {'вторник': ['', 'Русский язык', 'Внеурочная деятельность', 'Технология'],
'понедельник': [ 'Русский язык', 'Физическая культура', 'Математика', '',],
'пятница': ['Математика', 'Биология', 'Английский язык', 'Биология'],
…}}
```

Рис. Пример фрагмента словаря

```
import docx

doc = docx.Document('расписание1.docx')
table = doc.tables[0]
# Создание словаря для хранения расписания классов
class_lesson = dict()
# Заполнение словаря в соответствии с таблицей "Расписание учителей"
for ind_row in range(2, len(table.rows)):
    for ind_col in range(2, len(table.columns)):
        cell_text = table.cell(ind_row, ind_col).text
        if not cell_text.isspace():

            if cell_text not in class_lesson:
                class_lesson[cell_text] = {table.cell(0, ind_col).text: [''] * 7}

            name_day = table.cell(0, ind_col).text
            if name_day not in class_lesson[cell_text]:
                class_lesson[cell_text][name_day] = [''] * 7
            number_lesson = int(table.cell(1, ind_col).text)

            class_lesson[cell_text][name_day][number_lesson] +=
              table.cell(ind_row, 0).text + '\n'

#Создание документов с расписаниями классов в соответствии с элементами словаря
days = ['понедельник', 'вторник', 'среда', 'четверг', 'пятница', 'суббота']
for cl in class_lesson:
    doc = docx.Document()
    table = doc.add_table(7, 6)
    table.style = 'Table Grid'
    a = class_lesson[cl]
    for key in a:
        col_ind = days.index(key)
        table.cell(0, col_ind).text = days[col_ind]
        for j in range(1, len(a[key])):
            table.cell(j, col_ind).text = a[key][j]

    doc.save('расписание'+cl+'.docx')
```

Листинг 5

```
table = doc.tables[0]
# Словарь документов с расписаниями классов
class_lesson = dict()
days = ['понедельник', 'вторник', 'среда', 'четверг', 'пятница', 'суббота']
# Чтение данных из документа с расписанием учителей, создание словаря документов #для
каждого класса
for ind_row in range(2, len(table.rows)):
    for ind_col in range(2, len(table.columns)):
        cell_text: str = table.cell(ind_row, ind_col).text
        if not cell_text.isspace():
            if cell_text not in class_lesson:
                class_lesson[cell_text] = docx.Document()
                class_lesson[cell_text].add_table(8, 6)
                row0 = class_lesson[cell_text].tables[0].rows[0]
                for i in range(len(days)):
                    row0.cells[i].text = days[i]
            number_lesson = int(table.cell(1, ind_col).text)
            name_lesson = table.cell(ind_row, 0).text
            day = table.cell(0, ind_col).text
            timetable = class_lesson[cell_text].tables[0]
            timetable.cell(number_lesson, days.index(day)).text = name_lesson

print(class_lesson)
for namedoc in class_lesson:
    class_lesson[namedoc].tables[0].style = 'Table Grid'
    class_lesson[namedoc].save(namedoc + '.docx')
```

Листинг 6

3. Надежность кода:
   - обработка исключений и ошибок;
   - предотвращение сбоев при работе с документами.
4. Дружественный пользовательский интерфейс программы.
5. Качество оформления документации.
6. Творческий подход:
   - оригинальность и креативность в решении поставленной задачи;
   - наличие дополнительных функций или возможностей, выходящих за рамки основных требований;
   - использование нестандартных подходов или технологий.

7. Заключение

Дидактической целью использования в учебном процессе описанных в данной статье проектов является создание условий, при которых учащиеся могут самостоятельно приобретать новые знания и практические навыки программирования, решая реальные задачи из повседневной школьной практики.

Разумное сочетание систематического изучения основ программирования и проектной деятельности школьников способствует повышению качества усвоения учебного материала и мотивации к изучению программирования. Выполнение учебных проектов, результатом которых выступает некоторый программный код, позволяет школьникам приобрести опыт постановки цели и выделения задач деятельности, поиска информации, необходимой для решения поставленных задач; выбора наиболее эффективных способов решения задач в зависимости от конкретных условий.

Список источников

1. *Баженова И. В.* Методика проективно-рекурсивного обучения программированию студентов математических направлений подготовки: дис. ... канд. пед. наук. Красноярск, 2015. 156 с. EDN: VJENGV.

2. *Барышева И. В., Малкина Е. В., Козлов О. А.* Проектный метод обучения программированию студентов профильных специальностей в условиях дистанционной работы // Вопросы методики преподавания в вузе. 2021. Т. 10. № 38. С. 40–55. EDN: VMIGWI. DOI: 10.18720/HUM/ISSN2227-8591.38.04.

3. *Бордюгова Т. Н.* Методические подходы к формированию компетенций в области программирования на основе реализации индивидуальной траектории обучения: на примере подготовки бакалавров по направлению «Педагогическое образование», профиль «Информатика»: дис. ... канд. пед. наук. М., 2011. 141 с. EDN: QFRLYV.

4. *Босова Л. Л.* Как учат программированию в XXI веке: отечественный и зарубежный опыт обучения программированию в школе // Информатика в школе. 2018. № 6. С. 3–11. EDN: XZOOJV.

5. *Босова Л. Л.* Международные тенденции развития школьного образования в области информатики и ИКТ // Материалы Шестнадцатой открытой Всероссийской конференции «Преподавание информационных технологий в Российской Федерации» (Москва, 14–15 мая 2018 года). М.: МГТУ имени Н. Э. Баумана, 2018. С. 350–352. EDN: XUJFSX.

6. *Босова Л. Л., Аквилянов Н. А., Кочергин И. О., Штепа Ю. Л., Бурцева Т. А.* Информатика. 8–9 классы. Начала программирования на языке Python. Дополнительные главы к учебникам. М.: БИНОМ. Лаборатория знаний, 2022. 96 с.

7. Документация python-docx 1.1.2. https://python-docx.readthedocs.io/en/latest/index.html

8. *Каракозов С. Д., Маняхина В. Г.* Python как базовый язык обучения программированию // Информатика в школе. 2020. № 1. С. 26–30. EDN: GPEZPH. DOI: 10.32517/2221-1993-2020-19-1-26-30.

9. *Поляков К. Ю., Еремин Е. А.* Информатика. 11 класс. Базовый и углубленный уровни. В 2 ч. Ч. 2. М.: БИНОМ. Лаборатория знаний, 2017. 304 с.

10. *Поляков К. Ю., Еремин Е. А.* Новая линейка учебников информатики для основной и средней школы // Актуальные проблемы физико-математического образования. Материалы II Международной научно-практической конференции (Набережные Челны, 20–22 октября 2017 года). Набережные Челны: Набережночелнинский государственный педагогический университет, 2017. С. 233–234. EDN: YUXTFB.

11. Приказ Министерства образования и науки Российской Федерации от 17 мая 2012 г. № 413 «Об утверждении федерального государственного образовательного стандарта среднего общего образования». https://fgos.ru/fgos/fgos-soo/

12. Работа с документами, модуль python-docx // Платформа электронного обучения GeeksforGeeks. https://www.geeksforgeeks.org/working-with-documents-python-docx-module/?ref=ml_lbp

13. Работа с текстом, модуль python-docx // Справочник по языку Python3. https://docs-python.ru/packages/modul-python-docx-python/rabota-tekstom/