

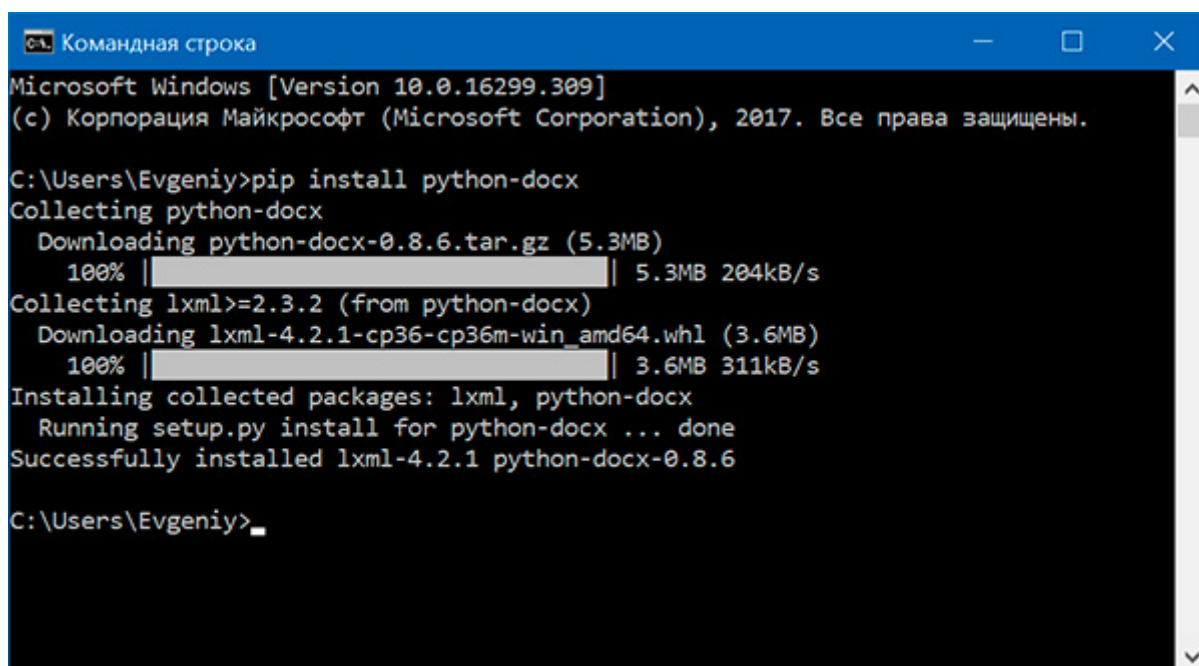
# Работа с файлами MS Word в Python

06.04.2018

Теги: [MS Word](#) • [Python](#) • [Web-разработка](#) • [Модуль](#)

С помощью модуля `python-docx` можно создавать и изменять документы MS Word с расширением `.docx`. Чтобы установить этот модуль, выполняем команду

```
> pip install python-docx
```



```
Командная строка
Microsoft Windows [Version 10.0.16299.309]
(c) Корпорация Майкрософт (Microsoft Corporation), 2017. Все права защищены.

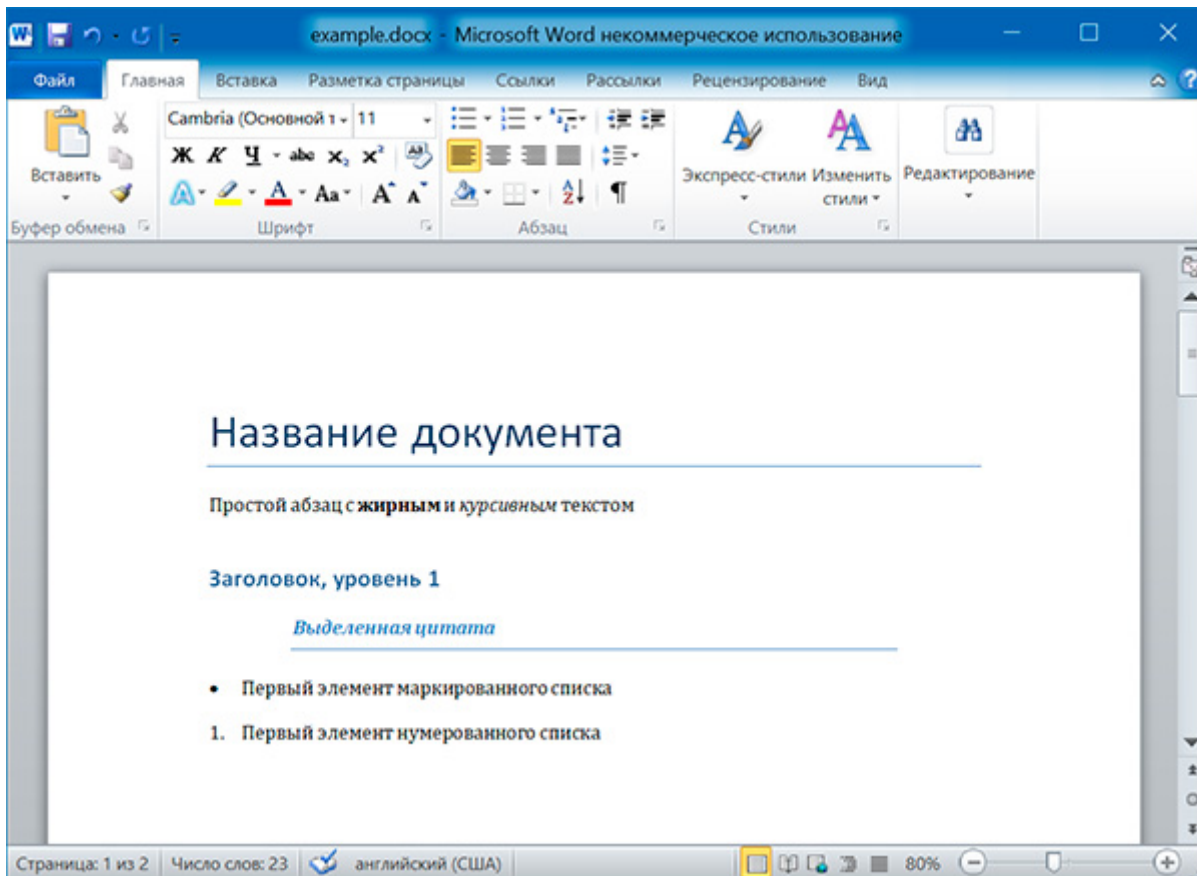
C:\Users\Evgeniy>pip install python-docx
Collecting python-docx
  Downloading python-docx-0.8.6.tar.gz (5.3MB)
    100% |#####| 5.3MB 204kB/s
Collecting lxml>=2.3.2 (from python-docx)
  Downloading lxml-4.2.1-cp36-cp36m-win_amd64.whl (3.6MB)
    100% |#####| 3.6MB 311kB/s
Installing collected packages: lxml, python-docx
  Running setup.py install for python-docx ... done
Successfully installed lxml-4.2.1 python-docx-0.8.6

C:\Users\Evgeniy>
```

## Чтение документов MS Word

При установке модуля надо вводить `python-docx`, а не `docx` (это другой модуль). В то же время при импортировании модуля `python-docx` следует использовать `import docx`, а не `import` файлы с расширением `.docx` обладают развитой внутренней структурой. В модуле `python-docx` эта структура представлена тремя различными типами данных. На самом верхнем уровне объект `Document` представляет собой весь документ. Объект `Document` содержит список объектов `Paragraph`, которые представляют собой абзацы документа. Каждый из абзацев содержит список, состоящий из одного или нескольких объектов `Run`, представляющих собой фрагменты текста с различными стилями форматирования.





```
import docx

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

# количество абзацев в документе
print(len(doc.paragraphs))

# текст первого абзаца в документе
print(doc.paragraphs[0].text)

# текст второго абзаца в документе
print(doc.paragraphs[1].text)

# текст первого Run второго абзаца
print(doc.paragraphs[1].runs[0].text) Копировать
```

```
6
Название документа
Простой абзац с жирным и курсивным текстом
Простой абзац с Копировать
```

Получаем весь текст из документа:

```
text = []
for paragraph in doc.paragraphs:
```

```
text.append(paragraph.text)
print('\n'.join(text)) Копировать
```

Название документа

Простой абзац с жирным и курсивным текстом

Заголовок, уровень 1

Выделенная цитата

Первый элемент маркированного списка

Первый элемент нумерованного списка Копировать

## Стилевое оформление

В документах MS Word применяются два типа стилей: **стили абзацев**, которые могут применяться к объектам Paragraph, **стили символов**, которые могут применяться к объектам Run. Как объектам Paragraph, так и объектам Run можно назначать стили, присваивая их атрибутам `style` значение в виде строки. Этой строкой должно быть имя стиля. Если для стиля задано значение `None`, то у объекта Paragraph или Run не будет связанного с ним стиля.

### Стили абзацев

- Normal
- Body Text
- Body Text 2
- Body Text 3
- Caption
- Heading 1
- Heading 2
- Heading 3
- Heading 4
- Heading 5
- Heading 6
- Heading 7
- Heading 8
- Heading 9
- Intense Quote
- List
- List 2
- List 3
- List Bullet
- List Bullet 2
- List Bullet 3
- List Continue
- List Continue 2

- List Continue 3
- List Number
- List Number 2
- List Number 3
- List Paragraph
- Macro Text
- No Spacing
- Quote
- Subtitle
- TOCHHeading
- Title

## Стили символов

- Emphasis
- Strong
- Book Title
- Default Paragraph Font
- Intense Emphasis
- Subtle Emphasis
- Intense Reference
- Subtle Reference

```
paragraph.style = 'Quote'
run.style = 'Book Title'Копировать
```

## Атрибуты объекта Run

Отдельные фрагменты текста, представленные объектами `Run`, могут подвергаться дополнительному форматированию с помощью атрибутов. Для каждого из этих атрибутов может быть задано одно из трех значений: `True` (атрибут активизирован), `False` (атрибут отключен) и `None` (применяется стиль, установленный для данного объекта `Run`).

- `bold` — Полужирное начертание
- `underline` — Подчеркнутый текст
- `italic` — Курсивное начертание
- `strike` — Зачеркнутый текст

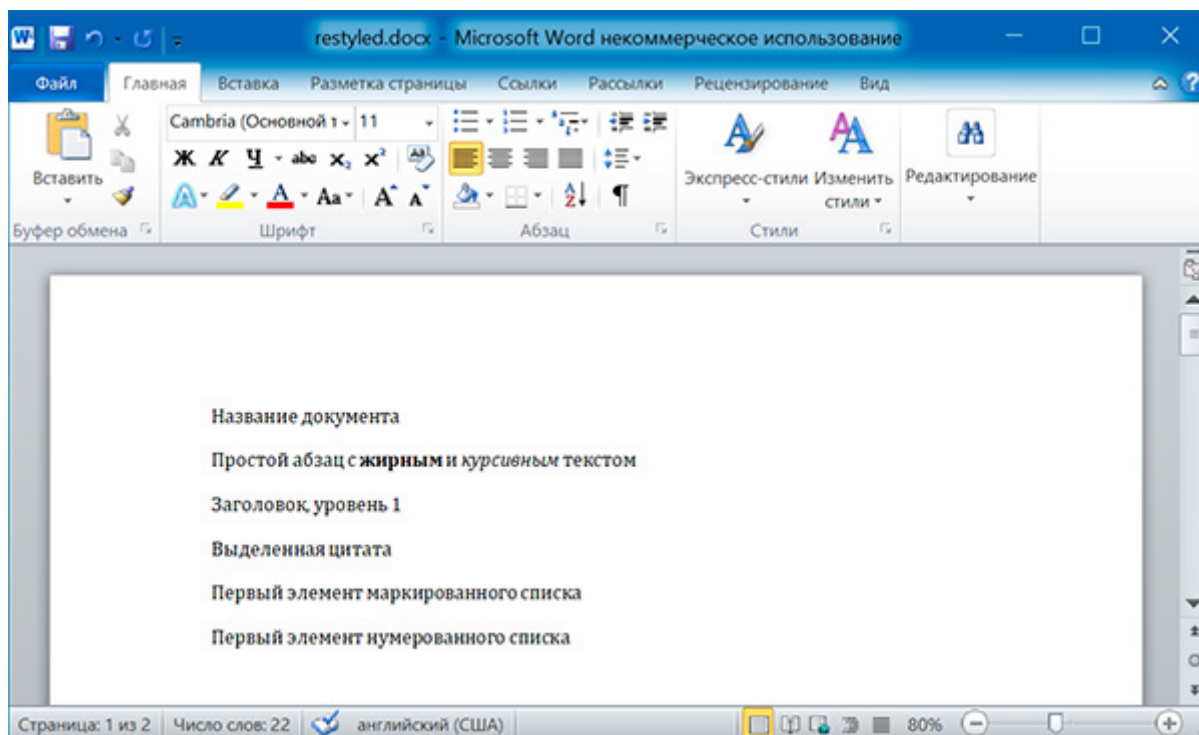
Изменим стили для всех параграфов нашего документа:

```
import docx

doc = docx.Document('example.docx')
```

```
# изменяем стили для всех параграфов
for paragraph in doc.paragraphs:
    paragraph.style = 'Normal'

doc.save('restyled.docx') Копировать
```



А теперь восстановим все как было:

```
import docx

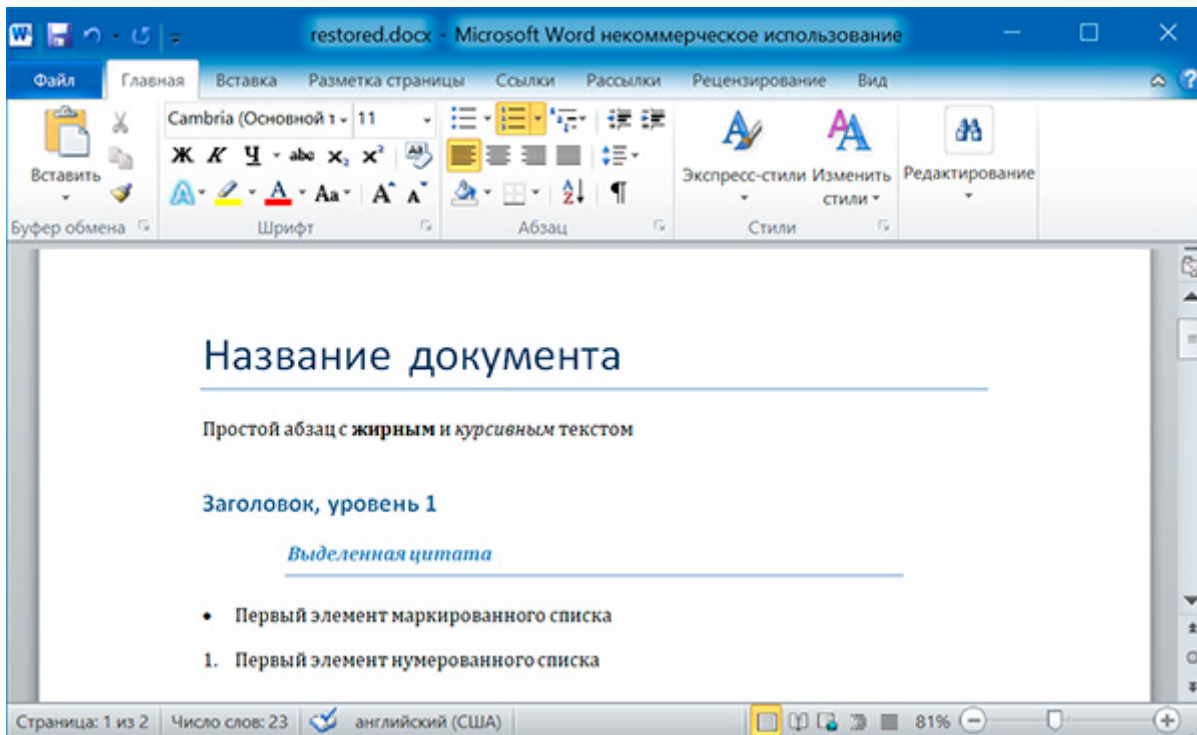
os.chdir('C:\\example')

doc1 = docx.Document('example.docx')
doc2 = docx.Document('restyled.docx')

# получаем из первого документа стили всех абзацев
styles = []
for paragraph in doc1.paragraphs:
    styles.append(paragraph.style)

# применяем стили ко всем абзацам второго документа
for i in range(len(doc2.paragraphs)):
    doc2.paragraphs[i].style = styles[i]

doc2.save('restored.docx') Копировать
```



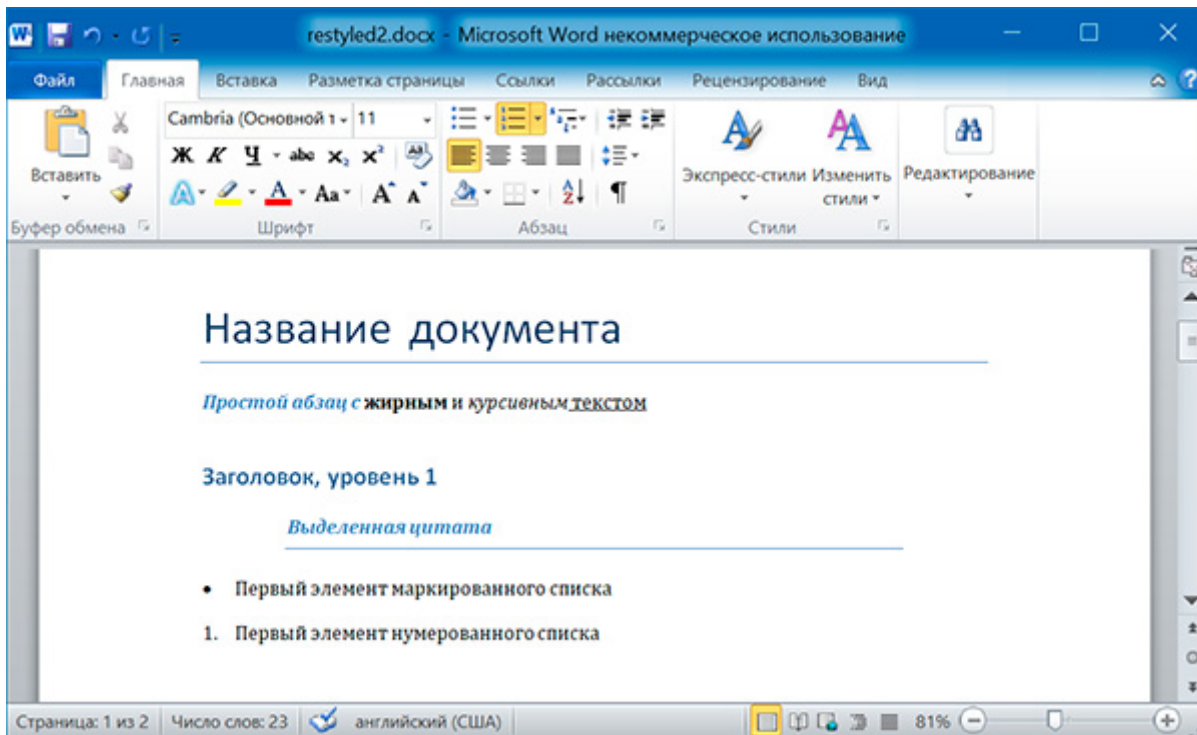
Изменим форматирование объектов Run второго абзаца:

```
import docx

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

# добавляем стиль символов для runs[0]
doc.paragraphs[1].runs[0].style = 'Intense Emphasis'
# добавляем подчеркивание для runs[4]
doc.paragraphs[1].runs[4].underline = True

doc.save('restyled2.docx') Копировать
```



## Запись документов MS Word

Добавление абзацев осуществляется вызовом метода `add_paragraph()` объекта `Document`. Для добавления текста в конец существующего абзаца, надо вызвать метод `add_run()` объекта `Paragraph`:

```
import docx

doc = docx.Document()

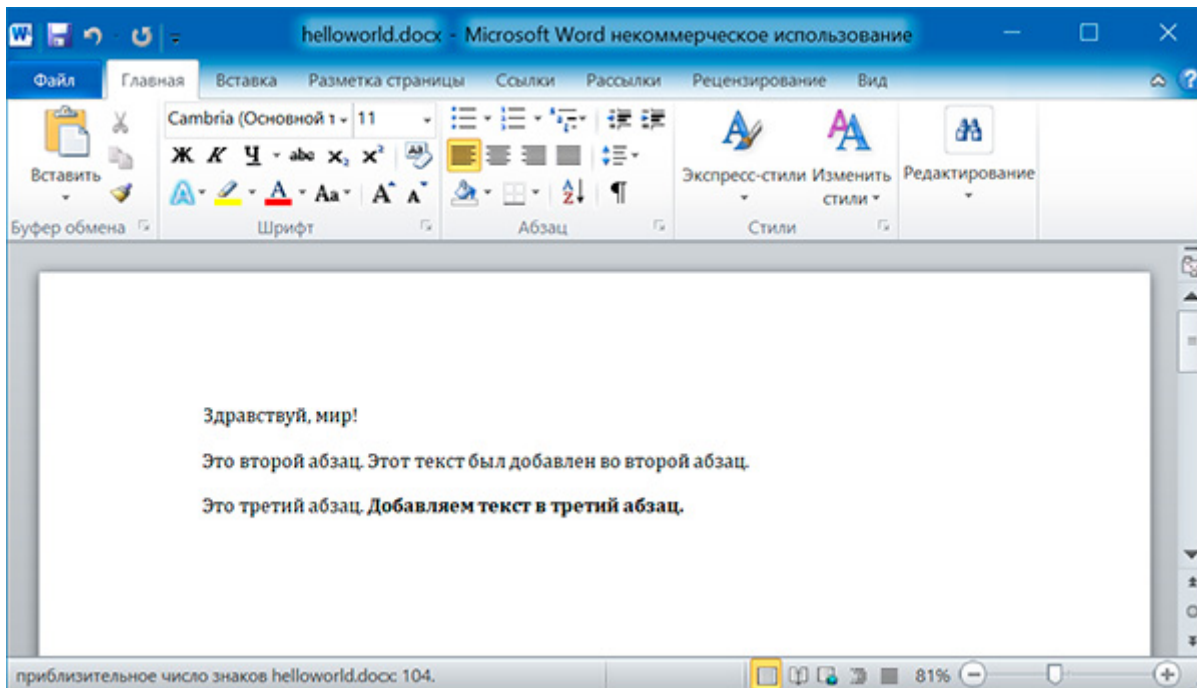
# добавляем первый параграф
doc.add_paragraph('Здравствуй, мир!')

# добавляем еще два параграфа
par1 = doc.add_paragraph('Это второй абзац.')
par2 = doc.add_paragraph('Это третий абзац.')

# добавляем текст во второй параграф
par1.add_run(' Этот текст был добавлен во второй абзац.')

# добавляем текст в третий параграф
par2.add_run(' Добавляем текст в третий абзац.').bold = True

doc.save('helloworld.docx') Копировать
```



Оба метода, `add_paragraph()` и `add_run()` принимают необязательный второй аргумент, содержащий строку стиля, например:

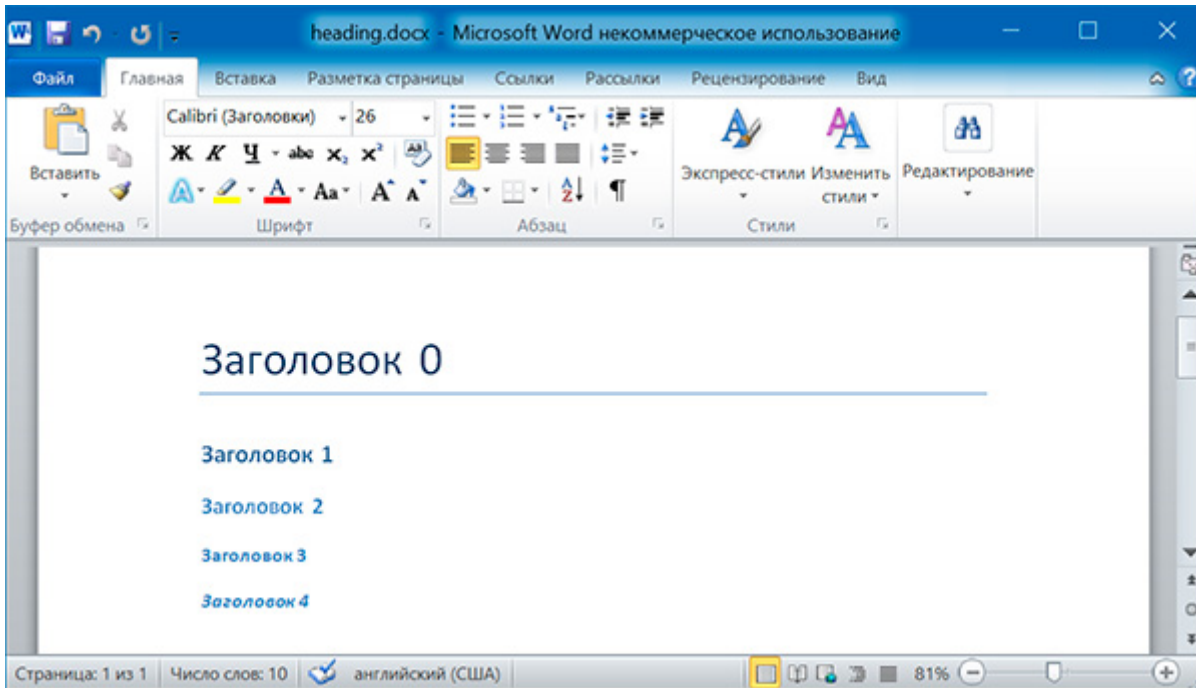
```
doc.add_paragraph('Здравствуй, мир!', 'Title') Копировать
```

## Добавление заголовков

Вызов метода `add_heading()` приводит к добавлению абзаца, отформатированного в соответствии с одним из возможных стилей заголовков:

```
doc.add_heading('Заголовок 0', 0)
doc.add_heading('Заголовок 1', 1)
doc.add_heading('Заголовок 2', 2)
doc.add_heading('Заголовок 3', 3)
doc.add_heading('Заголовок 4', 4) Копировать
```





Аргументами метода `add_heading()` являются строка текста и целое число от 0 до 4. Значению 0 соответствует стиль заголовка `Title`.

## Добавление разрывов строк и страниц

Чтобы добавить разрыв строки (а не добавлять новый абзац), нужно вызвать метод `add_break()` объекта `Run`. Если же требуется добавить разрыв страницы, то методу `add_break()` надо передать значение `docx.enum.text.WD_BREAK.PAGE` в качестве единственного аргумента:

```
import docx

doc = docx.Document()

doc.add_paragraph('Это первая страница')
doc.paragraphs[0].runs[0].add_break(docx.enum.text.WD_BREAK.PAGE)
doc.add_paragraph('Это вторая страница')

doc.save('pages.docx') Копировать
```

## Добавление изображений

Метод `add_picture()` объекта `Document` позволяет добавлять изображения в конце документа. Например, добавим в конец документа изображение `kitten.jpg` шириной 10 сантиметров:

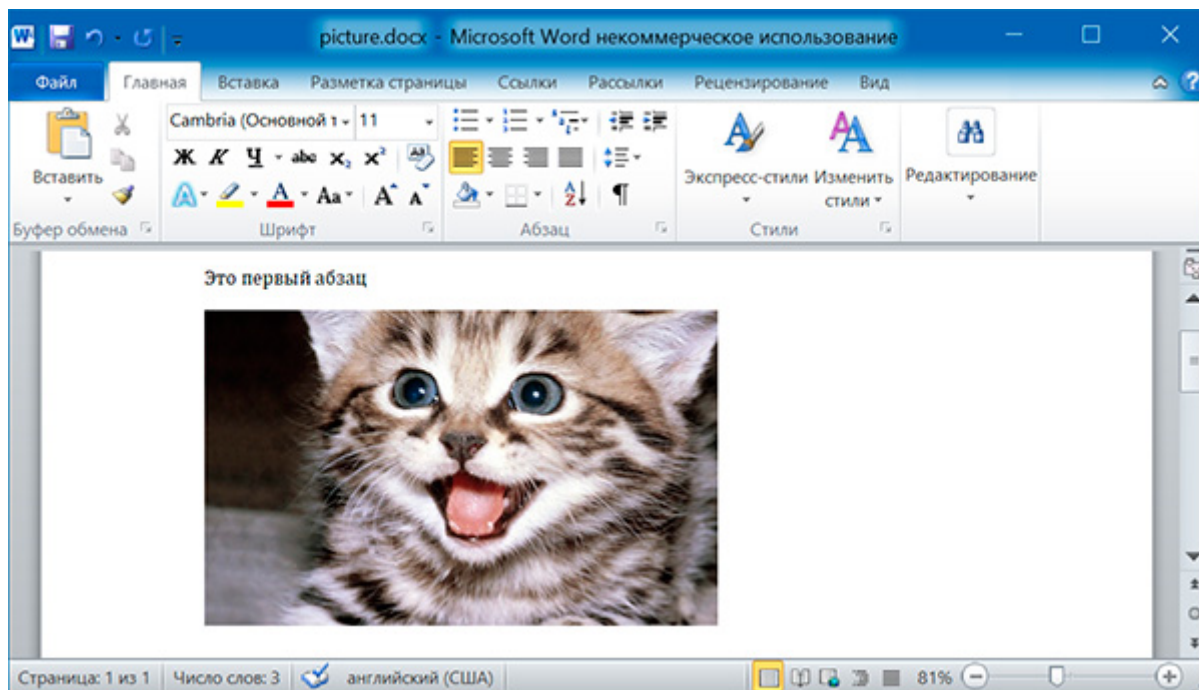
```
import docx

doc = docx.Document()

doc.add_paragraph('Это первый абзац')
doc.add_picture('kitten.jpg', width = docx.shared.Cm(10))
```

```
doc.save('picture.docx') Копировать
```

Именованные аргументы `width` и `height` задают ширину и высоту изображения. Если их опустить, то значения этих аргументов будут определяться размерами самого изображения.



## Добавление таблицы

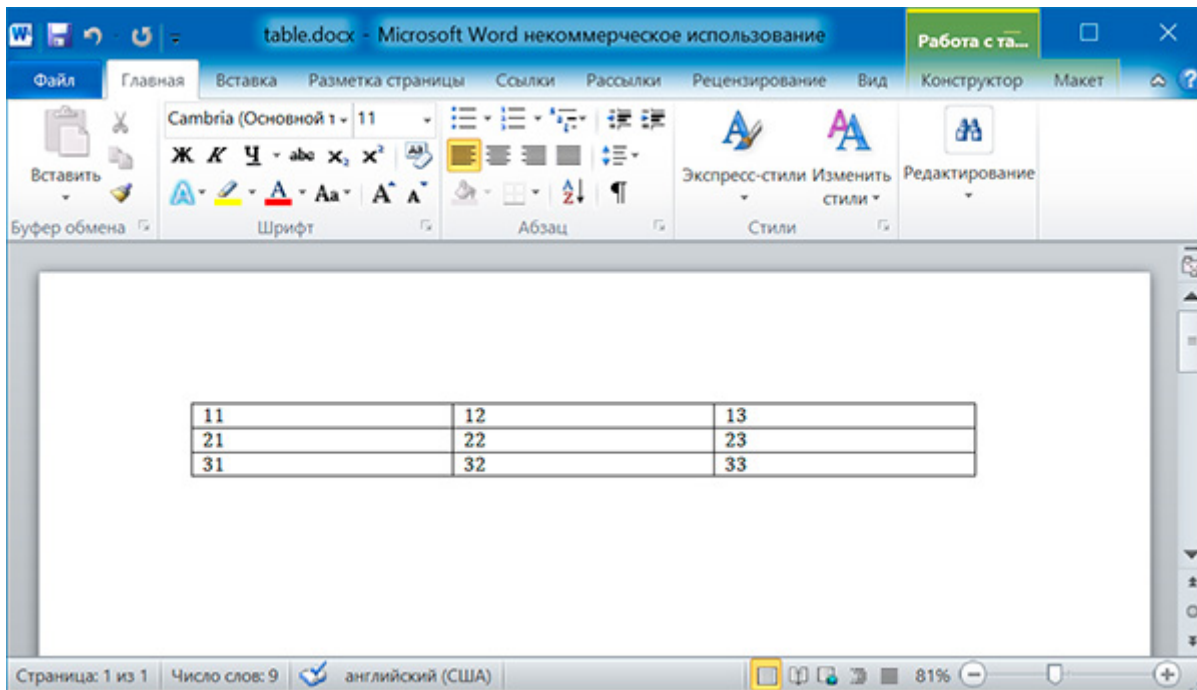
```
import docx

doc = docx.Document()

# добавляем таблицу 3x3
table = doc.add_table(rows = 3, cols = 3)
# применяем стиль для таблицы
table.style = 'Table Grid'

# заполняем таблицу данными
for row in range(3):
    for col in range(3):
        # получаем ячейку таблицы
        cell = table.cell(row, col)
        # записываем в ячейку данные
        cell.text = str(row + 1) + str(col + 1)

doc.save('table.docx') Копировать
```



```
import docx
```

```
doc = docx.Document('table.docx')
```

```
# получаем первую таблицу в документе
```

```
table = doc.tables[0]
```

```
# читаем данные из таблицы
```

```
for row in table.rows:
```

```
    string = ''
```

```
    for cell in row.cells:
```

```
        string = string + cell.text + ' '
```

```
    print(string) Копировать
```

```
11 12 13
```

```
21 22 23
```

```
31 32 33 Копировать
```

- [Документация python-docx](#)

Поиск: [MS](#) • [Python](#) • [Web-разработка](#) • [Word](#) • [Модуль](#)