МойОфис

Комплект Средств Разработки (SDK)

Руководство программиста

MYOFFICE DOCUMENT API (Python)

2022.01

ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«МОЙОФИС КОМПЛЕКТ СРЕДСТВ РАЗРАБОТКИ (SDK)»

MYOFFICE DOCUMENT APPLICATION PROGRAMMING INTERFACE (API). БИБЛИОТЕКА ДЛЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON

РУКОВОДСТВО ПРОГРАММИСТА

2022.01

На 151 листах

Москва

2023

Все упомянутые в этом документе названия продуктов, логотипы, торговые марки и товарные знаки принадлежат их владельцам.

Товарные знаки «МойОфис» и «MyOffice» принадлежат ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ».

Ни при каких обстоятельствах нельзя истолковывать любое содержимое настоящего документа как прямое или косвенное предоставление лицензии или права на использование товарных знаков, логотипов или знаков обслуживания, приведенных в нем.

Любое несанкционированное использование этих товарных знаков, логотипов или знаков обслуживания без письменного разрешения их правообладателя строго запрещено.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

В настоящем документе используются следующие сокращения (см. Таблица 1):

Таблица 1 - Сокращения и расшифровки

Сокращение	Расшифровка
ОС	Операционная система.
MyOffice Document API	Программное обеспечение «МойОфис Комплект Средств Разработки (SDK). MyOffice Document API. Библиотека для языка программирования Python».
API	Application Programming Interface (программный интерфейс приложения).
SDK	Software Development Kit (комплект для разработки программного обеспечения).

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Назначение программы

Библиотека MyOffice Document API для языка программирования Python используется в составе прикладных информационных систем или отдельных приложений под управлением ОС Microsoft Windows или Linux. Библиотека предназначена для решения задач по созданию и наполнению текстовых и табличных документов в пакетном режиме.

1.2 Библиотека MyOffice Document API для языка программирования Python

Библиотека MyOffice Document API для языка программирования Python предоставляет возможность выполнения следующих операций:

1. Создание, открытие, сохранение изменений в электронных текстовых и табличных документах в следующих форматах:
   - текстовые и табличные документы, создаваемые с помощью Microsoft Office в формате OOXML, расширения файлов DOCX и XLSX;
   - текстовые и табличные документы, создаваемые с помощью LibreOffice в формате ODF, расширения файлов ODT и ODS;
   - текстовые и табличные документы, создаваемые с помощью МойОфис в формате ODF, расширения файлов XODT и XODS;
   - экспорт документов в формате PDF/A-1.
2. Изменение содержимого документов в пакетном режиме, в том числе:
   - добавление, удаление, изменение текста абзаца;
   - вставка, удаление, форматирование таблиц в текстовом документе;
   - вставка, удаление, переименование отдельных листов в табличном документе;
   - установка значения ячейки электронной таблицы и расчет формул;
   - оформление документа с использованием различных шрифтов и цветового оформления.
3. Поиск и замена фрагмента текста в документе.
4. Управление режимом рецензирования документа, отслеживание изменений в документе.
5. Управление закладками в текстовом документе.
6. Написание и запуск макрокоманд.

Для управления содержимым документа используется объектная модель, представляющая собой совокупность структур данных текстового или табличного документа.

1.3 Уровень подготовки пользователя

Пользователь MyOffice Document API должен иметь:

1. Опыт разработки на языке Python для ОС Microsoft Windows или Linux. Полный список поддерживаемых ОС приведен в документе «МойОфис Комплект Средств Разработки (SDK). MyOffice Document Application Programming Interface (API). Системные требования».
2. Навык работы со стандартными офисными приложениями.

1.4 Системные требования

Полный перечень требований к программному и аппаратному обеспечению приведен в документе «МойОфис Комплект Средств Разработки (SDK). MyOffice Document Application Programming Interface (API). Системные требования».

2 ПОДГОТОВКА К РАБОТЕ

2.1 Список дистрибутивов

Дистрибутив MyOffice Document API поставляется в виде архивных файлов (см. Таблица 2).

Таблица 2 - Список дистрибутивов MyOffice Document API

ОС	Дистрибутив
Microsoft Windows	MyOffice_SDK_Document_API_Python_Win_2022.01_x64.zip
Linux	MyOffice_SDK_Document_API_Python_Linux_2022.01_x64.zip

2.2 Установка в ОС Microsoft Windows

Для установки MyOffice Document API в ОС Microsoft Windows необходимо разместить файл дистрибутива в локальной папке и осуществить следующие действия:

1. Открыть окно командной строки ОС Microsoft Windows.
2. Перейти в локальную папку с файлом дистрибутива.
3. Развернуть архивный файл **MyOffice_SDK_Document_API_Python_Win_2022.01_x64.zip.**
4. Установить программный пакет MyOffice Document API с помощью системы управления пакетами, например: **pip install MyOfficeSDKDocumentAPI-2.0-cp38-cp38m-win_amd64.whl.**

Внимание ! Убедитесь в актуальности установленной версии Python. Для использования MyOffice Document API 2022.01 необходима версия Python 3.8.6.

2.3 Установка в ОС Linux

Для установки MyOffice Document API в ОС Linux необходимо разместить файл дистрибутива в локальной папке и осуществить следующие действия:

1. Открыть окно терминала ОС Linux.
2. Перейти в локальную папку с файлом дистрибутива.
3. Развернуть архивный файл **MyOffice_SDK_Document_API_Python_Linux_2022.01_x64.zip.**

4. Установить программный пакет MyOffice Document API с помощью системы управления пакетами, например:
   pip install MyOfficeSDKDocumentAPI-2.0-cp38-cp38mu-linux_x86_64.whl.

Внимание ! Убедитесь в актуальности установленной версии Python. Для использования MyOffice Document API 2022.01 необходима версия Python 3.8.6.

2.4 Проверка работоспособности

Для проверки работоспособности MyOffice Document API необходимо выполнить тестовый пример.

Тестовый пример использует вызовы MyOffice Document API для создания текстового документа в формате DOCX.

```
from MyOfficeSDKDocumentAPI import DocumentAPI as mof

application = mof.Application()
document = application.createDocument(mof.DocumentType_Text)
document.getRange().getBegin().insertText("Hello! This is an example!")
document.saveAs("BasicExample.docx")
```

Сохраните код в файле **basic-app.py** и выполните команду:

```
python basic-app.py
```

В результате работы программы в текущем каталоге создается файл **BasicExample.docx**, содержащий текст «Hello! This is an example!».

MyOffice Document API считается работоспособным, если приложение выполнено успешно.

2.5 Распространение разработанных приложений

Распространение разработанного приложения осуществляется посредством передачи файла, содержащего исходный код приложения.

Для запуска разработанного приложения на компьютере пользователя должны присутствовать:

- интерпретатор Python, версии 3.8.6;
- установленный пакет MyOffice Document API для языка программирования Python.

3 ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

Приведенные примеры предназначены для ознакомления с основными функциями MyOffice Document API и содержат примеры использования классов, методов, структур и функций при разработке приложений.

3.1 Работа с текстовым документом

3.1.1 Создание и сохранение документа

Для создания документа необходимо создать экземпляр объекта типа Application и вызвать метод createDocument, который создаст новый документ указанного в параметре типа с настройками по умолчанию. Возможен также вызов метода createDocument с использованием в качестве параметра объекта, в котором указан тип создаваемого документа DocumentType. Во всех дальнейших примерах данный фрагмент не будет дублироваться, предполагается, что документ уже создан или открыт.

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk
application = sdk.Application()

# Создание текстового документа
document = application.createDocument(sdk.DocumentType_Text)
```

Сохранение документа осуществляется с помощью метода saveAs, содержащегося в объектной модели документа Document.

```
# Наполнение документа
document.getRange().getBegin().insertText("Hello! This is an example!")

# Сохранение документа в формате DOCX
document.saveAs("BasicExample.docx")
```

3.1.2 Работа с текстом

3.1.2.1 Настройка свойств текста

Для настройки свойств текста необходимо создать объект TextProperties и инициализировать его поля.

```
# Вставка содержимого документа
document.getRange().getBegin().insertText("Hello! This is an example!")

# Доступ к тексту документа
docText = document.getRange()

# Установить свойства текста
textProperties = sdk.TextProperties()
textProperties.bold = True
textProperties.textColor = sdk.ColorRGBA(255,0,0,0)
docText.setTextProperties(textProperties)
```

```
document.saveAs("BasicExample.docx")
```

3.1.3 Работа с таблицами

3.1.3.1 Вставка таблицы

Для вставки таблицы необходимо задать ее положение в документе. В примере таблица вставляется в начало документа. В случае успешного выполнения вставки возвращается объект вставленной таблицы.

```
# Вставка таблицы 3x3
position = document.getRange().getBegin()

# Вставка таблицы
rowsCount     = 3
columnsCount  = 3
tableName     = "MyTable"
table = position.insertTable(rowsCount, columnsCount, tableName)

# Сохранить документ в формате DOCX
document.saveAs("BasicExample.docx")
```

3.1.3.2 Удаление таблицы

Для удаления таблицы необходимо получить объект таблицы. Необходимо обратить внимание на то, что объект таблицы перестает существовать после удаления из документа.

```
# Вставка таблицы 3x3
position = document.getRange().getBegin()

# Вставка таблицы
rowsCount     = 3
columnsCount  = 3
tableName     = "MyTable"
table = position.insertTable(rowsCount, columnsCount, tableName)

# Удалить таблицу
table.remove()

# Сохранение документа в формате DOCX
document.saveAs("BasicExample.docx")
```

Объект таблицы также может быть получен из объекта Document по индексу таблицы. Необходимо обратить внимание на то, что объект таблицы перестает существовать после удаления из документа.

```
table = document.getBlocks().getTable(0)
table.remove()
```

3.1.4 Работа с ячейками таблицы

3.1.4.1 Объединение ячеек таблицы

Для объединения ячеек необходим объект CellRange. Объект CellRange имеет методы объединения и разъединения ячеек.

```
cellRange = table.getCellRange(DocumentAPI.CellRangePosition(1, 1, 3, 3))
cellRange.merge()
```

3.1.4.2 Разъединение ячеек таблицы

Для того чтобы разъединить ячейки, необходимо получить объект ячейки Cell из диапазона объединения ячеек. Объект Cell содержит метод для разъединения всех ячеек из диапазона объединения, частью которого является эта ячейка.

```
cell = table.getCell(CellPosition(2, 2))
cell.unmerge()
```

3.1.4.3 Поворот текста в ячейке

Для того чтобы задать поворот текста в ячейке, необходимо получить объект ячейки CellProperties. Объект CellProperties содержит поле TextOrientation, в котором содержится ориентация текста ячейки.

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

cellProperties = sdk.CellProperties()
cellProps.textOrientation = sdk.TextOrientation(90)
table.getCell("D10").setCellProperties(cellProps)
```

3.1.5 Форматирование таблицы

3.1.5.1 Изменение ширины столбца таблицы

Объект Table содержит метод для изменения ширины указанного столбца. Следующий пример демонстрирует настройку ширины второго столбца до 300 единиц.

```
table.setColumnWidth(1, 300)
```

3.1.5.2 Установка свойств форматирования ячеек таблицы

Установка свойств форматирования ячеек таблицы и значений ячейки.

```
# Вставка таблицы 3x3
position = document.getRange().getBegin()

rowsCount     = 3
columnsCount  = 3
tableName     = "MyTable"
tableEmployes = position.insertTable(rowsCount, columnsCount, tableName)

# Установка свойств форматирования для ячейки
```

```
cellProperties = sdk.CellProperties()
cellProperties.verticalAlignment = sdk.VerticalAlignment_Center
cellProperties.backgroundColor   = sdk.ColorRGBA(211,211,211,1) #Light-Gray

cellOrder = tableEmployes.getCell(sdk.CellPosition(0,0))
cellOrder.setText("№ п/п")
cellOrder.setCellProperties(cellProperties)

cellName = tableEmployes.getCell(sdk.CellPosition(0,1))
cellName.setText("ФИО")
cellOrder.setCellProperties(cellProperties)

cellAge = tableEmployes.getCell(sdk.CellPosition(0,2))
cellAge.setText("Возраст")
cellOrder.setCellProperties(cellProperties)

# Сохранение документа в формате DOCX
document.saveAs("BasicExample.docx")
```

3.1.5.3 Установка границ ячеек таблицы

Установка внешних и внутренних границ ячеек таблицы.

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()

# Создание текстового документа
document = application.createDocument(sdk.DocumentType_Text)

# Вставка таблицы 3x3
position = document.getRange().getBegin()

# Вставка таблицы
rowsCount     = 3
columnsCount  = 3
tableName     = "MyTable"
tableEmployees = position.insertTable(rowsCount, columnsCount, tableName)

# Установка внешних границ
cellRangeOuter = tableEmployees.getCellRange(sdk.CellRangePosition(0,0,2,2))

lineOuter = sdk.LineProperties()
lineOuter.style = sdk.LineStyle_Solid
lineOuter.width = 2.0
lineOuter.color = sdk.ColorRGBA(255,0,255,1)

bordersOuter = sdk.RangeBorders()
cellRangeOuter.setBorders(bordersOuter.setOuter(lineOuter))

# Установка внутренних границ
lineInner = sdk.LineProperties()
lineInner.style = sdk.LineStyle_LongDash
lineInner.width = 1.0
lineInner.color = sdk.ColorRGBA(128,0,128,1)

bordersInner = sdk.RangeBorders()
```

```
cellRangeOuter.setBorders(bordersInner.setInnerHorizontal(lineInner))
cellRangeOuter.setBorders(bordersInner.setInnerVertical(lineInner))

# Сохранение документа в формате DOCX
document.saveAs("BasicExample.docx")
```

3.1.6 Работа с закладками

3.1.6.1 Вставка закладки

Для вставки в документ закладки необходимо задать ее расположение. В примере приведено позиционирование закладки в начале документа. Наименование закладки должно быть уникальным.

```
# Вставка закладки в текстовый документ
bmkName = "Executor"
position = document.getRange().getBegin()
position.insertBookmark(bmkName)

# Сохранение документа в формате DOCX
document.saveAs("BasicExample.docx")
```

3.1.6.2 Изменение содержимого закладки

В документе можно изменить содержимое закладки. Для этого необходимо задать ее наименование. Наименование закладки должно быть уникальным.

```
# Изменение содержимого закладки
collectionBookmarks = document.getBookmarks()
rangeExecutor = collectionBookmarks.getBookmarkRange(bmkName)
txtExecutor = "Исполнитель: Иванов И.И., доб.1234"
rangeExecutor.replaceText(txtExecutor)

# Сохранение документа в формате DOCX
document.saveAs("BasicExample.docx")
```

3.1.6.3 Удаление закладки

Для удаления закладки из документа используется метод списка закладок `Bookmarks` `removeBookmark`, где `name` - наименование удаляемой закладки.

```
# Вставка закладки в текстовый документ
bmkName = "Executor"
position = document.getRange().getBegin()
position.insertBookmark(bmkName)

# Удаление закладки
document.getBookmarks().removeBookmark(bmkName)

document.saveAs("BasicExample.docx")
```

3.1.7 Работа с комментариями

3.1.7.1 Получение списка комментариев

```
# Получение коллекции комментариев
auto enumerator = document.getComments().getEnumerator()
for comment in enumerator:

    # Поля комментария
    print(comment.getText())
    print(comment.getInfo().author).name)
    print(comment.getRange().extractText())
```

3.1.7.2 Получение списка ответов

```
# Получение коллекции комментариев
enumerator = document.getRange().getComments().getEnumerator()
for comment in enumerator:

    # Список ответов на комменатрий
    repliesEnumerator = comment.getReplies().getEnumerator()
    for reply in repliesEnumerator:

        # Поля ответа на комментария
        print(reply.getText())
        print(reply.getInfo().author.name)
```

3.1.8 Экспорт текстового документа

Средствами Document API поддерживается операция экспорта документа только в формат PDF/A-1b.

```
# Наполнение документа
document.getRange().getBegin().insertText("Hello! This is an example!")

# Экспорт в формат PDF
outputFile = "./BasicExample.pdf"
document.exportAs(outputFile, sdk.ExportFormat_PDFA1)
```

3.1.9 Поиск в текстовом документе

Для поиска в текстовом документе необходимо с помощью глобальной функции `createSearch` создать экземпляр объекта `Search` и вызвать метод `findText`.

```
# Вставка содержимого документа
text = "API documentation describes what services an API offers and how to " \
       "use those services, aiming to cover everything a client would need " \
       "to know for practical purposes. Documentation is crucial for the " \
       "development and maintenance of applications using the API. API " \
       "documentation is traditionally found in documentation files but can " \
       "also be found in social media such as blogs, forums, and Q&A websites."
document.getRange().getBegin().insertText(text)
```

```
# Настройка свойств текста
textProperties = sdk.TextProperties()
textProperties.bold = True

# Поиск текстового фрагмента в документе
textSearch = sdk.createSearch(document)
ranges = textSearch.findText("API")

# Установить свойства текста для найденных фрагментов
for range in ranges:
    range.setTextProperties(textProperties)
document.saveAs("BasicExample.docx")
```

3.1.10 Управление ориентацией и свойствами страниц раздела

Для управления ориентацией и свойствами страниц раздела требуется выбрать раздел.

Выбор раздела возможен посредством использования метода `getSectionsEnumerator`.

Выбор первого раздела документа.

```
sections = list(document.getSectionsEnumerator())
section = sections[0]
```

Выбор раздела возможен из блока.

Выбор раздела в первом блоке документа:

```
section = document.getBlocks().getBlock(0).getSection()
```

Выбор раздела также возможен из абзаца (только в текстовом документе).

Выбор раздела для первого абзаца:

```
section = document.getBlocks().getParagraph(0).getSection()
```

После выбора раздела становится возможной работа с ориентацией и свойствами страниц раздела.

Установка свойств страниц раздела:

```
section.setPageProperties(properties)
```

Получение свойств страниц раздела:

```
page_properties = section.getPageProperties()
```

Установка альбомной ориентации для раздела:

```
section.setPageOrientation(PageOrientation_Landscape)
```

Получение ориентации страниц раздела:

```
page_orientation = section.getPageOrientation()
```

Установка книжной ориентации для всего документа:

```
document.setPageOrientation(PageOrientation_Portrait)
```

Создание свойства страницы с высотой, равной 841.9, и шириной, равной 595.3:

```
properties = DocumentAPI.PageProperties()
properties.height = 841.9
properties.width = 595.3
```

3.1.11 Работа с отслеживаемыми изменениями

```
# Получение коллекции отслеживаемых изменений из всего диапазона документа
enumerator = document.getRange().getTrackedChangesEnumerator()
for trackedChange in enumerator:
    # Поля структуры TrackedChange:
    print(trackedChange.getType())
    print(trackedChange.getInfo().author).name)
    print(trackedChange.getRange().extractText())
```

3.1.12 Работа с колонтитулами раздела

Получение верхнего и нижнего колонтитулов первого раздела документа.

```
# Получаем коллекцию разделов
sectionsEnumerator = document.getSectionsEnumerator()

for section in sectionsEnumerator:
    headers = section.getHeaders()
    footers = section.getFooters()
    for header in headers:
        # Верхний колонтитул первого раздела
        print(header.getRange().extractText())
    for footer in footers:
        # Нижний колонтитул первого раздела
        print(footer.getRange().extractText())
```

Получение типов колонтитулов разделов документа.

```
# Получаем коллекцию разделов
sectionsEnumerator = document.getSectionsEnumerator()

for section in sectionsEnumerator:
    headers = section.getHeaders()
    for header in headers:
        # Получаем тип колонтитула
        print(header.getType())
```

3.2 Работа с табличным документом

3.2.1 Создание и сохранение документа

Для создания документа необходимо создать экземпляр объекта типа `Application` и вызвать метод `createDocument`, который создаст новый документ указанного в параметре

типа с настройками по умолчанию. Возможен также вызов метода createDocument с использованием в качестве параметра заданных свойств создаваемого документа (см. DocumentSettings).

Во всех дальнейших примерах данный фрагмент не будет дублироваться, предполагается, что документ уже создан или открыт.

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
#Создание табличного документа
settings = sdk.DocumentSettings()
settings.documentType = sdk.DocumentType_Workbook
document = application.createDocument(settings)
```

Сохранение документа осуществляется с помощью метода saveAs, содержащегося в объектной модели документа (см. Document.saveAs).

```
# Вставка таблицы
position = document.getRange().getBegin()
rowsCount = 3
columnsCount = 3
sheetName = "List1"
table = position.insertTable(rowsCount, columnsCount, sheetName)

#Вставка текста в ячейку
text = "Create Settings Example"
table.getCell("B2").setText(text)

#Сохранить документ в формате XLSX
document.saveAs("BasicExample.xlsx")
```

3.2.2 Работа с текстом

3.2.2.1 Настройка свойств текста

Для настройки свойств текста необходимо создать объект TextProperties и инициализировать его поля.

```
text_properties = DocumentAPI.TextProperties()
text_properties.bold = True
text_properties.textColor = DocumentAPI.ColorRGBA(255, 0, 0, 0)
document
.getBlocks().getParagraph(0).getRange().setTextProperties(text_properties)
```

Существует более короткий путь настройки свойств текста с помощью специального конструктора.

```
document.getBlocks().getParagraph(0).getRange()
    .setTextProperties(
    DocumentAPI.TextProperties(bold = True, textColor = DocumentAPI.ColorRGBA(255, 0, 0, 0)
    ))
```

3.2.3 Работа с листами табличного документа

3.2.3.1 Вставка рабочего листа в табличный документ

Не существует разницы между вставками таблицы в текстовом документе и рабочими листами в табличном документе. Для вставки листа также необходимо иметь позицию вставки. В примере используется позиция в конце документа. Уникальный идентификатор таблицы должен быть возвращен в случае успешного завершения операции вставки.

```
position = document.getRange().getEnd();
position.insertTable(10, 10, "Sheet");
```

3.2.3.2 Удаление рабочего листа табличного документа

Для удаления таблицы необходимо получить объект таблицы. Следует обратить внимание на то, что объект таблицы перестает существовать после удаления из документа.

```
table = document.getBlocks().getTable(table_name)
table.remove()
```

Объект таблицы также может быть получен из объекта Document по индексу таблицы. Следует обратить внимание на то, что объект таблицы перестает существовать после удаления из документа.

```
table = document.getBlocks().getTable(0)
table.remove()
```

3.2.4 Работа с ячейками таблицы

3.2.4.1 Настройка свойств ячейки

Для настройки свойств ячейки в общем случае необходимо создать объект CellProperties и инициализировать необходимые поля.

```
cell_properties = DocumentAPI.CellProperties()
cell_properties.verticalAlignment = DocumentAPI.VerticalAlignment_Center
cell_properties.backgroundColor = DocumentAPI.ColorRGBA(255, 0, 0, 0)
cell.setCellProperties(cell_properties)
```

Но существует более короткий путь – с использованием специального конструктора.

```
cell.setCellProperties(DocumentAPI.CellProperties(
                        verticalAlignment = DocumentAPI.VerticalAlignment_Center,
                        backgroundColor = DocumentAPI.ColorRGBA(255, 0, 0, 0)
                       ))
```

Пример установки выравнивания текста абзаца по ширине ячейки:

```
# Вставка таблицы
position = document.getRange().getBegin()
rowsCount = 3
```

```
columnsCount = 10
sheetName = "List1"
table = position.insertTable(rowsCount, columnsCount, sheetName)

#Вставка текста в ячейку
text = "Text Alignment Example"
table.getCell("B2").setText(text)

#Выравнивание текста по ширине ячейки
paraProps = sdk.ParagraphProperties()
paraProps.alignment = sdk.Alignment_Justify
table.getCell("B2").setParagraphProperties(paraProps)

#Сохранить документ в формате XLSX
document.saveAs("BasicExample.xlsx")
```

3.2.4.2 Объединение ячеек таблицы

Для объединения ячеек необходим объект `CellRange`. Объект `CellRange` имеет методы объединения и разъединения ячеек.

```
cellRange = table.getCellRange(DocumentAPI.CellRangePosition(1, 1, 3, 3))
cellRange.merge()
```

3.2.4.3 Разъединение ячеек таблицы

Для того чтобы разъединить ячейки, необходимо получить объект ячейки `Cell` из диапазона объединения ячеек. Объект `Cell` содержит метод для разъединения всех ячеек из диапазона объединения, частью которого является эта ячейка.

```
cell = table.getCell(CellPosition(2, 2))
cell.unmerge()
```

3.2.4.4 Поворот текста в ячейке

Для поворота текста в ячейке необходимо создать объект `CellProperties` и задать значение полю `textOrientation`.

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

try:
    application = sdk.Application()
    #Создание табличного документа
    document = application.createDocument(sdk.DocumentType_Workbook)
    # Вставка таблицы
    position = document.getRange().getBegin()
    rowsCount = 3
    columnsCount = 10
    sheetName = "List1"
    table = position.insertTable(rowsCount, columnsCount, sheetName)

    #Вставка текста в ячейку
    text = "Text Orientation Example"
    table.getCell("B2").setText(text)
```

```
    #Поворот текста на 90 градусов
    cellProps = sdk.CellProperties()
    cellProps.textOrientation = sdk.TextOrientation(90)
    table.getCell("B2").setCellProperties(cellProps)

    #Сохранить документ в формате XLSX
    document.saveAs("BasicExample.xlsx")
    print('OK')

except:
    print ("Error")
```

3.2.5 Форматирование таблицы

3.2.5.1 Изменение ширины столбца таблицы

Объект Table содержит метод для изменения ширины указанного столбца. Следующий пример демонстрирует настройку ширины второго столбца таблицы до 300 единиц.

```
table.setColumnWidth(1, 300)
```

3.2.6 Экспорт табличного документа

Средствами Document API поддерживается операция экспорта документа только в формат PDF/A-1b.

```
table = document.getRange().getBegin().insertTable(40,40, "Text Search")

# Наполнение документа
txtA1 = "Hello! This is an example!"
table.getCell("A1").setText(txtA1)

#Сохранение табличного документа в формате XLSX
document.saveAs("wbk.xlsx")

# Экспорт в формат PDF
outputFile = "BasicExample.pdf"
document.exportAs(outputFile, sdk.ExportFormat_PDFA1)
```

3.2.7 Поиск в табличном документе

Для поиска в табличном документе необходимо с помощью глобальной функции `createSearch` создать экземпляр объекта Search и вызвать метод `findText`.

```
table = document.getRange().getBegin().insertTable(40,40, "Text Search")

# Вставка содержимого документа
txtA1 = "API documentation describes what services an API offers and how to " \
        "use those services, aiming to cover everything a client would need " \
        "to know for practical purposes."
txtA2 = "Documentation is crucial for the development and maintenance of " \
        "applications using the API. "
txtA3 = "API documentation is traditionally found in documentation files but " \
```

```
        "can also be found in social media such as blogs, forums, and Q&A " \
        "websites."

table.getCell("A1").setText(txtA1)
table.getCell("A2").setText(txtA2)
table.getCell("A3").setText(txtA3)

# Настройка свойств текста
textProperties = sdk.TextProperties()
textProperties.bold = True

# Поиск текстового фрагмента в документе
textSearch = sdk.createSearch(document)
collect = textSearch.findText("API")

# Установка свойств текста для найденных фрагментов
for r in collect:
    r.setTextProperties(textProperties)

document.saveAs("BasicExample.xlsx")
```

3.2.8 Работа со сводными таблицами

3.2.8.1 Получение диапазона исходных данных сводной таблицы

```
# Загруженный документ содержит сводную таблицу

# Получаем ячейку, находящуюся в диапазоне исходных данных сводной таблицы
pivotRootCell = document.getBlocks().getTable(1).getCell(CellPosition(2, 0))

# Получаем сводную таблицу
pivotTable = pivotRootCell.getPivotTable()

# Получаем диапазон исходных данных сводной таблицы
sourceCellRange = pivotTable.getSourceRange()

# Для получения границ диапазона используем поля CellRange:
print(sourceCellRange.getBeginRow())
print(sourceCellRange.getBeginColumn())
print(sourceCellRange.getLastRow())
print(sourceCellRange.getLastColumn())
```

3.2.8.2 Получение диапазона размещения сводной таблицы

```
# Получаем диапазон размещения сводной таблицы
pivotCellRange = pivotTable.getPivotRange()
```

3.2.8.3 Получение неподдерживаемых свойств сводной таблицы

```
# Получаем неподдерживаемые свойства сводной таблицы
pivotUnsuportedFeatures = pivotTable.getUnsupportdeFeatures()
```

3.2.8.4 Получение флагов отображения общих итогов для строк и колонок

```
// Получаем флаги отображения общих итогов для строк и колонок
isRowGrandTotalEnabled = pivotTable.isRowGrandTotalEnabled()
```

```
isColGrandTotalEnabled = pivotTable.isColumnGrandTotalEnabled()
```

3.2.8.5 Получение заголовков сводной таблицы

```
# Получение заголовков сводной таблицы
captions = pivotTable.getPivotTableCaptions()

# Используем поля структуры PivotTableCaptions:
print(captions.emptyCaption.get())
print(captions.errorCaption.get())
print(captions.rowHeaderCaption)
print(captions.columnHeaderCaption)
print(captions.valuesHeaderCaption)
```

3.2.8.6 Получение и применение фильтра для сводной таблицы

```
# По названию поля сводной таблицы получаем фильтр
filter = pivotTable.getFilter("Category")

# Делаем элементы `Car` и `Technology` скрытыми
filter.setHidden("Car", true)
filter.setHidden("Technology", true)

# Делаем элемент `Furniture` видимым
filter.setHidden("Furniture", false)

# Применяем фильтр к сводной таблице
pivotTable.createPivotTableEditor().setFilter(filter).apply()
```

3.2.8.7 Получение полей из области фильтров

```
# Получение полей из области фильтров
pageFields = pivotTable.getPageFields()
# Перебираем все поля из области фильтров
for i in pageFields:
    fieldProps = pageFields[i].fieldProperties

    #  Далее используем поля структуры PivotTableFieldProperties:
    print(fieldProps.fieldName)
    print(fieldProps.fieldAlias)
    print(fieldProps.subtotalAlias)
```

3.2.8.8 Получение полей из области значений

```
# Получение полей из области значений
PivotTableValueFields valueFields = pivotTable->getValueFields()
# Перебираем все поля из области значений
for i in valueFields:
    valueField = valueFields[i]

    # Далее используем поля структуры PivotTableValueField:
    print(valueField.baseFieldName)
    print(valueField.valueFieldName)
    print(valueField.cellNumberFormat)
    print(valueField.totalFunction)
    print(valueField.customFormula)
```

3.2.8.9 Получение полей из области колонок

```
# Получение полей из области колонок
columnFields = pivotTable->getColumnFields()
# Перебираем все поля из области колонок
for i in columnFields:
    fieldProperties = columnFields[i].fieldProperties
    subtotalFunctions = columnFields[i].subtotalFunctions

    #  Далее используем поля структуры PivotTableCategoryField:
    print(fieldProperties.fieldName)
    print(fieldProperties.fieldAlias)
    print(fieldProperties.subtotalAlias)
```

3.2.8.10 Получение полей из области строк

```
# Получение полей из области строк
rowFields = pivotTable->getRowFields()
# Перебираем все поля из области строк
for i in rowFields:
    fieldProperties = rowFields[i].fieldProperties
    subtotalFunctions = rowFields [i].subtotalFunctions

    #  Далее используем поля структуры PivotTableCategoryField:
    print(fieldProperties.fieldName)
    print(fieldProperties.fieldAlias)
    print(fieldProperties.subtotalAlias)
```

3.2.8.11 Получение настроек отображения сводной таблицы

```
layoutSettings = pivotTable->getPivotTableLayoutSettings()

#  Далее используем поля структуры PivotTableLayoutSettings:
print(layoutSettings.reportLayout)
print(layoutSettings.pageFieldOrder)
print(layoutSettings.useGridDropZones)
print(layoutSettings.pageFieldWrapCount)
print(layoutSettings.displayFieldCaptions)
print(layoutSettings.indentForCompactLayout)
print(layoutSettings.valueFieldsOrientation)
print(layoutSettings.isMergeAndCenterLabelsEnabled)
```

3.2.8.12 Обновление сводной таблицы

```
# Пересчет и перезаполнение сводной таблицы в соответствии с исходными данными.
# Обновление сводной таблицы приводит к потере всех неподдерживаемых свойств.
pivotTable.update()
```

3.2.9 Работа с именованными выражениями

3.2.9.1 Получение именованного выражения

```
# Загруженный документ содержит именованное выражение

# Получаем именованное выражение с именем “Age_of_Majority”
ageOfMajority = document.getNamedExpressions().get("Age_of_Majority")
```

3.2.9.2 Получение именованного выражения таблицы

```
# Получаем именованное выражение таблицы с именем “Alice_Age”
aliceAge =
document.getBlocks().getTable(0).getNamedExpressions().get("Alice_Age")
```

3.2.9.3 Получение свойств именованного выражения

```
#  Получаем именованное выражение с именем "Alice_Age"
expression =
document.getBlocks().getTable(0).getNamedExpressions().get("Alice_Age")
#  Далее используем поля структуры NamedExpression:
name = expression.getName()
formula = expression.getExpression()
range = expression.getCellRange()
```

3.2.9.4 Получение коллекции именованных выражений

```
# Получение коллекции именованных выражений
enumerator =
document.getBlocks().getTable(0).getNamedExpressions().getEnumerator()

# Перебор коллекции именованных выражений
for namedExpression in enumerator:

    # Использование полей структуры NamedExpression
    print(namedExpression.getName())
    print(namedExpression.getExpression())
    print(namedExpression.getCellRange())
```

3.3 Работа со встроенными объектами

```
# Загруженный документ содержит именованное выражение
document = application.createDocument(sdk.DocumentType_Text)

# Все встроенные объекты в документе доступны в коллекции встроенных объектов.
# Имея коллекцию, используйте ее перечислитель для доступа к каждому
# встроенному объекту в документе.

enumerator = document.getRange().getInlineObjects().getEnumerator()
for inlineObject in enumerator:

    # Проверка на то, что встроенный объект является изображением.
    image = inlineObject.toImage()
    if image is not None:
        # Использовать изображение
        print("Has image")

# С помощью объекта Inline можно изменить положение встроенного объекта в
# текстовом документе, его размеры и стиль обтекания текстом.

enumerator = document.getRange().getInlineObjects().getEnumerator()
for inlineObject in enumerator:

    # Позиция встроенного объекта не может быть задана, если стиль переноса
```

```
    # текста - inline.
    # Сначала его следует изменить на тип не inline.

    inlineFrame = inlineObject.getFrame()
    if (inlineFrame.getWrapType() == sdk.TextWrapType.Inline)
        inlineFrame.setWrapType(sdk.TextWrapType.TopAndBottom):

        # Используя поля HorizontalTextAnchoredPosition,
        # VerticalTextAnchoredPosition
        # и offset, можно задать положение встроенных объектов
        # в текстовом документе с относительными смещениями
        # в единицах измерения.

        inlineFrame = inlineObject.getFrame()

        position = sdk.TextAnchoredPosition()

        horizontalPosition = sdk.HorizontalTextAnchoredPosition(
                             sdk.HorizontalRelativeTo.Page, 12.f)
        position.horizontal = horizontalPosition

        verticalPosition = sdk.VerticalTextAnchoredPosition(
                           sdk.VerticalRelativeTo.PageTopMargin, 122.f)
        position.vertical = verticalPosition

        inlineFrame.setPosition(position)

        # Используя поля HorizontalTextAnchoredPosition,
        # VerticalTextAnchoredPosition и alignment,
        # можно изменить положение встроенных объектов в текстовом
        # документе с его относительным выравниванием.

        inlineFrame = inlineObject.getFrame()

        position = sdk.TextAnchoredPosition()

        horizontalPosition = sdk.HorizontalTextAnchoredPosition(
                             sdk.HorizontalRelativeTo.Page,
                             sdk.HorizontalAnchorAlignment.Center)
        position.horizontal = horizontalPosition;

        verticalPosition = sdk.VerticalTextAnchoredPosition(
                           sdk.VerticalRelativeTo.PageTopMargin,
                           sdk.VerticalAnchorAlignment.Top)
        position.vertical = verticalPosition

        inlineFrame.setPosition(position)

        # Используя смещения TextAnchoredPosition по горизонтали и вертикали,
        # можно установить абсолютное положение встроенного объекта в текстовом
        # документе, которое является относительным положением со значениями
        # HorizontalRelativeTo::Column и VerticalRelativeTo::Page.

        inlineFrame = inlineObject.getFrame()

        position = sdk.TextAnchoredPosition()
```

```
        position.horizontal = sdk.HorizontalTextAnchoredPosition(
                              sdk.HorizontalRelativeTo.Column, 125.f)
        position.vertical = sdk.VerticalTextAnchoredPosition(
                            sdk.VerticalRelativeTo.Page, 345.f)

        inlineFrame.setPosition(position)

        # Размеры встроенных объектов могут быть изменены с помощью
        # объекта фрейма.
        inlineFrame = inlineObject.getFrame()

        size = sdk.Size(300.f, 400.f)
        inlineFrame.setDimensions(size)
```

4 СПРАВОЧНИК КЛАССОВ, СТРУКТУР И МЕТОДОВ

4.1 Поддерживаемые типы документов DocumentType

Поддерживаемые типы документов представлены в Таблице 3.

Таблица 3 - Типы документов

Имя константы типа документа	Использование	Тип документа
`DocumentType_Text`	Для работы с текстовыми документами – файлы DOCX, ODT, XODT, TXT.	Text
`DocumentType_Workbook`	Для работы с табличными документами – файлы XLSX, ODS, XODS.	Workbook
`DocumentType_Presentation`	Для работы с презентационными документами – файлы PPTX, ODP. Работа с презентационными документами средствами Document API в настоящий момент не поддерживается.	Presentation

Работа с презентационными документами средствами Document API в настоящий момент не поддерживается.

4.2 Поддерживаемые форматы документов DocumentFormat

Поддерживаемые форматы документов представлены в Таблице 4.

Таблица 4 - Поддерживаемые форматы документов

Имя константы формата документа	Формат документа	Использование
`DocumentFormat_PlainText`	PlainText	Используется для работы с файлами TXT.
`DocumentFormat_DSV`	DSV	Используется для работы с табличными данными в текстовой форме (CSV, DSV). Строка текста содержит одно или несколько полей данных, разделенных запятыми или иным разделителем.
`DocumentFormat_OXML`	OXML	Используется для работы с текстовыми (DOCX) или табличными (XLSX) документами в формате Open Office XML.
`DocumentFormat_ODF`	ODF	Используется для работы с текстовыми (ODT) или табличными (ODS) документами формата Open Document Format (ГОСТ Р ИСО/МЭК 26300-2010).
`DocumentFormat_HTML`	HTML	Используется для работы с веб-документами в формате HTML. Работа с веб-документами в

Имя константы формата документа	Формат документа	Использование
		формате HTML средствами Document API в настоящий момент не поддерживается.
`DocumentFormat_PDF`	PDF	Используется для работы с документами в формате Portable Document Format (PDF), версии 1.4.
`DocumentFormat_PDFA`	PDFA	Используется для работы с документами в формате Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b).

Средствами Document API поддерживается операция экспорта документа только в формат PDF/A-1b.

Средствами MyOffice Document API не поддерживается работа с текстовыми и табличными документами в двоичном формате Microsoft Word (DOC) и Microsoft Excel (XLS, XLSB), а также документами, содержащими макрокоманды VBA (DOCM, XLSM).

4.3 Форматы экспорта документов

Поддерживаемые форматы экспорта документов представлены в Таблице 5.

Таблица 5 - Форматы экспорта документов

Имя константы формата экспорта документа	Формат экспорта документа	Использование
`ExportFormat_PDFA1`	PDF/A-1b	Для экспорта документа в формат PDF/A-1b.

Средствами Document API поддерживается операция экспорта документа только в формат PDF/A-1b.

4.4 Неподдерживаемые свойства документа SaveUnsupportedFeature

Имеются свойства документа, которые могут быть утеряны при сохранении документа. Данные свойства описаны в Таблице 6.

Таблица 6 - Форматы неподдерживаемых свойств документа

Имя константы неподдерживаемого свойства документа	Описание
`SaveUnsupportedFeature_CommentsInHeaderFooter`	Комментарии в колонтитулах.

Имя константы неподдерживаемого свойства документа	Описание
SaveUnsupportedFeature_CommentsInFootnote	Комментарии в сносках.
SaveUnsupportedFeature_MixedNoteAlignment	Параграфы с разным выравниванием в заметках.

4.5 Системы адресации ячеек

Поддерживаемые системы адресации ячеек (стили ссылок) в табличном документе представлены в Таблице 7.

Таблица 7 - Системы адресации ячеек в табличном документе

Имя константы системы адресации ячеек	Система адресации ячеек	Описание
FormulaType_A1	A1	Вариант A1 соответствует наиболее распространенной системе адресации ячеек, при которой столбцы задаются буквами, а строки – числами.
FormulaType_R1C1	R1C1	Вариант R1C1 соответствует альтернативной системе адресации ячеек, при которой столбцы и строки задаются числами.

4.6 Кодировки документов

Поддерживаемые кодировки документов представлены в Таблице 8.

Таблица 8 - Кодировки документов

Имя константы кодировки документа	Кодировка
Encoding_Unknown	Невозможно определить кодировку
Encoding_UTF8	UTF8
Encoding_UTF16BE	UTF16BE
Encoding_UTF16LE	UTF16LE
Encoding_UTF32BE	UTF32BE
Encoding_UTF32LE	UTF32LE
Encoding_Windows1250	Windows1250

Имя константы кодировки документа	Кодировка
Encoding_Windows1251	Windows1251
Encoding_Windows1252	Windows1252
Encoding_ISO8859Part5	ISO8859Part5
Encoding_KOI8R	KOI8R
Encoding_KOI8U	KOI8U
Encoding_CP866	CP866

4.7 Типы горизонтального выравнивания текста

Поддерживаемые типы горизонтального выравнивания текста представлены в Таблице 9.

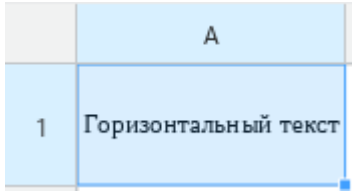
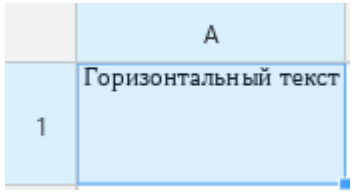
Таблица 9 - Типы горизонтального выравнивания текста

Имя константы типа выравнивания текста	Описание
Alignment_Default	Выравнивание текста по умолчанию.
Alignment_Left	Выравнивание текста по левому краю.
Alignment_Center	Выравнивание текста по центру.
Alignment_Right	Выравнивание по правому краю.
Alignment_Justify	Выравнивание по ширине.
Alignment_Distributed	Распределенное выравнивание, при применении между словами добавляется пробел, чтобы оба края каждой строки были выровнены по обеим сторонам. Последняя строка в абзаце также выравнивается по обеим сторонам, но если строка состоит из одного слова, то выравнивание по правой стороне не осуществляется.
Alignment_Fill	Распределение текста по горизонтали – заполнение строки текстом.

4.8 Типы выравнивания текста в ячейке таблицы по вертикали

Поддерживаемые типы выравнивания текста в ячейках таблицы по вертикали, представлены в Таблице 10.

Таблица 10 - Типы выравнивания текста в ячейке таблицы по вертикали

Имя константы типа выравнивания	Представление в интерфейсе	
`VerticalAlignment_Bottom`		
`VerticalAlignment_Center`		
`VerticalAlignment_Top`		

4.9 Типы расположения текста в ячейках таблиц

Поддерживаемые типы размещения текста в ячейках таблиц представлены в Таблице 11.

Таблица 11 - Типы расположения текста в ячейках таблиц

Имя константы типа отображения длинного текста	Описание типа отображения
`TextLayout_SingleLine`	Отображение длинного текста в одну строку с наложением на соседние ячейки.
`TextLayout_WrapByWords`	Перенос данных внутри ячейки по словам.

Имя константы типа отображения длинного текста	Описание типа отображения
TextLayout_ShrinkSizeToFitWidth	Текст или число отображаются так, чтобы содержимое поместилось в ячейке полностью.

4.10 Типы линий LineStyle

Поддерживаемые типы линий представлены в Таблице 12.

Таблица 12 - Типы линий

Имя константы типа линии	Наименование типа линии
LineStyle_NoLine	Нет границ.
LineStyle_Solid	Обычная линия.
LineStyle_Dot	Пунктирная линия.
LineStyle_Dash	Штриховая линия.
LineStyle_DashDot	Штрихпунктирная линия.
LineStyle_DotDotDash	Штрихпунктирная линия с двумя точками.
LineStyle_Double	Двойная линия.
LineStyle_Wave	Волнистая линия.

4.11 Типы надстрочного и подстрочного форматирования ScriptPosition

Поддерживаемые типы надстрочного/подстрочного форматирования представлены в Таблице 13.

Таблица 13 - Типы надстрочного/подстрочного форматирования

Имя константы типа надстрочного/подстрочного форматирования	Наименование типа форматирования
ScriptPosition_SuperScript	Надстрочный
ScriptPosition_SubScript	Подстрочный
ScriptPosition_NormalScript	Нормальный

4.12 Типы форматов ячеек CellFormat

Поддерживаемые форматы ячеек представлены в Таблице 14.

Таблица 14 - Форматы ячеек

Имя константы формата ячеек	Описание	Пример
CellFormat_General	Общий	12

Имя константы формата ячеек	Описание	Пример
CellFormat_Percentage	Процентный	120,00%
CellFormat_Number	Числовой	12,00
CellFormat_Text	Текстовый	12
CellFormat_Currency	Денежный	12 000,00₽
CellFormat_Accounting	Финансовый	12 000,00₽
CellFormat_Date	Дата	01.01.2020
CellFormat_Time	Время	0:00:00
CellFormat_DateTime	Дата и время	12.12.2020 0:00:00
CellFormat_Fraction	Дробный	12 1/5
CellFormat_Scientific	Экспоненциальный	1,22E+01
CellFormat_Custom	Пользовательский	

4.13 Типы форматов даты DatePatterns

Поддерживаемые типы форматов даты представлены в Таблице 15.

Таблица 15 - Форматы даты

Имя константы формата даты	Формат
DatePatterns_DayMonthTextLongYearLong	'mmmm dd, yyyy' для языка en_US.
DatePatterns_FullDate	'день недели, mmmm dd, yyyy' для языка en_US.
DatePatterns_DayMonthNumberLongYearLong	'mm/dd/yyyy' для языка en_US.
DatePatterns_DayMonthNumberLongYearShort	'm/dd/yy' для языка en_US.
DatePatterns_DayMonthNumberShortYearShort	'dd-mmm' для языка en_US.
DatePatterns_DayMonthTextShort	'mmm-yy' для языка en_US.
DatePatterns_DayMonthTextShortYearShort	'mmm dd, yy' для языка en_US.
DatePatterns_DayMonthYearLongNonLocalizableHyphen	Нелокализуемый шаблон 'dd-mm-yyyy'.
DatePatterns_DayMonthYearLongNonLocalizableSlash	Нелокализуемый шаблон 'dd/mm/yyyy'.

4.14 Типы форматов времени TimePatterns

Поддерживаемые типы форматов времени представлены в Таблице 16.

Таблица 16 - Форматы времени

Имя константы формата времени	Формат
TimePatterns_ShortTime	'hh:mm AM/PM' для языка en_US.
TimePatterns_LongTime	'hh:mm:ss AM/PM' для языка en_US.

4.15 Типы межстрочного интервала LineSpacingRule

Поддерживаемые типы межстрочного интервала представлены в Таблице 17.

Таблица 17 - Типы межстрочного интервала

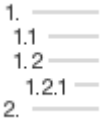
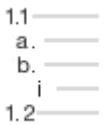
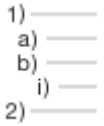
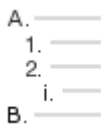
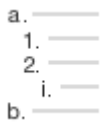
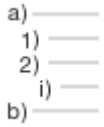
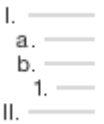
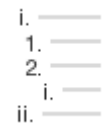
Имя константы межстрочного интервала	Наименование типа	Описание
`LineSpacingRule_Multiple`	Множитель	Межстрочный интервал с использованием множителя.
`LineSpacingRule_Exact`	Точно	Межстрочный интервал с использованием точного значения.
`LineSpacingRule_AtLeast`	Минимум	Межстрочный интервал с использованием минимального значения.

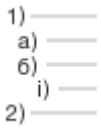
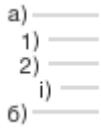
4.16 Типы схем абзацев ListSchema

Типы схем абзацев представлены в Таблице 18.

Таблица 18 - Типы типов схем абзацев

Имя константы типа схемы абзаца	Описание типа схемы абзаца	Изображение
`ListSchema_Unknown`	Неизвестно.	
`ListSchema_UnknownBullet`	Список без маркера.	Соответствует варианту «нет»
`ListSchema_UnknownNumbering`	Нумерация без номера.	Соответствует варианту «нет»
`ListSchema_BulletCircleSolid`	Список с маркерами в виде круга.	
`ListSchema_BulletCircleContour`	Список с маркерами в виде окружности.	
`ListSchema_BulletSquareSolid`	Список с маркерами в виде квадрата	
`ListSchema_BulletDiamondDots`	Список с маркерами в виде четырех ромбов.	

Имя константы типа схемы абзаца	Описание типа схемы абзаца	Изображение
`ListSchema_Unknown`	Неизвестно.	
`ListSchema_BulletHyphen`	Список с маркерами в виде дефиса.	
`ListSchema_BulletConcaveArrowSolid`	Список с маркерами в виде вогнутой стрелки.	
`ListSchema_BulletCheckmark`	Список с маркерами в виде галочки.	
`ListSchema_EnumeratorDecimalDot`	Десятичная нумерация с точкой.	
`ListSchema_EnumeratorDecimalDotMultiLevel`	Многоуровневая десятичная нумерация с точкой.	
`ListSchema_EnumeratorDecimalBracket`	Десятичная нумерация со скобкой.	
`ListSchema_EnumeratorLatinUppercaseDot`	Нумерация латинскими прописными буквами с точкой.	
`ListSchema_EnumeratorLatinLowercaseDot`	Нумерация латинскими строчными буквами с точкой.	
`ListSchema_EnumeratorLatinLowercaseBracket`	Нумерация латинскими строчными буквами со скобкой.	
`ListSchema_EnumeratorRomanUppercaseDot`	Нумерация римскими прописными цифрами с точкой.	
`ListSchema_EnumeratorRomanLowercaseDot`	Нумерация римскими строчными цифрами с точкой.	

Имя константы типа схемы абзаца	Описание типа схемы абзаца	Изображение
ListSchema_Unknown	Неизвестно.	
ListSchema_EnumeratorDecimalRussianBracket	Десятичная нумерация через запятую со скобкой.	
ListSchema_EnumeratorRussianLowercaseBracket	Нумерация с русскими строчными буквами со скобкой.	

4.17 Типы размещения знака валюты CurrencySignPlacement

Типы вариантов размещения знака валюты представлены в Таблице 19.

Таблица 19 - Типы вариантов размещения знака валюты

Имя константы варианта размещения знака валюты	Вариант размещения знака валюты	Пример
CurrencySignPlacement_Prefix	До значения.	$12.00
CurrencySignPlacement_Suffix	После значения.	12,00 ₽

4.18 Типы ориентации страницы PageOrientation

Поддерживаемые типы ориентации страницы представлены в Таблице 20.

Таблица 20 - Типы ориентации страницы

Имя константы типа ориентации страницы	Наименование типа ориентации страницы	Изображение
PageOrientation_Portrait	Книжная	
PageOrientation_Landscape	Альбомная	

4.19 Типы отслеживаемых изменений TrackedChangeType

Поддерживаемые типы отслеживаемых изменений представлены в Таблице 21.

Таблица 21 - Типы отслеживаемых изменений

Имя константы	Наименование типа отслеживания изменений
TrackedChangeType_Added	Добавленные изменения.
TrackedChangeType_Removed	Удаленные изменения.

4.20 Типы колонтитулов страниц документов

Поддерживаемые типы колонтитулов страниц документов представлены в Таблице 22.

Таблица 22 - Типы колонтитулов страниц документа

Имя константы типа колонтитула	Наименование типа колонтитула
HeaderFooterType_Header	Верхний.
HeaderFooterType_Footer	Нижний.

4.21 Масштабирование при печати табличных документов

Поддерживаемые варианты масштабирования при печати табличных документов представлены в Таблице 23.

Таблица 23 - Варианты масштабирования при печати табличных документов

Имя константы варианта масштабирования	Вариант масштабирования
WorksheetPrinterFitType_ActualSize	Фактический размер.
WorksheetPrinterFitType_ByPageScale	По масштабу страницы.
WorksheetPrinterFitType_ByPageBreaksOnly	По разрыву страниц.
WorksheetPrinterFitType_FitToPage	Вписать в страницу.
WorksheetPrinterFitType_FitToWidth	Вписать по ширине.
WorksheetPrinterFitType_FitToHeight	Вписать по высоте.

4.22 Выбор страниц для экспорта и печати

Варианты отбора страниц для экспорта и печати представлены в Таблице 24.

Таблица 24 - Варианты отбора страниц для экспорта и печати

Имя константы варианта отбора страниц	Вариант
PageParity_Odd	Печать только нечетных страниц.
PageParity_Even	Печать только четных страниц.
PageParity_All	Печать всех страниц.

4.23 Диапазон для экспорта и печати PrintingScopeType

Варианты выбора для печатаемого диапазона страниц представлены в Таблице 25.

Таблица 25 - Диапазон страниц для экспорта и печати

Имя константы	Вариант
Type_PrintArea	Выбранная область печати (по умолчанию).
Type_WholeSheet	Печать всего документа.

4.24 Типы окончания линии

В Таблице 26 приведены типы окончания линий.

Таблица 26 - Типы окончания линии

Имя константы типа окончания линии	Описание
LineEndingStyle_Arrow	
LineEndingStyle_Diamond	
LineEndingStyle_Oval	
LineEndingStyle_Stealth	
LineEndingStyle_Triangle	
LineEndingStyle_None	

4.25 Типы идентификаторов цветов тем

В Таблице 27 представлены типы идентификаторов цветов тем.

Таблица 27 - Типы идентификаторов цветов тем

Имя константы типа идентификатора темы	Описание
ThemeColorID_Background1	Фон1
ThemeColorID_Text1	Текст1
ThemeColorID_Background2	Фон2
ThemeColorID_Text2	Текст2
ThemeColorID_Dark1	Темная1
ThemeColorID_Dark2	Темная2
ThemeColorID_Light1	Светлая1
ThemeColorID_Light2	Светлая2
ThemeColorID_Accent1	Акцент1
ThemeColorID_Accent2	Акцент2
ThemeColorID_Accent3	Акцент3
ThemeColorID_Accent4	Акцент4
ThemeColorID_Accent5	Акцент5

Имя константы типа идентификатора темы	Описание
ThemeColorID_Accent6	Акцент6
ThemeColorID_Hyperlink	Гиперссылка
ThemeColorID_FollowedHyperlink	Следующая гиперссылка

4.26 Варианты обтекания текстом встроенного объекта

В Таблице 28 представлены варианты обтекания текстом встроенного объекта.

Таблица 28 - Варианты обтекания текстом встроенного объекта

Имя константы варианта обтекания текстом встроенного объекта	Описание
TextWrapType_Inline	Встроенный объект располагается в тексте.
TextWrapType_InFrontOfText	Встроенный объект располагается перед текстом.
TextWrapType_BehindText	Встроенный объект располагается за текстом.
TextWrapType_TopAndBottom	Текст располагается сверху и снизу встроенного объекта.
TextWrapType_Square	Текст располагается вокруг прямоугольной рамки встроенного объекта.
TextWrapType_Through	Текст обтекает встроенный объект по сторонам и внутри.
TextWrapType_Tight	Текст обтекает встроенный объект по сторонам.

4.27 Типы размещения объекта по вертикали

В Таблице 29 представлены типы размещения объекта относительно закрепленной позиции по вертикали.

Таблица 29 - Типы размещения объекта относительно закрепленной позиции по вертикали

Имя константы типа размещения объекта относительно закрепленной позиции по вертикали	Описание
VerticalRelativeTo_Character	Символ.
VerticalRelativeTo_BaseLine	Базовая линия.
VerticalRelativeTo_Paragraph	Абзац.
VerticalRelativeTo_Page	Страница.
VerticalRelativeTo_PageContent	Содержимое страницы.
VerticalRelativeTo_PageTopMargin	Верхнее поле страницы.
VerticalRelativeTo_PageBottomMargin	Нижнее поле страницы.
VerticalRelativeTo_PageInsideMargin	Внутреннее поле страницы.
VerticalRelativeTo_PageOutsideMargin	Внешнее поле страницы.

4.28 Типы размещения объекта по горизонтали

В Таблице 30 представлены типы размещения объекта относительно закрепленной позиции по горизонтали.

Таблица 30 - Типы размещения объекта относительно закрепленной позиции по горизонтали

Имя константы типа размещения объекта относительно закрепленной позиции по вертикали	Описание
HorizontalRelativeTo_Character	Символ.
HorizontalRelativeTo_Column	Столбец.
HorizontalRelativeTo_ColumnLeftMargin	Левое поле столбца.
HorizontalRelativeTo_ColumnRightMargin	Правое поле столбца.
HorizontalRelativeTo_ColumnInsideMargin	Внутреннее поле столбца.
HorizontalRelativeTo_ColumnOutsideMargin	Внешнее поле столбца.
HorizontalRelativeTo_Page	Страница.
HorizontalRelativeTo_PageContent	Содержимое страницы.
HorizontalRelativeTo_PageLeftMargin	Левое поле страницы.
HorizontalRelativeTo_PageRightMargin	Правое поле страницы.
HorizontalRelativeTo_PageInsideMargin	Внутреннее поле страницы.
HorizontalRelativeTo_PageOutsideMargin	Внешнее поле страницы.

4.29 Типы выравнивания объекта по вертикали

В Таблице 31 представлены типы выравнивания объекта относительно закрепленной позиции по вертикали.

Таблица 31 - Типы выравнивания объекта относительно закрепленной позиции по вертикали

Имя константы типа выравнивания объекта относительно закрепленной позиции по вертикали	Описание
VerticalAnchorAlignment_Top	По верхнему краю.
VerticalAnchorAlignment_Bottom	По нижнему краю.
VerticalAnchorAlignment_Center	По центру.
VerticalAnchorAlignment_Inside, VerticalAnchorAlignment_Outside	По границам.

4.30 Типы выравнивания объекта по горизонтали

В Таблице 32 представлены типы выравнивания объекта относительно закрепленной позиции по горизонтали.

Таблица 32 - Типы выравнивания объекта относительно закрепленной позиции по горизонтали

Имя константы типа выравнивания объекта относительно закрепленной позиции по горизонтали	Описание
HorizontalAnchorAlignment_Left	По верхнему краю.
HorizontalAnchorAlignment_Right	По нижнему краю.
HorizontalAnchorAlignment_Center	По центру.
HorizontalAnchorAlignment_Inside, HorizontalAnchorAlignment_Outside	По границам.

4.31 Масштабирование при печати табличных документов

В Таблице 33 представлены варианты масштабирования при печати табличных документов.

Таблица 33 - Варианты масштабирования при печати табличных документов

Имя константы варианта масштабирования при печати табличных документов	Описание
WorksheetPrinterFitType_ActualSize	Фактический размер.
WorksheetPrinterFitType_ByPageScale	По масштабу страницы.
WorksheetPrinterFitType_ByPageBreaksOnly	По разрыву страниц.
WorksheetPrinterFitType_FitToPage	Вписать в страницу.
WorksheetPrinterFitType_FitToWidth	Вписать по ширине.
WorksheetPrinterFitType_FitToHeight	Вписать по высоте.

4.32 Форматирование ячеек таблицы

4.32.1 Шаблоны даты DatePatterns

В Таблице 34 содержатся варианты форматов даты.

Таблица 34 - Форматы даты

Имя константы шаблона даты	Описание
DatePatterns_DayMonthTextLongYearLong	'mmmm dd, yyyy' для языка en_US.
DatePatterns_FullDate	'день недели, mmmm dd, yyyy' для языка en_US.
DatePatterns_DayMonthNumberLongYearLong	'mm/dd/yyyy' для языка en_US.
DatePatterns_DayMonthNumberLongYearShort	'm/dd/yy' для языка en_US.
DatePatterns_DayMonthNumberShortYearShort	'dd-mmm' для языка en_US.
DatePatterns_DayMonthTextShort	'mmm-yy' для языка en_US.
DatePatterns_DayMonthTextShortYearShort	'mmm dd, yy' для языка en_US.
DatePatterns_DayMonthYearLongNonLocalizableHyphen	нелокализуемый шаблон 'dd-mm-yyyy'.
DatePatterns_DayMonthYearLongNonLocalizableSlash	нелокализуемый шаблон 'dd/mm/yyyy'.

4.32.2 Шаблоны времени TimePatterns

В Таблице 35 содержатся варианты форматов времени.

Таблица 35 - Форматы времени

Имя константы шаблона времени	Описание
TimePatterns_ShortTime	'hh:mm AM/PM' для языка en_US.
TimePatterns_LongTime	'hh:mm:ss AM/PM' для языка en_US.

4.32.3 Класс AccountingCellFormatting

Класс `AccountingCellFormatting` содержит параметры для финансового формата ячеек таблицы.

```
class AccountingCellFormatting(object)
```

Свойства класса:

- `decimalPlaces` – количество десятичных позиций;
- `symbol` – символ денежной единицы;
- `localeCode` – идентификатор кода языка (MS-LCID);
- `fillSymbol` – символ заполнения;
- `useThousandsSeparator` – использовать разделитель для тысячных;
- `currencySignPlacement` – тип размещения знака валюты `CurrencySignPlacement`.

4.32.4 Класс PercentageCellFormatting

Класс `PercentageCellFormatting` содержит параметр для процентного формата ячеек таблицы.

```
class PercentageCellFormatting(object)
```

Свойство класса:

- `decimalPlaces` – количество десятичных позиций.

4.32.5 Класс NumberCellFormatting

Класс `NumberCellFormatting` содержит параметры для числового формата ячеек таблицы.

```
class NumberCellFormatting(object)
```

Свойства класса:

- `decimalPlaces` – количество десятичных позиций;

- `useThousandsSeparator` – использовать разделитель для тысячных;
- `useRedForNegative` – использовать красный цвет для отрицательных значений;
- `useBracketsForNegative` – использовать скобки для отрицательных значений;
- `hideSign` – скрывать знак «минус» для отрицательных значений.

4.32.6 Класс CurrencyCellFormatting

Класс `CurrencyCellFormatting` содержит параметры для денежного формата ячеек таблицы.

```
class CurrencyCellFormatting(object)
```

Свойства класса:

- `decimalPlaces` – количество десятичных позиций;
- `symbol` – символ денежной единицы;
- `localeCode` – идентификатор кода языка (MS-LCID);
- `useThousandsSeparator` – использовать разделитель для тысячных;
- `useRedForNegative` – использовать красный цвет для отрицательных значений;
- `useBracketsForNegative` – использовать скобки для отрицательных значений;
- `hideSign` – скрывать знак «минус» для отрицательных значений.

4.32.7 Класс DateTimeCellFormatting

Класс `DateTimeCellFormatting` содержит параметры для формата ячеек таблицы типа Дата и Время.

```
class DateTimeCellFormatting(object)
```

Свойства класса:

- `dateListID` – формат даты `DatePatterns`;
- `timeListId` – формат времени `TimePatterns`.

4.32.8 Класс FractionCellFormatting

Класс `FractionCellFormatting` представляет параметры для дробного формата ячеек таблицы.

```
class FractionCellFormatting(object)
```

Свойства класса:

- `minNumeratorDigits` – количество позиций числителя;
- `minDenominatorDigits` – количество позиций знаменателя;
- `denominatorValue` – знаменатель.

4.32.9 Класс ScientificCellFormatting

Класс `ScientificCellFormatting` представляет параметры для экспоненциального формата ячеек таблицы.

```
class ScientificCellFormatting(object)
```

Свойства класса:

- `decimalPlaces` – количество десятичных позиций;
- `minExponentDigits` – минимальное количество позиций экспоненты.

4.33 Класс ColorRGBA

Позволяет настроить пользовательский цвет для оформления элементов документа, текста, границ таблиц и т. п. Для описания цвета используется расширенная цветовая модель RGB, позволяющая установить непрозрачность цвета.

```
class ColorRGBA(object)
```

Список свойств:

- `r` – значение компонента красного цвета;
- `g` – значение компонента зеленого цвета;
- `b` – значение компонента синего цвета;
- `a` – значение прозрачности цвета.

Список методов:

```
def __eq__(other)
def __ne__(other)
```

4.33.1 ColorRGBA.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `ColorRGBA`.

```
ColorRGBA.__eq__(other)
```

4.33.2 ColorRGBA.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `ColorRGBA`.

```
ColorRGBA.__ne__(other)
```

4.34 Класс UserInfo

Предоставляет информацию о пользователе.

```
class UserInfo(object)
```

Список свойств:

- `name` – имя пользователя;
- `email` – адрес электронной почты пользователя.

Список методов:

```
def __eq__(other)
def __ne__(other)
```

4.34.1 UserInfo.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `UserInfo`.

```
UserInfo.__eq__(other)
```

4.34.2 UserInfo.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `UserInfo`.

```
UserInfo.__ne__(other)
```

4.35 Класс DateTime

Предоставляет дату и время с точностью до секунды.

```
class DateTime(object)
```

Список свойств:

- `year` – год;
- `month` – месяц;
- `day` – день;
- `hour` – часы;

- `minute` – минуты;
- `second` – секунды.

Список методов:

```
def __eq__(other)
def __ne__(other)
```

4.35.1 DateTime.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `DateTime`.

```
DateTime.__eq__(other)
```

4.35.2 DateTime.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `DateTime`.

```
DateTime.__ne__(other)
```

4.36 Класс PageProperties

Содержит размеры страницы в пунктах (pt). Например, для листа формата A4 значение ширины составляет 595,29 pt, высоты – 841,90 pt (без округления).

```
class PageProperties(object)
```

Список свойств:

- `height` – высота страницы в пунктах;
- `width` – ширина страницы в пунктах;
- `margins` – поля страницы `Insets`.

Список методов:

```
def __eq__(other)
def __ne__(other)
```

4.36.1 PageProperties.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `PageProperties`.

```
PageProperties.__eq__(other)
```

4.36.2 PageProperties.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `PageProperties`.

```
PageProperties.__ne__(other)
```

4.37 Класс Insets

Класс `Insets` описывает поля страницы (`left`, `right`, `top`, `bottom`).

```
class Insets(object)
```

Список свойств:

- `left` – левое поле страницы;
- `right` – правое поле страницы;
- `top` – верхнее поле страницы;
- `bottom` – нижнее поле страницы.

Список методов:

```
def __eq__(other)
def __ne__(other)
```

4.37.1 Insets.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `Insets`.

```
Insets.__eq__(other)
```

4.37.2 Insets.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `Insets`.

```
Insets.__ne__(other)
```

4.38 Класс Comment

Предоставляет доступ к свойствам комментария примененного к области текста.

```
class Comment(object)
```

Список методов:

```
def getRange()
def getText()
def getInfo()
```

```
def getReplies()
def isResolved()
```

4.38.1 Comment.getRange

Возвращает объект `Range`, соответствующий комментарию.

```
Comment.getRange()
```

4.38.2 Comment.getText

Возвращает текст комментария.

```
Comment.getText()
```

4.38.3 Comment.getAudioUrl

Для открытого документа возвращает временный путь к файлу аудиокомментария в формате MP3.

```
Comment.getAudioUrl()
```

4.38.4 Comment.getInfo

Предоставляет доступ к отслеживаемой информации об авторе и дате/времени изменения комментария (`TrackedChangeInfo`: автор изменения, дата и т. д).

```
Comment.getInfo()
```

4.38.5 Comment.getReplies

Средство доступа к ответам комментариев `Comments`.

```
Comment.getReplies()
```

4.38.6 Comment.isResolved

Возвращает `true`, если комментарий решен.

```
Comment.isResolved()
```

4.39 Класс Comments

Предоставляет доступ к коллекции комментариев диапазона.

```
class Comments(object)
```

Пример использования:

```
comments = document.getComments()
for comment in comments:
    dt = '{}:{} {}/{}/{}'.format(comment.getInfo().timeStamp.hour,
                                 comment.getInfo().timeStamp.minute,
                                 comment.getInfo().timeStamp.day,
```

```
                                    comment.getInfo().timeStamp.month,
                                    comment.getInfo().timeStamp.year)
    trc = 'Пользователь {} ' \
            'добавил {} ' \
            'комментарий \"{}\"'.format(comment.getInfo().author.name,
                                       dt,
                                       str(comment.getText()).rstrip())
    print(trc)
```

4.40 Класс Position

Представляет местоположение в документе при вставке нового объекта.

```
class Position(object)
```

Список методов:

```
def insertText(text)
def insertTable(rowsCount, columnsCount, name)
def insertPageBreak()
def insertLineBreak()
def insertBookmark(name)
def insertImage(url, size)
def insertSectionBreak(breakType)
def removeBackward(count=1)
def removeForward(count=1)
def __eq__(other)
def __ne__(other)
```

4.40.1 Position.insertText

Вставляет текст в указанную позицию.

```
Position.insertText(text)
```

4.40.2 Position.insertTable

Вставляет таблицу в текущую позицию, возвращает объект `Table`.

```
Position.insertTable(rowsCount, columnsCount, name)
```

Параметры:

- `rowsCount` – количество строк в таблице;
- `columnsCount` – количество столбцов в таблице;
- `name` – наименование таблицы.

4.40.3 Position.insertPageBreak

Вставляет разрыв страницы в текущую позицию.

```
Position.insertPageBreak()
```

4.40.4 Position.insertLineBreak

Вставляет перевод строки, не разрывающий абзац, в текущую позицию.

```
Position.insertLineBreak()
```

4.40.5 Position.insertBookmark

Вставляет закладку с наименованием `name` в текущую позицию.

```
Position.insertBookmark(name)
```

4.40.6 Position.insertImage

Вставляет изображение из файла формата PNG или JPG в текущую позицию. С помощью параметра `url` задается полный путь к файлу. Параметр `size`, задает геометрические размеры изображения для вставки (в типографических пунктах).

```
Position.insertImage(url, size)
```

Параметры:

- `url` – полный путь к файлу изображения;
- `size` – геометрические размеры изображения для вставки.

4.40.7 Position.insertSectionBreak

Вставляет разрыв раздела типа `SectionBreakType` в указанную позицию.

```
Position.insertSectionBreak(breakType)
```

4.40.8 Position.removeBackward

Удаляет `count` объектов (символов, картинок и т.д.) до текущей позиции.

```
Position.removeBackward(count)
```

4.40.9 Position.removeForward

Удаляет `count` объектов (символов, картинок и т.д.) после текущей позиции.

```
Position.removeForward(count)
```

4.40.10 Position.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `Position`.

```
Position.__eq__(other)
```

4.40.11 Position.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `Position`.

```
Position.__ne__(other)
```

4.41 Тип SectionBreakType

Тип `SectionBreakType` определяет тип начала следующей секции после вставки разрыва раздела (см. Таблица 36).

Таблица 36 - Тип начала секции

Имя константы типа начала секции	Описание
`SectionBreakType_NextPage`	Следующий раздел начинается с новой страницы.
`SectionBreakType_Continuous`	Следующий раздел продолжается на текущей странице без вставки разрыва страницы.
`SectionBreakType_EvenPage`	Следующий раздел начинается на ближайшей четной странице.
`SectionBreakType_OddPage`	Следующий раздел начинается на ближайшей нечетной странице.

4.42 Класс Block

Является базовым классом для всех блоков документа.

```
class Block(object)
```

Список методов:

```
def toParagraph()
def toTable()
def toShape()
def toField()

def getRange()
def remove()
def getSection()
```

4.42.1 Block.toParagraph

Если блок является в абзацем, возвращает тип `Paragraph`. В противном случае возвращает `none` .

```
Block.toParagraph()
```

4.42.2 Block.toTable

Если блок является таблицей, возвращает тип `Table`. В противном случае возвращает `none`.

```
Block.toTable()
```

4.42.3 Block.toShape

Если блок является фигурой, возвращает тип `Shape`. В противном случае возвращает `none`.

```
Block.toShape()
```

4.42.4 Block.toField

Если блок является полем, возвращает тип `Field`. В противном случае возвращает `none`.

```
Block.toField()
```

4.42.5 Block.getRange

Возвращает диапазон `Range`, в котором содержится данный блок.

```
Block.getRange()
```

4.42.6 Block.remove

Удаляет блок из документа. Текущий экземпляр объекта `Block` становится недействительным.

```
Block.remove()
```

4.42.7 Block.getSection

Возвращает раздел `Section`, содержащийся в блоке.

```
Block.getSection
```

4.43 Класс Field

Класс `Field` наследуется от класса `Block` и предназначен для реализации некоторых полей, например, таких как содержание.

```
class Field(Block)
```

4.44 Класс Fill

Класс описывает свойства заполнения фигуры, ячейки и т.д.

```
class Fill(object)
```

Список методов:

```
def getColor()
def getUrl()
def isNoFill()
```

4.44.1 Fill.getColor

Метод возвращает цвет заполнения Color.

```
Fill.getColor()
```

4.44.2 Fill.getUrl

Метод возвращает путь к изображению, которое используется в качестве заполнения.

```
Fill.getUrl()
```

4.44.3 Fill.isNoFill

Метод возвращает свойство заполнения.

```
Fill.isNoFill()
```

4.45 Класс Shape

Класс Shape описывает фигуру, содержит методы для установки и получения ее свойств ShapeProperties.

```
class Shape(Block)
```

Список методов:

```
def getShapeProperties()
def setShapeProperties(properties)
```

4.45.1 Shape.getShapeProperties

Метод возвращает свойства фигуры ShapeProperties.

```
Shape.getShapeProperties()
```

4.45.2 Shape.setShapeProperties

Метод устанавливает свойства фигуры ShapeProperties.

```
Shape.setShapeProperties(properties)
```

4.46 Класс ShapeProperties

Класс описывает свойства фигуры.

```
class ShapeProperties(object)
```

Список свойств:

- `verticalAlignment` – вертикальное выравнивание `VerticalAlignment`;
- `borderProperties` - свойства границ фигуры `LineProperties`;
- `fill` – свойства заполнения фигуры `Fill`;
- `shapeTextLayout` - свойства текста внутри фигуры `ShapeTextLayout`.

4.47 Тип ShapeTextLayout

Тип `ShapeTextLayout` описывает свойства текста, находящегося внутри фигуры. (см.Таблица 37).

Таблица 37 - Свойства текста фигуры

Имя константы свойства текста фигуры	Описание
ShapeTextLayout_DoNotAutoFit	Размещение текста в фигуре по умолчанию.
ShapeTextLayout_FitShapeExtentToText	Расширение фигуры под текст.
ShapeTextLayout_FitTextToShape	Заполнение фигуры текстом.

4.48 Класс Paragraph

В объектной модели документа представляет отдельный абзац и является наследником класса `Block`.

```
class Paragraph(Block)
```

Список методов:

```
def getParagraphProperties()
def setParagraphProperties(properties)
def getListSchema()
def setListSchema(schema)
def getListLevel()
def setListLevel(level)
def increaseListLevel()
def decreaseListLevel()
```

4.48.1 Paragraph.getParagraphProperties

Позволяет получать свойства абзаца `ParagraphProperties`.

```
Paragraph.getParagraphProperties()
```

4.48.2 Paragraph.setParagraphProperties

Позволяет установить свойства абзаца `ParagraphProperties`.

```
Paragraph.setParagraphProperties(properties)
```

4.48.3 Paragraph.getListSchema

Возвращает схему маркированного или нумерованного списка `ListSchema`.

```
Paragraph.getListSchema()
```

4.48.4 Paragraph.setListSchema

Устанавливает схему `ListSchema` маркированного или нумерованного списка.

```
Paragraph.setListSchema(schema)
```

4.48.5 Paragraph.getListLevel

Возвращает глубину вложенности элемента списка.

```
Paragraph.getListLevel()
```

4.48.6 Paragraph.setListLevel

Устанавливает глубину вложенности элемента списка.

```
Paragraph.setListLevel(level)
```

4.48.7 Paragraph.increaseListLevel

Увеличивает уровень списка на единицу. В случае, если максимальный уровень уже установлен, увеличения не происходит.

```
Paragraph.increaseListLevel()
```

4.48.8 Paragraph.decreaseListLevel

Уменьшает уровень списка на единицу. В случае, если минимальный уровень уже установлен, уменьшения не происходит.

```
Paragraph.decreaseListLevel()
```

4.49 Класс Paragraphs

Класс `Paragraphs` предоставляет доступ к коллекции абзацев.

```
class Paragraphs(object)
```

Список методов:

```
def setListSchema(schema)
def setListLevel(level)
def getEnumerator()
```

4.49.1 Paragraphs.setListSchema

Устанавливает тип маркированного или нумерованного списка `ListSchema`.

```
Paragraphs.setListSchema(schema)
```

4.49.2 Paragraphs.setListLevel

Устанавливает глубину вложенности элемента списка.

```
Paragraphs.setListLevel(level)
```

4.49.3 Paragraphs.getEnumerator

Возвращает коллекцию классов `Paragraph`.

```
Paragraphs.getEnumerator()
```

4.50 Класс Range

Представляет непрерывную область в документе, определяемую начальной и конечной позицией.

```
class Range(object)
```

Список методов:

```
def getBegin()
def getEnd()
def extractText()
def removeContent()
def lockContent()
def unlockContent()
def isContentLocked()
def replaceText(text)
def getTextProperties()
def setTextProperties(textProperties)
def getBlocksEnumerator()
def getTrackedChangesEnumerator()
def getComments()
def getParagraphs()
def getImages()
```

4.50.1 Range.getBegin

Возвращает начальную позицию диапазона `Position`.

```
Range.getBegin()
```

4.50.2 Range.getEnd

Возвращает конечную позицию диапазона `Position`.

```
Range.getEnd()
```

4.50.3 Range.extractText

Возвращает текстовое представление содержимого в диапазоне. При этом часть содержимого (например, фотографии) будет пропущена, часть (например, таблицы) будет преобразована.

```
Range.extractText()
```

4.50.4 Range.removeContent

Удаляет содержимое диапазона.

```
Range.removeContent()
```

4.50.5 Range.lockContent

Запрещает изменения содержимого диапазона.

```
Range.lockContent()
```

4.50.6 Range.unlockContent

Разрешает изменения содержимого диапазона.

```
Range.unlockContent()
```

4.50.7 Range.isContentLocked

Возвращает `true`, если изменения содержимого диапазона запрещены.

```
Range.isContentLocked()
```

4.50.8 Range.replaceText

Заменяет содержимое диапазона на заданный текст.

```
Range.replaceText(text)
```

4.50.9 Range.getTextProperties

Возвращает свойства `TextProperties` оформления текста в диапазоне.

```
Range.getTextProperties()
```

4.50.10 Range.setTextProperties

Настройка свойств оформления текста `TextProperties`.

```
Range.setTextProperties(textProperties)
```

4.50.11 Range.getBlocksEnumerator

Возвращает коллекцию объектов Block в диапазоне.

```
Range.getBlocksEnumerator()
```

4.50.12 Range.getTrackedChangesEnumerator

Возвращает коллекцию отслеживаемых изменений в диапазоне.

```
Range.getTrackedChangesEnumerator()
```

4.50.13 Range.getComments

Обеспечивает доступ к комментариям Comments в диапазоне. Комментарии, примененные к одному и тому же диапазону, упорядочиваются по датам, если таковые имеются. Если дат нет, то порядок комментариев не определен.

```
Range.getComments()
```

4.50.14 Range.getParagraphs

Обеспечивает доступ к абзацам Paragraphs в диапазоне.

```
Range.getParagraphs()
```

4.50.15 Range.getImages

Обеспечивает доступ к изображениям Images в диапазоне.

```
Range.getImages()
```

4.51 Класс Section

Предоставляет методы для управления отдельными разделами (sections) в документе.

Объявление класса:

```
class Section(object)
```

Список методов:

```
def setPageProperties(pageProperties)
def getPageProperties()
def setPageOrientation(pageOrientation)
def getPageOrientation()
def getRange()
def getHeaders()
def getFooters()
```

4.51.1 Section.setPageProperties

Устанавливает параметры страниц PageProperties.

```
Section.setPageProperties(pageProperties)
```

4.51.2 Section.getPageProperties

Возвращает параметры страниц PageProperties.

```
Section.getPageProperties()
```

4.51.3 Section.setPageOrientation

Задает ориентацию страниц раздела PageOrientation.

```
Section.setPageOrientation(pageOrientation)
```

4.51.4 Section.getPageOrientation

Возвращает ориентацию страниц раздела PageOrientation.

```
Section.getPageOrientation()
```

4.51.5 Section.getRange

Возвращает диапазон (Range) в документе, соответствующий данному разделу.

```
Section.getRange()
```

4.51.6 Section.getHeaders

Предоставляет доступ к коллекции верхних колонтитулов (HeadersFooters), содержащихся в разделе.

```
Section.getHeaders()
```

4.51.7 Section.getFooters

Предоставляет доступ к коллекции HeadersFooters нижних колонтитулов, содержащихся в разделе.

```
Section.getFooters()
```

4.52 Класс Search

Предоставляет интерфейс для выполнения поиска в документе.

```
class Search(object)
```

Список методов:

```
def findText(text)
def findText(text, range)
```

4.52.1 Search.findText

Обеспечивает поиск текста в документе без учета регистра. Возвращает экземпляр класса `Range` – диапазон, содержащий искомый фрагмент.

```
Search.findText(text)
Search.findText(text, range)
```

Параметры:

Параметр метода может содержать только искомый фрагмент текста, в этом случае поиск осуществляется во всем документе.

Дополнительно, в параметрах может быть указан диапазон, в этом случае поиск производится в указанном диапазоне документа.

4.53 Глобальный метод createSearch

Создает новый объект `Search` в указанном экземпляре объекта `document`.

```
createSearch(document)
```

4.54 Класс CellPosition

Представляет положение ячейки в таблице. Нумерация позиции начинается с нуля, так что значение 0,0 соответствует верхней левой ячейке таблицы для редактора текста, либо ячейке A1 для редактора таблиц.

```
class CellPosition(object)
```

Список свойств:

- `row` – номер строки ячейки, нумерация начинается с нуля;
- `column` – номер столбца ячейки, нумерация начинается с нуля.

Список методов:

```
def toString()
```

4.54.1 CellPosition.toString

Возвращает информацию о положении ячейки в виде строкового значения формата `(row: <value>, column: <value>)`.

```
CellPosition.toString()
```

4.55 Класс CellRangePosition

Представляет положение диапазона ячеек в таблице. По умолчанию диапазон включает одну ячейку в позиции 0,0 что соответствует верхней левой ячейке таблицы для редактора текста, либо ячейке A1 для редактора таблиц.

```
class CellRangePosition(object)
```

Список свойств:

- `topLeft` – позиция левой верхней ячейки таблицы прямоугольного диапазона. Нумерация позиции начинается с нуля, так что значение 0,0 соответствует верхней левой ячейке таблицы для редактора текста, либо ячейке A1 для редактора таблиц;
- `bottomRight` – позиция правой нижней ячейки таблицы прямоугольного диапазона.

Список методов:

```
def toString()
```

4.55.1 CellRangePosition. toString

Возвращает информацию о диапазоне ячеек в виде строкового значения формата `(topLeft: <value>, bottomRight: <value>)`.

```
CellRangePosition.toString()
```

4.56 Класс LineSpacing

Позволяет управлять межстрочным интервалом.

```
class LineSpacing(object)
```

Список методов:

```
def __eq__(other)
def __ne__(other)
```

Список свойств:

- `value` – значение межстрочного интервала;
- `rule` – тип межстрочного интервала `LineSpacingRule`.

4.56.1 LineSpacing.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `LineSpacing`.


```
LineSpacing.__eq__(other)
```

4.56.2 LineSpacing.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `LineSpacing`.

```
LineSpacing.__ne__(other)
```

4.57 Класс LineProperties

Содержит свойства оформления линий.

```
class LineProperties(object)
```

Список методов:

```
def __eq__(other)
def __ne__(other)
```

Список свойств:

- `style` – стиль линии `LineStyle`;
- `width` – ширина линии;
- `color` – цвет линии `Color`;
- `headLineEndingProperties` – оформление начала линии `LineEndingProperties`;
- `tailLineEndingProperties` – оформление конца линии `LineEndingProperties`.

4.57.1 LineProperties.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `LineProperties`.

```
LineProperties.__eq__(other)
```

4.57.2 LineProperties.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `LineProperties`.

```
LineProperties.__ne__(other)
```

4.58 Класс LineEndingProperties

Содержит свойства оформления окончаний линий.

```
class LineEndingProperties(object)
```

Список методов:

```
def __eq__(other)
def __ne__(other)
```

Список свойств:

- `style` – стиль окончания линии `LineEndingStyle`;
- `relativeExtent` – размер окончания линии относительно ее ширины.

4.58.1 LineEndingProperties.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `LineEndingProperties`.

```
LineEndingProperties.__eq__(other)
```

4.58.2 LineEndingProperties.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `LineEndingProperties`.

```
LineEndingProperties.__ne__(other)
```

4.59 Класс ParagraphProperties

Содержит свойства абзаца.

```
class ParagraphProperties(object)
```

Список методов:

```
def __eq__(other)
def __ne__(other)
```

Список свойств:

- `beforeSpacing` – интервал перед абзацем;
- `afterSpacing` – интервал после абзаца;
- `lineSpacing` – межстрочный интервал;
- `alignment` – выравнивание абзаца;
- `firstLineIndent` – отступ первой строки;

- `leftIndent` – отступ слева;
- `rightIndent` – отступ справа.

4.59.1 ParagraphProperties.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `ParagraphProperties`.

```
ParagraphProperties.__eq__(other)
```

4.59.2 ParagraphProperties.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `ParagraphProperties`.

```
ParagraphProperties.__ne__(other)
```

4.60 Класс Borders

Предоставляет методы для управления границами отдельной ячейки.

```
class Borders(object)
```

Список методов:

```
def getLeft()
def getRight()
def getTop()
def getBottom()
def getDiagonalDown()
def getDiagonalUp()
def getInnerHorizontal()
def getInnerVertical()
def setLeft(lineProperties)
def setRight(lineProperties)
def setTop(lineProperties)
def setBottom(lineProperties)
def setDiagonalDown(lineProperties)
def setDiagonalUp(lineProperties)
def setOuter(lineProperties)
def setDiagonals(lineProperties)
def setInnerHorizontal(lineProperties)
def setInnerVertical(lineProperties)
def setInner(lineProperties)
def setAll(lineProperties)
```

4.60.1 Borders.getLeft

Позволяет получить свойства `LineProperties` левой границы ячейки.

```
Borders.getLeft()
```

4.60.2 Borders.getRight

Позволяет получить настройки свойств LineProperties правой границы ячейки.

```
Borders.getRight()
```

4.60.3 Borders.getTop

Позволяет получить настройки свойств LineProperties верхней границы ячейки.

```
Borders.getTop()
```

4.60.4 Borders.getBottom

Позволяет получить настройки свойств LineProperties нижней границы ячейки.

```
Borders.getBottom()
```

4.60.5 Borders.getDiagonalDown

Позволяет получить настройки свойств LineProperties диагональной границы ячейки.

```
Borders.getDiagonalDown()
```

4.60.6 Borders.getDiagonalUp

Позволяет получить настройки свойств LineProperties диагональной границы ячейки.

```
Borders.getDiagonalUp()
```

4.60.7 Borders.getInnerHorizontal

Позволяет получить свойства LineProperties горизонтальных внутренних границ диапазона ячеек.

```
Borders.getInnerHorizontal()
```

4.60.8 Borders.getInnerVertical

Позволяет получить свойства LineProperties вертикальных внутренних границ диапазона ячеек.

```
Borders.getInnerVertical()
```

4.60.9 Borders.setLeft

Позволяет установить настройки свойств LineProperties левой границы ячейки.

```
Borders.setLeft(lineProperties)
```

4.60.10 Borders.setRight

Позволяет установить настройки свойств LineProperties правой границы ячейки.

```
Borders.setRight(lineProperties)
```

4.60.11 Borders.setTop

Позволяет установить настройки свойств LineProperties верхней границы ячейки.

```
Borders.setTop(lineProperties)
```

4.60.12 Borders.setBottom

Позволяет установить настройки свойств LineProperties нижней границы ячейки.

```
Borders.setBottom(lineProperties)
```

4.60.13 Borders.setDiagonalDown

Позволяет установить настройки свойств LineProperties диагональной границы ячейки.

```
Borders.setDiagonalDown(lineProperties)
```

4.60.14 Borders.setDiagonalUp

Позволяет установить настройки свойств LineProperties диагональной границы ячейки.

```
Borders.setDiagonalUp(lineProperties)
```

4.60.15 Borders.setOuter

Позволяет установить настройки свойств LineProperties одновременно левой, правой, верхней и нижней границы ячейки.

```
Borders.setOuter(lineProperties)
```

4.60.16 Borders.setDiagonals

Позволяет установить настройки LineProperties свойств диагональных границ ячейки.

```
Borders.setDiagonals(lineProperties)
```

4.60.17 Borders.setInnerHorizontal

Позволяет установить свойства LineProperties внутренних горизонтальных границ диапазона ячеек.

```
Borders.setInnerHorizontal(lineProperties)
```

4.60.18 Borders.setInnerVertical

Позволяет установить свойства LineProperties внутренних вертикальных границ диапазона ячеек.

```
Borders.setInnerVertical(lineProperties)
```

4.60.19 Borders.setInner

Позволяет установить свойства LineProperties внутренних границ диапазона ячеек.

```
Borders.setInner(lineProperties)
```

4.60.20 Borders.setAll

Позволяет установить свойства LineProperties всех границ диапазона ячеек.

```
Borders.setAll(lineProperties)
```

4.61 Класс RangeBorders

Класс RangeBorders оставлен для совместимости. Вместо него необходимо использовать класс Borders.

```
class RangeBorders(Borders)
```

4.62 Класс Cell

Представляет отдельную ячейку на листе электронной таблицы, либо ячейку таблицы в составе текстового документа.

```
class Cell(object)
```

Список методов:

```
def getRange()
def getFormat()
def setFormat(format)
def setFormat(numberCellFormat)
def setFormat(percentageCellFormat)
def setFormat(currencyCellFormat)
def setFormat(accountingCellFormat)
def setFormat(dateTimeCellFormat)
def setFormat(fractionCellFormat)
def setFormat(scientificCellFormat)
def getCustomFormat()
def setCustomFormat(formatString)
def setBool(value)
def setNumber(value)
def setText(value)
```

```
def setFormula(value)
def getFormulaAsString()
def getFormattedValue()
def getRawValue()
def setFormattedValue(value)
def setContent(value)
def getCellProperties()
def setCellProperties(cellProperties)
def getBorders()
def setBorders(borders)
def getParagraphProperties()
def setParagraphProperties(ParagraphProperties)
def isPivotTableRoot()
def getPivotTable()
def unmerge()
```

4.62.1 Cell.getRange

Возвращает диапазон Range, позволяющий работать с содержимым ячейки как с абзацем (Paragraph) текста.

```
Cell.getRange()
```

4.62.2 Cell.getFormat

Возвращает тип данных ячейки CellFormat.

```
Cell.getFormat()
```

4.62.3 Cell.setFormat

Устанавливает тип данных ячейки: CellFormat, NumberCellFormatting, PercentageCellFormatting, CurrencyCellFormatting, AccountingCellFormatting, DateTimeCellFormatting, FractionCellFormatting, ScientificCellFormatting.

```
Cell.setFormat(format)
Cell.setFormat(numberCellFormat)
Cell.setFormat(percentageCellFormat)
Cell.setFormat(currencyCellFormat)
Cell.setFormat(accountingCellFormat)
Cell.setFormat(dateTimeCellFormat)
Cell.setFormat(fractionCellFormat)
Cell.setFormat(scientificCellFormat)
```

4.62.4 Cell.getCustomFormat

Возвращает строку формата ячейки.

```
Cell.getCustomFormat()
```

4.62.5 Cell.setCustomFormat

Устанавливает формат ячейки в соответствии с значением параметра `formatString`.

```
Cell.setCustomFormat(formatString)
```

4.62.6 Cell.setBool

Устанавливает значение типа `bool` для ячейки.

```
Cell.setBool(value)
```

4.62.7 Cell.setNumber

Устанавливает числовое значение ячейки, указанное в параметре `value`.

```
Cell.setNumber(value)
```

4.62.8 Cell.setText

Устанавливает текстовое значение ячейки.

```
Cell.setText(value)
```

4.62.9 Cell.setFormula

Используется для ввода формулы в ячейку.

```
Cell.setFormula(value)
```

Формула – это любое выражение в ячейке, которое начинается со знака равенства (=). Формулы могут содержать функции, значения, адреса ячеек, имена, операторы действий и др.

Функция – это предустановленная формула приложения МойОфис Таблица, для вычисления которой необходимо использовать аргументы. Полный список функций приведен в Приложении 1 «Перечень функций и их описание» в документе «МойОфис Таблица. Руководство пользователя».

Основные принципы ввода формул и функций:

- формула всегда начинается со знака равенства (=);
- после знака равенства могут следовать функции, константы, адреса ячеек, операторы действий и другие элементы;
- все открывающие и закрывающие скобки должны быть согласованы;
- обязательные аргументы используемых функций должны быть указаны;
- константы не должны содержать символ «$».

4.62.10 Cell.getFormulaAsString

Позволяет получить текст формулы в данной ячейке. Формула – это любое выражение в ячейке, которое начинается со знака равенства (=).

```
Cell.getFormulaAsString()
```

4.62.11 Cell.getFormattedValue

Возвращает значение ячейки, в виде строки, форматированной в соответствии с требованиями типа формата ячейки `CellFormat`.

```
Cell.getFormattedValue()
```

4.62.12 Cell.getRawValue

Возвращает значение ячейки в формате «Общий» (без форматирования), см. `CellFormat`.

```
Cell.getRawValue()
```

4.62.13 Cell.setFormattedValue

Используется для установки значения ячейки, заранее форматированного в соответствии с требованиями типа ячейки. При невозможности сопоставить форматирование определенному типу данных, используется форматирование для типа данных «Текстовый».

```
Cell.setFormattedValue(value)
```

4.62.14 Cell.setContent

Используется для установки значения ячейки и ее типа, с использованием значения и типа параметра `value`. При невозможности сопоставить форматирование определенному типу данных, используется форматирование для типа данных «Текстовый».

```
Cell.setContent(value)
```

4.62.15 Cell.getCellProperties

Используется для получения оформления ячейки `CellProperties` (например, цвет фона, вертикальное выравнивание и т. д).

```
Cell.getCellProperties()
```

4.62.16 Cell.setCellProperties

Устанавливает свойства элементов оформления ячейки `CellProperties` (например, цвет фона, вертикальное выравнивание и т. д.). Неустановленные свойства, которые имеют значение `none` не будут изменены.

```
Cell.setCellProperties(cellProperties)
```

4.62.17 Cell.getBorders

Возвращает границы `Borders` отдельной ячейки.

```
Cell.getBorders()
```

4.62.18 Cell.setBorders

Устанавливает границы ячейки `Borders`.

```
Cell.setBorders(borders)
```

4.62.19 Cell.getParagraphProperties

Возвращает свойства абзаца `ParagraphProperties`, содержащего в ячейке.

```
Cell.getParagraphProperties()
```

4.62.20 Cell.setParagraphProperties

Метод `setParagraphProperties` предназначен для управления свойствами абзаца `ParagraphProperties`, такими как горизонтальное выравнивание текста, межстрочный интервал, межсимвольный интервал. В сочетании с методом `setCellProperties` достигается возможность управления всеми настройками ячейки и ее содержимого.

```
Cell.setParagraphProperties(ParagraphProperties)
```

4.62.21 Cell.unmerge

Разъединяет несколько ячеек, которые были объединены ранее. Допустимо разъединение только тех ячеек, которые были объединены ранее. После завершения операции данные, содержавшиеся в объединенной ячейке, будут помещены в верхнюю левую ячейку диапазона.

```
Cell.unmerge()
```

4.62.22 Cell.isPivotTableRoot

Возвращает `true`, если ячейчка является корнем сводной таблицы.

```
Cell.isPivotTableRoot()
```

4.62.23 Cell.getPivotTable

Возвращает сводную таблицу, если текущая ячейка входит в ее состав.

```
Cell.getPivotTable()
```

4.63 Класс CellProperties

Содержит свойства оформления ячейки.

```
class CellProperties(object)
```

Список методов:

```
def __eq__(other)
def __ne__(other)
```

Список свойств:

- `verticalAlignment` – вертикальное выравнивание ячейки `VerticalAlignment`;
- `backgroundColor` – цвет фона ячейки `Color`;
- `textLayout` – свойства форматирования текста в ячейке `TextLayout`;
- `textOrientation` – ориентация текста в ячейке `TextOrientation`.

4.63.1 CellProperties.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `CellProperties`.

```
CellProperties.__eq__(other)
```

4.63.2 CellProperties.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `CellProperties`.

```
CellProperties.__ne__(other)
```

4.64 Класс CellRange

Представляет диапазон (коллекцию) ячеек.

```
class CellRange(object)
```

Список методов:

```
def getBeginRow()
def getBeginColumn()
def getLastRow()
def getLastColumn()
def getTable()
```

```
def getEnumerator()
def setBorders(borders)
def setCellProperties(cellProperties)
def getCellProperties()
def merge()
def autoFill(destination)
```

4.64.1 CellRange.getBeginRow

Возвращает индекс начальной строки диапазона.

```
CellRange.getBeginRow()
```

4.64.2 CellRange.getBeginColumn

Возвращает индекс начального столбца диапазона.

```
CellRange.getBeginColumn()
```

4.64.3 CellRange.getLastRow

Возвращает индекс последней строки диапазона.

```
CellRange.getLastRow()
```

4.64.4 CellRange.getLastColumn

Возвращает индекс последнего столбца диапазона.

```
CellRange.getLastColumn()
```

4.64.5 CellRange.getTable

Возвращает таблицу `Table`, содержащую текущий диапазон.

```
CellRange.getTable()
```

4.64.6 CellRange.getEnumerator

Возвращает коллекцию ячеек диапазона.

```
CellRange.getEnumerator()
```

4.64.7 CellRange.setBorders

Устанавливает границы ячеек `Borders` в диапазоне.

```
CellRange.setBorders(borders)
```

4.64.8 CellRange.setCellProperties

Устанавливает свойства `CellProperties` ячеек диапазона.

```
CellRange.setCellProperties(cellProperties)
```

4.64.9 CellRange.getCellProperties

Позволяет получить свойства ячеек диапазона. Возвращает структуру `CellProperties` с одинаковыми свойствами для всех ячеек диапазона.

```
CellRange.getCellProperties()
```

4.64.10 CellRange.merge

Служит для объединения ячеек в диапазоне.

```
CellRange.merge()
```

4.64.11 CellRange.autoFill

Метод заполняет диапазон ячеек, используя данные из этого диапазона в качестве источника. Конечный диапазон вычисляется из начальной позиции исходного диапазона и последней позиции (аргумент `destination`). Таким образом, конечный диапазон всегда полностью содержит исходный диапазон.

Метод `autoFill` автоматически интерполирует исходные точки и находит алгоритм аппроксимации, который используется для экстраполяции значений в диапазоне ячеек назначения. Результат выполнения метода в текстовом редакторе может отличаться от табличного редактора из-за разных типов данных в ячейках.

Метод возвращает `true`, если ячейки успешно заполнены, и `false` в других случаях (например, если диапазон ячеек назначения содержит формулу или сводную таблицу и т. д.).

Метод вызывает `OutOfRangeError`, если исходный или целевой диапазоны находятся за пределами таблицы.

```
CellRange.autoFill(destination)
```

4.65 Класс Script

Предоставляет интерфейс для доступа к отдельным макрокомандам документа.

```
class Script(object)
```

Список методов:

```
def getName()
def setName(name)
def getBody()
def setBody(body)
```

4.65.1 Script.getName

Возвращает название макрокоманды.

```
Script.getName()
```

4.65.2 Script.setName

Устанавливает название макрокоманды.

```
Script.setName(name)
```

4.65.3 Script.getBody

Возвращает текст исходного кода макрокоманды на языке программирования Lua.

```
Script.getBody()
```

4.65.4 Script.setBody

Устанавливает текст исходного кода макрокоманды на языке программирования Lua.

```
Script.setBody(body)
```

4.66 Класс Scripts

Предоставляет интерфейс для доступа к коллекции макрокоманд документа.

```
class Scripts(object)
```

Список методов:

```
def getScript(name)
def setScript(name, script)
def removeScript(name)
def getEnumerator()
```

4.66.1 Scripts.getScript

Возвращает ссылку на макрокоманду с наименованием `name`.

```
Scripts.getScript(name)
```

4.66.2 Scripts.setScript

Добавляет в коллекцию новую макрокоманду с текстом `script` и названием `name`. Если макрокоманда с таким названием уже существует, то ее текст будет заменен.

```
Scripts.setScript(name, script)
```

4.66.3 Scripts.removeScript

Удаляет макрокоманду с наименованием `name`.

```
Scripts.removeScript(name)
```

4.66.4 Scripts.getEnumerator

Возвращает коллекцию макрокоманд.

```
Scripts.getEnumerator()
```

4.67 Класс TrackedChange

Класс TrackedChange предназначен для отслеживания изменений в документе.

```
class TrackedChange(object)
```

Список методов:

```
def getRange()
def getType()
def getInfo()
```

4.67.1 TrackedChange.getRange

Возвращает объект Range, который соответствует измененному диапазону внутри абзаца.

```
TrackedChange.getRange()
```

4.67.2 TrackedChange.getType

Позволяет получить информацию о типе TrackedChangeType отслеживаемого изменения. Если отслеживаемое изменение добавлено, то возвращает ноль, если удалено, то единицу.

```
TrackedChange.getType()
```

4.67.3 TrackedChange.getInfo

Позволяет получить информацию TrackedChangeInfo, содержащую сведения об отслеживаемых изменениях.

```
TrackedChange.getInfo()
```

4.68 Класс TrackedChangeInfo

Содержит информацию об авторе и дате изменения.

```
class TrackedChangeInfo(object)
```

Список свойств:

- `author` – автор изменений;
- `timeStamp` – дата и время создания изменений.

Список методов:

```
def __eq__(other)
def __ne__(other)
```

4.68.1 TrackedChangeInfo.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `TrackedChangeInfo`.

```
TrackedChangeInfo.__eq__(other)
```

4.68.2 TrackedChangeInfo.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `TrackedChangeInfo`.

```
TrackedChangeInfo.__ne__(other)
```

4.69 Класс Table

Предоставляет доступ к листу электронной таблицы или отдельной таблице в составе текстового документа.

```
class Table(Block)
```

Список методов:

```
def getName()
def setName(newName)
def getRowsCount()
def getColumnsCount()
def insertColumnAfter(columnIndex, copyColumnStyle=True, columnsCount=1)
def insertColumnBefore(columnIndex, copyColumnStyle=True, columnsCount=1)
def insertRowAfter(rowIndex, copyRowStyle=True, rowsCount=1)
def insertRowBefore(rowIndex, copyRowStyle=True, rowsCount=1)
def removeColumn(columnIndex, columnsCount=1)
def removeRow(rowIndex, rowsCount=1)
def setColumnWidth(columnIndex, width)
def setRowHeight(rowIndex, height, heightRule)
def getCellRange(range)
def getCell(position)
def duplicate()
def moveTo(index)
def groupRows(first, rowsCount)
def ungroupRows(first, rowsCount)
def clearRowGroups(first, rowsCount)
def groupColumns(first, columnsCount)
def ungroupColumns(first, columnsCount)
def clearColumnGroups(first, columnsCount)
def setColumnsVisible(first, columnsCount, visible)
def setRowsVisible(first, rowsCount, visible)
def setVisible(visible)
def isVisible()
```

```
def setPrintArea(range)
def setPrintAreas(range)
def getPrintAreas()
def getCharts()
def getNamedExpressions()
def __eq__(other)
def __ne__(other)
```

4.69.1 Table.getName

Позволяет получить наименование листа табличного документа.

```
Table.getName()
```

4.69.2 Table.setName

Позволяет установить значение `newName` в качестве наименования листа табличного документа.

```
Table.setName(newName)
```

4.69.3 Table.getRowsCount

Позволяет получить количество строк на листе табличного документа или для отдельной таблицы в составе текстового документа.

```
Table.getRowsCount()
```

4.69.4 Table.getColumnsCount

Позволяет получить количество столбцов на листе табличного документа или для отдельной таблицы в составе текстового документа.

```
Table.getColumnsCount()
```

4.69.5 Table.insertColumnAfter

Позволяет добавить в таблицу новые столбцы в количестве `columnsCount` после столбца с индексом `columnIndex`. Индексация столбцов начинается с нуля.

```
Table.insertColumnAfter(columnIndex, copyColumnStyle=True, columnsCount=1)
```

Параметры:

- `columnIndex` – индекс столбца после которого добавляются новые столбцы;
- `copyColumnStyle` – параметр для указания, следует ли использовать для новых столбцов элементы оформления столбца `columnIndex`, по умолчанию имеет значение `true`;
- `columnsCount` – количество добавляемых столбцов, по умолчанию равно 1.

4.69.6 Table.insertColumnBefore

Позволяет добавить в таблицу новые столбцы в количестве `columnsCount` до столбца с индексом `columnIndex`. Индексация столбцов начинается с нуля.

```
Table.insertColumnBefore(columnIndex, copyColumnStyle=True, columnsCount=1)
```

Параметры:

- `columnIndex` – индекс столбца до которого добавляются новые столбцы;
- `copyColumnStyle` – параметр для указания, следует ли использовать для новых столбцов элементы оформления столбца `columnIndex`, по умолчанию имеет значение `true`;
- `columnsCount` – количество добавляемых столбцов, по умолчанию равно 1.

4.69.7 Table.insertRowAfter

Позволяет добавить в таблицу новые строки в количестве `rowsCount` после строки с индексом `rowIndex`. Индексация строк начинается с нуля.

```
Table.insertRowAfter(rowIndex, copyRowStyle=True, rowsCount=1)
```

Параметры:

- `rowIndex` – индекс строки после которой добавляются новые строки;
- `copyRowStyle` – параметр для указания, следует ли использовать для новых строк элементы оформления строки `rowIndex`, по умолчанию равен `true`;
- `rowsCount` – количество добавляемых строк, по умолчанию равно 1.

4.69.8 Table.insertRowBefore

Позволяет добавить в таблицу новые строки в количестве `rowsCount` до строки с индексом `rowIndex`. Индексация строк начинается с нуля.

```
Table.insertRowBefore(rowIndex, copyRowStyle=True, rowsCount=1)
```

Параметры:

- `rowIndex` – индекс строки до которой добавляются новые строки;
- `copyRowStyle` – параметр для указания, следует ли использовать для новых строк элементы оформления строки `rowIndex`, по умолчанию равен `true`;
- `rowsCount` – количество добавляемых строк, по умолчанию равно 1.

4.69.9 Table.removeColumn

Позволяет удалить столбцы в количестве `columnsCount`, начиная с индекса `columnIndex`. Индексация столбцов начинается с нуля.

```
Table.removeColumn(columnIndex, columnsCount=1)
```

4.69.10 Table.removeRow

Позволяет удалить в таблице строки в количестве `rowsCount`, начиная с индекса `rowIndex`. Индексация строк начинается с нуля.

```
Table.removeRow(rowIndex, rowsCount=1)
```

4.69.11 Table.setColumnWidth

Позволяет установить значение `width` в качестве ширины столбца с индексом `columnIndex`. Метод вызывает исключение `OutOfRangeError`, если параметр `columnIndex` выходит за границы таблицы. Индексация столбцов начинается с нуля.

```
Table.setColumnWidth(columnIndex, width)
```

4.69.12 Table.setRowHeight

Метод `setRowHeight` позволяет установить значение `height` в качестве высоты строки с индексом `rowIndex`. Индексация строк начинается с нуля.

Метод вызывает исключение `OutOfRangeError`, если параметр `rowIndex` выходит за границы таблицы.

Тип `RowHeightRule` задает точность значения, где - `RowHeightRule_Exact` точно, а `RowHeightRule_AtLeast` – не меньше.

```
Table.setRowHeight(rowIndex, height, heightRule)
```

4.69.13 Table.getCellRange

Позволяет получить доступ к диапазону ячеек `CellRange` таблицы по адресу или положению, указанному в параметре.

```
Table.getCellRange(range)
```

Параметры:

В качестве параметра может использоваться:

- строковое значение для указания диапазона ячеек, например, «A1:C4»;
- экземпляр класса `CellRangePosition`, который позволяет указать строки и столбцы по индексу. Например, диапазон «A1:C4» задается как `CellRangePosition(0,0,2,3)`.

4.69.14 Table.getCell

Позволяет получить доступ к ячейкам `Cell` таблицы по адресу или положению в таблице.

```
Table.getCell(positions)
```

Параметры:

В качестве параметра могут использоваться:

- строковое значение для указания адреса ячейки, когда столбцы задаются буквами, а строки – числами, например, «A1»;
- экземпляр класса CellPosition, который позволяет указать индекс положения ячейки. Например, положение ячейки «A1» задается как `CellPosition(0,0)`.

4.69.15 Table.duplicate

Позволяет создать копию листа в табличном документе. Созданная копия листа размещается после копируемого листа. Метод может быть использован только в табличном документе.

```
Table.duplicate()
```

4.69.16 Table.moveTo

Позволяет переместить лист таблицы по указанному индексу. Индекс должен быть меньше или равен количеству листов в документе. Метод может быть использован только в табличном документе.

```
Table.moveTo(index)
```

4.69.17 Table.groupRows

Метод `groupRows` позволяет группировать строки в количестве `rowsCount`, начиная с индекса `first`. Индексация строк начинается с нуля.

```
Table.groupRows(first, rowsCount)
```

4.69.18 Table.ungroupRows

Метод `ungroupRows` позволяет разгруппировать строки в количестве `rowsCount`, начиная с индекса `first`. Индексация строк начинается с нуля.

```
Table.ungroupRows(first, rowsCount)
```

4.69.19 Table.clearRowGroups

Метод `clearRowGroups` позволяет очистить группы для строк в количестве `rowsCount`, начиная с индекса `first`. Индексация строк начинается с нуля.

```
Table.clearRowGroups(first, rowsCount)
```

4.69.20 Table.groupColumns

Метод `groupColumns` позволяет группировать столбцы в количестве `columnsCount`, начиная с индекса `first`. Индексация столбцов начинается с нуля.

```
Table.groupColumns(first, columnsCount)
```

4.69.21 Table.ungroupColumns

Метод `ungroupColumns` позволяет разгруппировать столбцы в количестве `columnsCount`, начиная с индекса `first`. Индексация столбцов начинается с нуля.

```
Table.ungroupColumns(first, columnsCount)
```

4.69.22 Table.clearColumnGroups

Метод `clearColumnGroups` позволяет очистить группы для столбцов в количестве `columnsCount`, начиная с индекса `first`. Индексация столбцов начинается с нуля.

```
Table.clearColumnGroups(first, columnsCount)
```

4.69.23 Table.setColumnsVisible

Метод `setColumnsVisible` позволяет задавать видимость столбцов в количестве `columnsCount`, начиная с индекса `first`. Индексация столбцов начинается с нуля.

```
Table.setColumnsVisible(first, columnsCount, visible)
```

4.69.24 Table.setRowsVisible

Метод `setRowsVisible` позволяет задавать видимость строк в количестве `rowsCount`, начиная с индекса `first`. Индексация строк начинается с нуля.

```
Table.setRowsVisible(first, rowsCount, visible)
```

4.69.25 Table.setVisible

Управляет видимостью листа таблицы в табличном документе.

```
Table.setVisible(visible)
```

4.69.26 Table.isVisible

Возвращает `true`, если лист таблицы в табличном документе отображается в редакторе таблиц.

```
Table.isVisible()
```

4.69.27 Table.getPrintAreas

Метод getPrintAreas возвращает коллекцию текущих областей печати `CellRangePositions`.

```
Table.getPrintAreas()
```

4.69.28 Table.setPrintAreas

Используется для задания множественных областей печати `CellRangePositions` в табличном документе.

```
Table.setPrintAreas(ranges)
```

4.69.29 Table.setPrintArea

Используется для задания области печати `CellRangePosition` в табличном документе.

```
Table.setPrintArea(range)
```

4.69.30 Table.getCharts

Метод используется для получения списка диаграмм `Charts`.

```
Table.getCharts()
```

4.69.31 Table.getNamedExpressions

Метод используется для получения списка именованных выражений `NamedExpressions`.

```
Table.getNamedExpressions ()
```

4.69.32 Table.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `Table`.

```
Table.__eq__(other)
```

4.69.33 Table.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `Table`.

```
Table.__ne__(other)
```

4.70 Именованные выражения

Именованное выражение – это выражение (являющееся описанием диапазона или формулой), которому присвоено имя. Преимуществом именованного выражения является его информативность. Именованные выражения упрощают работу с ячейками, также их удобно использовать при работе с формулами. На данный момент доступен функционал для работы с именованными выражениями, представляющими собой ссылки на диапазоны ячеек.

4.70.1 Класс NamedExpression

Класс описывает структуру именованного выражения.

```
class NamedExpression (object)
```

Список методов:

```
def getName()
def getExpression()
def getCellRange()
```

4.70.1.1 NamedExpression.getName

Возвращает имя именованного выражения.

```
NamedExpression.getName()
```

4.70.1.2 NamedExpression.getExpression

Возвращает текст выражения (формулы).

```
NamedExpression.getExpression()
```

4.70.1.3 NamedExpression.getCellRange

Возвращает диапазон ячеек `CellRange`, если выражение является ссылкой на диапазон.

```
NamedExpression.getCellRange()
```

4.70.2 Класс NamedExpressions

Список методов:

```
def get(name)
def getEnumerator()
```

4.70.2.1 NamedExpressions.get

Возвращает именованное выражение NamedExpression по имени, если оно существует.

```
NamedExpression.get(name)
```

4.70.2.2 NamedExpressions. getEnumerator

Возвращает объект типа Enumerator, позволяющий получить доступ ко всему списку именованных выражений.

```
NamedExpression.get()
```

4.71 Сводные таблицы

4.71.1 Класс PivotTable

Класс для представления сводной таблицы.

```
class PivotTable(object)
```

Список методов:

```
def remove()
def getSourceRangeAddress()
def getSourceRange()
def getPivotRange()
def changeSourceRange(sourceRange)
def isRowGrandTotalEnabled()
def isColumnGrandTotalEnabled()
def getPivotTableCaptions()
def getPivotTableLayoutSettings()
def getUnsupportedFeatures()
def getFieldsList()
def getRowFields()
def getColumnFields()
def getValueFields()
def getPageFields()
def getFieldCategories(fieldName)
def getFieldItems(fieldName)
def getFieldItemsByName(fieldName, itemName)
def getFilter(fieldName)
def getFilters()
def update()
def createPivotTableEditor()
```

4.71.1.1 PivotTable.remove

Метод удаляет сводную таблицу.

```
PivotTable.remove()
```

4.71.1.2 PivotTable.getSourceRangeAddress

Метод возвращает текстовое представление диапазона исходных данных сводной таблицы.

```
PivotTable.getSourceRangeAddress()
```

4.71.1.3 PivotTable.getSourceRange

Метод возвращает диапазон (`CellRange`) исходных данных сводной таблицы.

```
PivotTable.getSourceRange()
```

4.71.1.4 PivotTable.getPivotRange

Метод возвращает диапазон ячеек (`CellRange`), в котором размещена сводная таблица.

```
PivotTable.getPivotRange()
```

4.71.1.5 PivotTable.changeSourceRange

Метод позволяет задать новый диапазон исходных данных сводной таблицы без обновления самой таблицы.

```
PivotTable.changeSourceRange(sourceRange)
```

4.71.1.6 PivotTable.isRowGrandTotalEnabled

Метод возвращает `true`, если разрешено показывать общие итоги для строк.

```
PivotTable.isRowGrandTotalEnabled()
```

4.71.1.7 PivotTable.isColumnGrandTotalEnabled

Метод возвращает `true`, если разрешено показывать общие итоги для столбцов.

```
PivotTable.isColumnGrandTotalEnabled()
```

4.71.1.8 PivotTable.getPivotTableCaptions

Метод возвращает информацию о всех заголовках сводной таблицы `PivotTableCaptions`.

```
PivotTable.getPivotTableCaptions()
```

4.71.1.9 PivotTable.getPivotTableLayoutSettings

Метод возвращает настройки отображения PivotTableLayoutSettings сводной таблицы.

```
PivotTable.getPivotTableLayoutSettings()
```

4.71.1.10 PivotTable.getUnsupportedFeatures

Метод возвращает неподдерживаемые свойства PivotTableUnsupportedFeatures сводной таблицы.

```
PivotTable.getUnsupportedFeatures()
```

4.71.1.11 PivotTable.getFieldsList

Метод возвращает список всех полей сводной таблицы.

```
PivotTable.getFieldsList()
```

4.71.1.12 PivotTable.getRowFields

Метод возвращает список полей из области строк.

```
PivotTable.getRowFields()
```

4.71.1.13 PivotTable.getColumnFields

Метод возвращает список полей из области колонок.

```
PivotTable.getColumnFields()
```

4.71.1.14 PivotTable.getValueFields

Метод возвращает список полей из области значений.

```
PivotTable.getValueFields()
```

4.71.1.15 PivotTable.getPageFields

Метод возвращает список полей из области фильтров.

```
PivotTable.getPageFields()
```

4.71.1.16 PivotTable.getFieldCategories

Метод возвращает список категорий, содержащих заданное поле.

```
PivotTable.getFieldCategories()
```

4.71.1.17 PivotTable.getFieldItems

Метод возвращает все элементы сводной таблицы по заданному имени поля.

```
PivotTable.getFieldItems()
```

4.71.1.18 PivotTable.getFieldItemsByName

Метод возвращает все элементы из заданного поля по имени.

```
PivotTable.getFieldItemsByName()
```

4.71.1.19 PivotTable.getFilter

Метод возвращает фильтр по заданному имени поля.

```
PivotTable.getFilter()
```

4.71.1.20 PivotTable.getFilters

Метод возвращает список фильтров сводной таблицы.

```
PivotTable.getFilters()
```

4.71.1.21 PivotTable.update

Метод обновляет и полностью пересчитывает сводную таблицу.

```
PivotTable.update()
```

4.71.1.22 PivotTable.getPivotTableEditor

Метод возвращает объект `PivotTableEditor`, который служит для обновления свойств и редактирования сводной таблицы.

```
PivotTable.getPivotTableEditor()
```

4.71.2 Класс PivotTableCaptions

Данная структура хранит все пользовательские заголовки сводной таблицы.

```
class PivotTableCaptions(object)
```

Свойства класса:

- `errorCaption` – алиас для значений, которые возвращают ошибку;
- `emptyCaption` – алиас для значений, которые возвращают пустое значение;
- `grandTotalCaption` - алиас общих итогов;
- `valuesHeaderCaption` – алиас поля из области значений; это поле отображается в отчете в случае, если в сводной таблице наличие более двух полей из области значений, и макет имеет тип ‘outline’ или ‘tabular’;
- `rowHeaderCaption` – алиас заголовка строк (виден только при включенном компактном макете, это алиас по умолчанию);
- `columnHeaderCaption` – алиас заголовка колонок (виден только при включенном компактном макете, это алиас по умолчанию).

4.71.3 Класс PivotTableLayoutSettings

Класс предоставляет настройки отображения сводной таблицы.

```
class PivotTableLayoutSettings(object)
```

Свойства класса:

- `reportLayout` – настройка вида макета сводной таблицы `PivotTableReportLayout` (компактный, табличный, структурный);
- `valueFieldsOrientation` – позволяет расположить положение значений в случае, если в сводной таблице более двух полей значений (`ValueFieldsOrientation`);
- `pageFieldOrder` – настройка порядка полей фильтров `PageFieldOrder` (вниз, затем поперек или сначала поперек, потом вниз);
- `indentForCompactLayout` – размер отступа для полей в области строк в компактном макете (режим иерархии в случае наличия более двух полей);
- `pageFieldWrapCount` – настройка связана с полем `pageFieldOrder`, она показывает через сколько полей будет совершено указанное действие (перенос на следующую строку и т.д.);
- `isMergeAndCenterLabelsEnabled` – настройка позволяет объединить ячейки заголовков;
- `useGridDropZones` – использовать классический вид (как в Excel 2003). Влияет на расположение полей в отчете;
- `displayFieldCaptions` – отображать заголовки полей.

4.71.4 Тип PivotTableUnsupportedFeature

Перечисление, которое описывает неподдерживаемую функциональность сводных таблиц (см.Таблица 38).

Таблица 38 - Неподдерживаемая функциональность сводных таблиц

Имя константы вида отчетов	Описание вида отчетов
`PivotTableUnsupportedFeature_CalculatedField`	Вычисляемые поля.
`PivotTableUnsupportedFeature_CalculatedItem`	Вычисляемые элементы.
`PivotTableUnsupportedFeature_CollapsedValues`	Свернутые элементы поля.
`PivotTableUnsupportedFeature_ShowDataAs`	Дополнительные вычисления (Show data как в MS Excel).

4.71.5 Тип PivotTableReportlayout

Перечисление, описывающее внешний вид отчетов сводной таблицы (см.Таблица 39).

Таблица 39 - Вид отчетов сводной таблицы

Имя константы вида отчетов	Описание вида отчетов
PivotTableReportLayout_Compact	Компактный
PivotTableReportLayout_Tabular	Табличный
PivotTableReportLayout_Outline	Структурный

4.71.6 Тип ValueFieldsOrientation

Класс описывает варианты ориентации в случае, когда в сводной таблице более, чем одно поле из области значений (см. Таблица 40).

Таблица 40 - Виды ориентации полей

Имя константы ориентации полей	Описание
ValueFieldsOrientation_ByRows	По строкам.
ValueFieldsOrientation_ByColumns	По столбцам.

4.71.7 Тип PageFieldOrder

Тип описывает вид отображения полей из области фильтров (см.Таблица 41).

Таблица 41 - Вид отображения полей из области фильтров

Имя константы вида отображения полей	Описание
PageFieldOrder_DownThenOver	Вниз, затем поперек.
PageFieldOrder_OverThenDown	Поперек, затем вниз.
PivotTableFieldCategory_Columns	Область колонок.
PivotTableFieldCategory_Values	Область значений.

4.71.8 Тип PivotTableFieldCategory

Тип описывает категории области полей (см. Таблица 42).

Таблица 42 - Категории области полей

Имя константы категории области полей	Описание типа
PivotTableFieldCategory_Pages	Область фильтров.
PivotTableFieldCategory_Rows	Область строк.
PivotTableFieldCategory_Columns	Область колонок.
PivotTableFieldCategory_Values	Область значений.

4.71.9 Класс PivotTableFieldCategories

Класс предоставляет доступ к списку категорий области полей.

```
class PivotTableFieldCategories(object)
```

Список методов:

```
def getEnumerator()
```

4.71.9.1 PivotTableFieldCategories.getEnumerator

Возвращает объект типа `Enumerator`, позволяющий получить доступ ко всему списку категорий области полей.

```
PivotTableFieldCategories.getEnumerator()
```

4.71.10 Тип PivotTableFunction

Данный тип описывает функции, которые могут быть использованы в сводных таблицах (см. Таблица 43).

Таблица 43 - Функции сводной таблицы

Имя константы функции сводной таблицы	Описание типа
PivotTableFunction_Auto	Автозаполнение.
PivotTableFunction_Sum	Суммирует все числовые данные.
PivotTableFunction_Count	Количество всех ячеек.
PivotTableFunction_CountNums	Количество числовых ячеек.
PivotTableFunction_Average	Среднее значение.
PivotTableFunction_Max	Наибольшее значение.
PivotTableFunction_Min	Наименьшее значение.
PivotTableFunction_Product	Произведение всех ячеек.
PivotTableFunction_StdDeviation	Стандартное смещенное отклонение.
PivotTableFunction_StdDeviationPopulation	Стандартное несмещенное отклонение.
PivotTableFunction_Variance	Смещенная дисперсия.
PivotTableFunction_VariancePopulation	Несмещенная дисперсия.

4.71.11 Класс PivotTableFilter

Предназначен для доступа к списку фильтров таблицы, каждый из которых обладает свойством видимости. При любом изменении фильтров они должны быть применены к сводной таблице посредством использования метода `setFilter` класса `PivotTableEditor`.

```
class PivotTableFilter(object)
```

Список методов:

```
def getFieldName()
def getCount()
def getName(itemIndex)
def isHidden(itemIndex)
def isHidden(itemName)
def setHidden(itemIndex, hidden)
def setHidden(itemName, hidden)
```

4.71.11.1 PivotTableFilter.getFieldName

Возвращает имя поля, с которым ассоциирован фильтр.

```
PivotTableFilters.getFieldName()
```

4.71.11.2 PivotTableFilter.getCount

Возвращает количество фильтруемых полей.

```
PivotTableFilters.getCount()
```

4.71.11.3 PivotTableFilter.getName

Возвращает имя поля для заданного индекса `itemIndex`.

```
PivotTableFilters.getName(itemIndex)
```

4.71.11.4 PivotTableFilter.isHidden

Возвращает видимость поля для заданного индекса `itemIndex`. Если `true`, то поле скрыто.

```
PivotTableFilters.isHidden(itemIndex)
```

4.71.11.5 PivotTableFilter.setHidden

Устанавливает видимость поля. Параметры: `itemIndex` – индекс поля, `itemName` – имя поля, `hidden` – видимость (`true` – поле скрыто).

```
PivotTableFilters.setHidden(itemIndex, hidden)
PivotTableFilters.setHidden(itemName, hidden)
```

4.71.12 Класс PivotTableFilters

Класс обеспечивает доступ к списку фильтров сводной таблицы.

```
class PivotTableFilters(object)
```

4.71.12.1 PivotTableFilters.getEnumerator

Метод возвращает список фильтров сводной таблицы.

```
PivotTableFilters.getEnumerator()
```

4.71.13 Класс PivotTableFieldProperties

Класс содержит основные свойства полей сводной таблицы.

```
class PivotTableFieldProperties(object)
```

Свойства класса:

- `fieldName` – имя поля;
- `fieldAlias` – псевдоним поля (пользовательское имя поля);
- `subtotalAlias` – псевдоним подытогов конкретного поля.

4.71.14 Класс PivotTableField

Класс содержит свойства полей сводной таблицы.

```
class PivotTableField(object)
```

Свойства класса:

- `fieldProperties` – свойства полей сводной таблицы;
- `fieldCategories` – области полей сводной таблицы;
- `customFormula` – вычисляемая формула.

4.71.15 Класс PivotTableCategoryField

Класс, содержащий свойства поля сводной таблицы, использующегося как строка / столбец.

```
class PivotTableCategoryField(object)
```

Свойства класса:

- `fieldProperties` – свойства поля;
- `subtotalFunctions` – список функций для вычисления подытога.

4.71.16 Класс PivotTableValueField

Класс, содержащий свойства поля сводной таблицы, использующегося как значение.

```
class PivotTableValueField(object)
```

Свойства класса:

- `baseFieldName` – оригинальное поле на основе которого было создано данное поле;
- `valueFieldName` – автоматический уникальный псевдоним такой как “Sum of % имя поля%“;
- `cellNumberFormat` – числовой формат для конкретного поля значений;

- `totalFunction` – агрегирующая функция поля значений (SUM, COUNT, MAX и т.д.);
- `customFormula` – вычисляемая формула для поля значений.

4.71.17 Класс PivotTablePageField

Класс, содержащий свойства поля из области фильтров `PivotTableFieldProperties`.

```
class PivotTablePageField(object)
```

Свойства класса:

- `fieldProperties` – свойства поля из области фильтров.

4.71.18 Класс PivotTableItem

Класс описывает элемент сводной таблицы.

```
class PivotTableItem(object)
```

Список методов:

```
def getName()
def getAlias()
def getItemType()
def isCollapsed()
```

4.71.18.1 PivotTableItem.getName

Метод возвращает имя элемента сводной таблицы.

```
PivotTableItem.getName()
```

4.71.18.2 PivotTableItem.getAlias

Метод возвращает псевдоним элемента (идентификатор, созданный пользователем).

```
PivotTableItem.getAlias()
```

4.71.18.3 PivotTableItem.getItemType

Метод возвращает тип `PivotTableItemType` элемента сводной таблицы.

```
PivotTableItem.getItemType()
```

4.71.18.4 PivotTableItem.isCollapsed

Метод возвращает `true`, если элемент сводной таблицы свернут.

```
PivotTableItem.isCollapsed()
```

4.71.19 Тип PivotTableItemType

Данное перечисление описывает возможные типы элементов сводной таблицы (см.Таблица 44).

Таблица 44 - Типы элементов сводной таблицы

Имя константы типа сводной таблицы	Описание типа
PivotTableItemType_Number	Числовой.
PivotTableItemType_String	Строковый.
PivotTableItemType_Boolean	Логический.
PivotTableItemType_DateTime	Дата / время.
PivotTableItemType_Empty	Пустой тип.
PivotTableItemType_Error	Ошибка.
PivotTableItemType_NumberGroup	Интервальная группировка.
PivotTableItemType_DateIntervalGroup	Интервальная группировка по датам.
PivotTableItemType_DateTimeGroup	Группировка по дате / времени.
PivotTableItemType_CustomGroup	Пользовательская (произвольная) группировка.

4.71.20 Класс PivotTableEditor

Класс служит для редактирования сводных таблиц.

```
class PivotTableEditor(object)
```

Список методов:

```
def addField(fieldName, category, index)
def moveField(fieldName, category, index)
def removeField(fieldName, category)
def reorderField(fieldName, category, index)
def enableField(fieldName)
def disableField(fieldName)
def setSummarizeFunction(valueFieldName, summarizeFunction)
def setFilter(filter)
def setFilters(filters)
def setCaptions(captions)
def setLayoutSettings(layoutSettings)
def setGrandTotalSettings(isRowGrandTotalEnabled, isColGrandTotalEnabled)
def apply()
```

4.71.20.1 PivotTableEditor.addField

Метод добавляет новое поле в сводную таблицу, используя параметры: `fieldName` - имя поля, `toCategory` - область поля, `index` - позиция в области.

```
PivotTableEditor.addField(fieldName, category, index)
```

4.71.20.2 PivotTableEditor.moveField

Метод перемещает поле между областями. Параметры: `fieldName` - имя поля, `toCategory` - область, в которую перемещается поле, `index` - позиция в новой области.

```
PivotTableEditor.moveField(fieldName, category, index)
```

4.71.20.3 PivotTableEditor.removeField

Метод удаляет поле из области. Параметры: `fieldName` - имя поля, `fromCategory` - область, из которой удаляется поле.

```
PivotTableEditor.removeField(fieldName, category)
```

4.71.20.4 PivotTableEditor.reorderField

Метод изменяет позицию поля в пределах области. Параметры: `fieldName` - имя поля, `category` - область, `toIndex` - новая позиция поля.

```
PivotTableEditor.reorderField(fieldName, category, index)
```

4.71.20.5 PivotTableEditor.enableField

Метод добавлят поле в область, зависящую от типа поля. Параметр `fieldName` - имя поля.

```
PivotTableEditor.enableField(fieldName)
```

4.71.20.6 PivotTableEditor.disableField

Метод удаляет поле из всех областей. Параметр `fieldName` - имя поля.

```
PivotTableEditor.disableField(fieldName)
```

4.71.20.7 PivotTableEditor.setSummarizeFunction

Метод задает суммирующую функцию для поля из области значений. Параметр `valueFieldName` - имя поля, `summarizeFunction` – суммирующая функция.

```
PivotTableEditor.setSummarizeFunction(valueFueldName, summarizeFunction)
```

4.71.20.8 PivotTableEditor.setFilter

Метод задает фильтр сводной таблицы. Если фильтр не может быть применен, вызывается исключение `PivotTableError`.

```
PivotTableEditor.setFilter(filter)
```

4.71.20.9 PivotTableEditor.setFilters

Метод задает фильтры сводной таблицы. Если какой-то из фильтров не может быть применен, он пропускается.

```
PivotTableEditor.setFilters(filters)
```

4.71.20.10 PivotTableEditor.setCaptions

Метод задает заголовки сводной таблицы.

```
PivotTableEditor.setCaptions(captions)
```

4.71.20.11 PivotTableEditor.setLayoutSettings

Метод устанавливает настройки отображения `PivotTableLayoutSettings` сводной таблицы.

```
PivotTableEditor.setlayoutSettings(layoutSettings)
```

4.71.20.12 PivotTableEditor.setGrandTotalSettings

Метод задает настройки отображения общего итога. Параметры: `isRowGrandTotalEnabled` – показывать общие итоги для строк, `isColGrandTotalEnabled` – показывать общие итоги для столбцов.

```
PivotTableEditor
.setGrandTotalSettings(isRowGrandTotalEnabled,isColGrandTotalEnabled)
```

4.71.20.13 PivotTableEditor.apply

Метод обновляет сводную таблицу с заданными свойствами.

```
PivotTableEditor.apply()
```

4.71.21 Тип PivotTableUpdateResult

Данное перечисление описывает возможные результаты обновления сводной таблицы (см.Таблица 45).

Таблица 45 - Результаты обновления сводной таблицы

Имя константы результата сводной таблицы	Описание
`PivotTableUpdateResult_Success`	Успешное обновление таблицы.
`PivotTableUpdateResult_NoPivotTable`	Сводная таблица не найдена.
`PivotTableUpdateResult_NoSuchFieldInCategory`	Не найдено поле в области.
`PivotTableUpdateResult_NoSuchFieldInPivotTable`	Не найдено поле в сводной таблице.
`PivotTableUpdateResult_InvalidIndex`	Ошибка в индексе.
`PivotTableUpdateResult_FieldAlreadyEnabled`	Успешное обновление таблицы.
`PivotTableUpdateResult_MovingFieldToTheSameCategoryForbidden`	Попытка перемещения поля в рамках текущей области.
`PivotTableUpdateResult_InvalidFunction`	Неправильная функция.
`PivotTableUpdateResult_InvalidDataSourceRange`	Ошибка диапазона исходных данных.

Имя константы результата сводной таблицы	Описание
PivotTableUpdateResult_NoDataRowsInDataSource	В исходных данных нет строк с данными.
PivotTableUpdateResult_EmptyDataSourceHeaders	Пустые заголовки исходных данных.
PivotTableUpdateResult_NoReferenceUnderDefine	Попытка обновить/создать сводную таблицу на именованном диапазоне который не содержит ссылку, а содержит константу.
PivotTableUpdateResult_AnotherPivotInsideDataSource	Найдена другая сводная таблица в этом же диапазоне.

4.71.22 Класс PivotTablesManager

Класс служит для создания сводных таблиц.

```
class PivotTablesManager(object)
```

Список методов:

```
def create(cellRange, destination)
def create(formula, destination)
```

4.71.22.1 PivotTablesManager.create

Метод создает сводную таблицу на основе диапазона исходных данных.

Если местоположение (`destination`) не задано, создается новый лист (таблица), и сводная таблица будет расположена по умолчанию.

```
PivotTablesManager.create(cellRange, destination)
```

Второй вариант метода `create` создает сводную таблицу на основе строки формулы.

Если местоположение (`destination`) не задано, создается новая таблица и сводная таблица будет расположена по умолчанию.

```
PivotTablesManager.create(formula, destination)
```

4.72 Диаграммы

4.72.1 Класс Chart

Класс представляет диаграмму в документе со всеми элементами (заголовок, легенда, тип, данные, диапазон и т.д.).

```
class Chart(object)
```

Список методов:

```
def getType()
def setType(chartType)
def getRangesCount()
```

```
def getRange(rangesIndex)
def getTitle()
def setRange(chartRange)
def setRect(rect)
def isEmpty()
def isSolidRange()
def is3D()
def getDirectionType()
def getChartLabels()
def getRangeAsString()
def applySettings(rangeInfo, directionType, title, labelsInfo)
```

4.72.1.1 Chart.getType

Метод возвращает тип диаграммы `ChartType`.

```
Chart.getType()
```

4.72.1.2 Chart.setType

Метод устанавливает тип диаграммы `ChartType`. Параметр `chartType` - новый тип диаграммы.

```
Chart.setType(chartType)
```

4.72.1.3 Chart.getRangesCount

Метод возвращает количество серий диаграммы.

```
Chart.getRangesCount()
```

4.72.1.4 Chart.getRange

Метод возвращает диапазон ячеек с исходными данными диаграммы.

```
Chart.getRange(rangeIndex)
```

4.72.1.5 Chart.getTitle

Метод возвращает заголовок диаграммы `ChartType`.

```
Chart.getTitle()
```

4.72.1.6 Chart.setRange

Метод задает диапазон ячеек с исходными данными для диаграммы.

```
Chart.setRange(chartRange)
```

4.72.1.7 Chart.setRect

Метод задает область расположения диаграммы.

```
Chart.setRect(rect)
```

4.72.1.8 Chart.isEmpty

Метод возвращает `true`, если диаграмма не содержит значений.

```
Chart.isEmpty()
```

4.72.1.9 Chart.isSolidRange

Метод возвращает `true`, если диапазон исходных данных диаграммы может быть выделен одним прямоугольником и не имеет промежутков.

```
Chart.isSolidRange()
```

4.72.1.10 Chart.is3D

Метод возвращает `true`, если диаграмма трехмерная.

```
Chart.is3D()
```

4.72.1.11 Chart.getDirectionType

Метод возвращает направление серий диаграммы.

```
Chart.getDirectionType()
```

4.72.1.12 Chart.getChartLabels

Метод возвращает коллекцию меток диаграммы типа `ChartLabelsInfo`.

```
Chart.getChartLabels()
```

4.72.1.13 Chart.getRangeAsString

Метод возвращает диапазон ячеек диаграммы в формате строки.

```
Chart.getRangeAsString()
```

4.72.1.14 Chart.applySettings

Метод позволяет применить следующие параметры к текущей выбранной диаграмме: `rangeInfo` – обновленный диапазон исходных данных диаграммы, `directionType` – направление серий, `title` – заголовок диаграммы, `labelsInfo` – информация о метках диаграммы.

```
Chart.applySettings(rangeInfo, directionType, title, labelsInfo)
```

4.72.2 Класс Charts

Класс Charts используется для получения доступа к списку диаграмм документа.

```
class Charts(object)
```

Список методов:

```
def getChartsCount()
def getChart(chartIndex)
def getChartIndexByDrawingIndex(drawingIndex)
```

4.72.2.1 Charts.getChartsCount

Метод возвращает общее количество диаграмм.

```
Charts.getChartsCount()
```

4.72.2.2 Charts.getChart

Метод возвращает диаграмму `Chart` по индексу `chartIndex` в коллекции диаграмм.

```
Charts.getChart(chartIndex)
```

4.72.2.3 Charts.getChartIndexByDrawingIndex

Метод возвращает диаграмму `Chart` по индексу отрисовки `drawingIndex`.

```
Charts.getChartIndexByDrawingIndex(drawingIndex)
```

4.72.3 Тип ChartLabelsDetectionMode

Тип описывает режимы автоматического определения меток диаграмм (см. Таблица 46).

Таблица 46 - Режимы автоматического определения меток диаграмм

Имя константы типа диапазона исходных данных диаграммы	Описание
`ChartLabelsDetectionMode_Unknown`	Неопределенный тип.
`ChartLabelsDetectionMode_FirstRow`	Метка на первой строке.
`ChartLabelsDetectionMode_FirstColumn`	Метка на первой колонке.
`ChartLabelsDetectionMode_NoLabels`	Не отрисовывать метки.

4.72.4 Класс ChartLabelsInfo

Класс описывает настройки автоматического определения меток диаграммы. Инициализируется конструктором, в который передаются параметры: `categoriesMode` - режим автоматического определения меток для областей, `seriesNameMode` - режим автоматического определения меток для серий, `oneColumnRow` - передается `true`, если диапазон диаграммы содержит только одну строку или одну колонку.

```
class ChartLabelsInfo(object)
```

Поля класса:

- `categoriesMode` – режим автоматического определения меток для областей `ChartLabelsDetectionMode`;
- `seriesNameMode` – режим автоматического определения меток для серий `ChartLabelsDetectionMode`;
- `isOneColumnRowChart` – поле содержит `true`, если диапазон диаграммы содержит только одну строку или одну колонку.

4.72.5 Класс ChartRangeInfo

Класс описывает серию диаграммы. Инициализируется конструктором, в который передаются следующие параметры: `tableRangeInfo` - диапазон ячеек, `color` – цвет серии диаграммы, `hidden` – видимость серии, `rangeType` – тип диапазона исходных данных диаграммы.

```
class ChartRangeInfo(object)
```

Поля класса:

- `tableRangeInfo` – исходный диапазон ячеек для серии;
- `rangeColor` – цвет для отрисовки серии;
- `isHidden` – задает видимость серии диаграммы;
- `rangeType` – тип диапазона диаграммы.

4.72.6 Тип ChartRangeType

Класс описывает тип диапазона исходных данных диаграммы (см. Таблица 47).

Таблица 47 - Направление серий диаграмм

Имя константы типа диапазона исходных данных диаграммы	Описание
`ChartRangeType_Series`	Серии.
`ChartRangeType_SeriesName`	Имена серий.
`ChartRangeType_Categories`	Области.
`ChartRangeType_DataPoint`	Разметка данных.

4.72.7 Тип ChartSeriesDetectionType

Тип описывает направление серий диаграмм (см. Таблица 48).

Таблица 48 - Направление серий диаграмм

Имя константы направления серии диаграммы	Описание
ChartSeriesDirectionType_Unknown	Неопределенный тип.
ChartSeriesDirectionType_ByRow	Серии направлены по строкам.
ChartSeriesDirectionType_ByColumn	Серии направлены по колонкам.

4.72.8 Тип ChartType

Перечисление `ChartType` описывает все поддерживаемые типы диаграмм (см. Таблица 49)

Таблица 49 - Типы диаграмм

Имя константы типа диаграммы	Описание
ChartType_Unknown	Неопределенный тип.
ChartType_Bar	Линейчатая диаграмма с группировкой.
ChartType_BarStacked	Линейчатая диаграмма с накоплением.
ChartType_BarPercentStacked	Линейчатая нормированная диаграмма с накоплением.
ChartType_Column	Гистограмма с группировкой.
ChartType_ColumnStacked	Гистограмма с накоплением.
ChartType_ColumnPercentStacked	Нормированная гистограмма с накоплением.
ChartType_Line	Стандартный график.
ChartType_LineStacked	График с накоплением.
ChartType_LinePercentStacked	Нормированный график с накоплением.
ChartType_LineWithMarker	Стандартный график с маркерами.
ChartType_LineWithMarkerStacked	График с накоплением и маркерами.
ChartType_LineWithMarkerPercentStacked	Нормированный график с накоплением и маркерами.
ChartType_Area	Стандартная диаграмма с областями.
ChartType_AreaStacked	Диаграмма с областями с накоплением.
ChartType_AreaPercentStacked	Нормированная диаграмма с областями с накоплением.
ChartType_Pie	Круговая диаграмма.
ChartType_PieExploded	Круговая диаграмма с отделенными секторами.
ChartType_Scatter	Диаграмма рассеяния.

4.73 Класс Blocks

Предоставляет доступ к блокам документа.

```
class Blocks(object)
```

Список методов:

```
def getBlock(blockIndex)
def getParagraph(paragraphIndex)
```

```
def getField(fieldIndex)
def getShape(shapeIndex)
def getTable(tableIndex)
def getTable(tableName)
def getEnumerator()
def getParagraphsEnumerator()
def getTablesEnumerator()
def getShapesEnumerator()
def getFieldsEnumerator()
```

4.73.1 Blocks.getBlock

Возвращает блок `Block` по индексу или `none`, если блок с таким индексом не существует в документе.

```
Blocks.getBlock(blockIndex)
```

4.73.2 Blocks.getParagraph

Возвращает абзац `Paragraph` по индексу или `none`, если абзац с таким индексом не существует в документе.

```
Blocks.getParagraph(paragraphIndex)
```

4.73.3 Blocks.getTable

Возвращает таблицу `Table` или `none`, если таблица с таким индексом или именем не существует в документе.

```
Blocks.getTable(tableIndex)
Blocks.getTable(tableName)
```

Параметры:

- `tableIndex` – индекс таблицы в документе, индексация начинается с нуля.
- `tableName` – имя таблицы.

4.73.4 Blocks.getField

Возвращает поле `Field` или `none`, если поле с таким индексом не существует в документе.

```
Blocks.getField(fieldIndex)
```

4.73.5 Blocks.getShape

Возвращает фигуру `Shape` или `none`, если фигура с таким индексом не существует в документе.

```
Blocks.getShape(shapeIndex)
```

4.73.6 Blocks.getEnumerator

Возвращает коллекцию классов Block.

```
Blocks.getEnumerator()
```

4.73.7 Blocks.getParagraphsEnumerator

Возвращает коллекцию классов Paragraphs.

```
Blocks.getParagraphsEnumerator()
```

4.73.8 Blocks.getTablesEnumerator

Возвращает коллекцию классов Table.

```
Blocks.getTablesEnumerator()
```

4.73.9 Blocks.getShapesEnumerator

Возвращает коллекцию классов Shape.

```
Blocks.getShapesEnumerator()
```

4.73.10 Blocks.getFieldsEnumerator

Возвращает коллекцию объектов Field.

```
Blocks.getFieldsEnumerator()
```

4.74 Класс Bookmarks

Предоставляет интерфейс для доступа к закладкам документа.

```
class Bookmarks(object)
```

Список методов:

```
def getBookmarkRange(bookmarkName)
def removeBookmark(bookmarkName)
```

4.74.1 Bookmarks.getBookmarkRange

Возвращает диапазон Range, содержащий закладку с наименованием bookmarkName, или none, если нет такой закладки.

```
Bookmarks.getBookmarkRange(bookmarkName)
```

4.74.2 Bookmarks.removeBookmark

Удаляет закладку с наименованием bookmarkName.

```
Bookmarks.removeBookmark(bookmarkName)
```

4.75 Класс Document

Предоставляет доступ к объектной модели документа и используется для управления содержимым текстового или табличного документа.

```
class Document(object)
```

Список методов:

```
def saveAs(filePath)
def saveAs(filePath, settings)
def exportAs(filePath, format)
def exportAs(filePath, format, settingsText)
def exportAs(filePath, format, settingsWorkbook)
def getBlocks()
def getBookmarks()
def getComments()
def getNamedExpressions()
def getSections()
def getParagraphs()
def getScripts()
def getRange()
def merge(other)
def setChangesTrackingEnabled(enabled)
def isChangesTrackingEnabled()
def setPageProperties(properties)
def setPageOrientation(orientation)
def getSectionsEnumerator()
def setMirroredMarginsEnabled(enabled)
def areMirroredMarginsEnabled()
def merge(document)
def getPivotTablesManager()
```

4.75.1 Document.saveAs

Сохраняет документ в файл по указанному пути. Формат и тип документа определяются расширением файла, если не указаны в явном виде.

```
Document.saveAs(filePath)
Document.saveAs(filePath, settings)
```

Параметры:

- `filePath` – строковое значение, содержит путь сохранения файла, включая имя файла и расширение;
- при необходимости в качестве второго аргумента можно использовать объект типа `SaveDocumentSettings`, который содержит формат документа, тип документа и пароль для защиты документа от несанкционированного доступа.

4.75.2 Document.exportAs

Экспортирует документ в файл по указанному пути и с указанным форматом. В настоящее время поддерживается операция экспорта документа только в формат PDF/A-1b.

```
Document.exportAs(filePath, format)
Document.exportAs(filePath, format, settingsText)
Document.exportAs(filePath, format, settingsWorkbook)
```

Параметры:

- `filePath` – строковое значение содержит путь сохранения файла, включая имя файла и расширение;
- `format` – формат экспорта файла `ExportFormat`;
- `settingsText` – дополнительные настройки для текстовых документов `TextExportSettings`;
- `settingsWorkbook` – дополнительные настройки для табличных документов `WorkbookExportSettings`.

Внимание ! Скрытые листы в табличном документе не экспортируются.

4.75.3 Document.getBlocks

Обеспечивает доступ к блокам данных `Blocks`, содержащихся в документе.

```
Document.getBlocks()
```

4.75.4 Document.getBookmarks

Обеспечивает доступ к коллекции закладок `Bookmarks` в документе.

```
Document.getBookmarks()
```

4.75.5 Document.getComments

Обеспечивает доступ к коллекции комментариев `Comments`, которые хранятся в документе.

```
Document.getComments()
```

4.75.6 Document.getParagraphs

Обеспечивает доступ к коллекции абзацев `Paragraphs`, которые хранятся в документе.

```
Document.getParagraphs()
```

4.75.7 Document.getScripts

Обеспечивает доступ к коллекции макрокоманд Scripts в документе.

```
Document.getScripts()
```

4.75.8 Document.getRange

Возвращает диапазон Range, содержащий весь документ.

```
Document.getRange()
```

4.75.9 Document.merge

Возвращает Document, в котором изменения отмечены с помощью механизма отслеживания изменений.

```
Document.merge(other)
```

4.75.10 Document.setChangesTrackingEnabled

Устанавливает включение/выключение отслеживания изменений в документе. Параметр enabled должен быть установлен в true, если отслеживание изменений в документе необходимо включить и в false, если отслеживание изменений в документе необходимо выключить.

```
Document.setChangesTrackingEnabled(enabled)
```

4.75.11 Document.isChangesTrackingEnabled

Возвращает значение true, если отслеживание изменений в документе включено, или false, если отслеживание изменений в документе выключено.

```
Document.isChangesTrackingEnabled()
```

4.75.12 Document.setPageProperties

Устанавливает свойства PageProperties страниц в документе.

```
Document.setPageProperties(properties)
```

4.75.13 Document.setPageOrientation

Устанавливает ориентацию PageOrientation страниц в документе.

```
Document.setPageOrientation(orientation)
```

4.75.14 Document.getSectionsEnumerator

Возвращает коллекцию классов Section.

```
Document.getSectionsEnumerator()
```

4.75.15 Document.setMirroredMarginsEnabled

Устанавливает включение или выключение зеркальных полей в документе. Параметр `enabled` должен быть установлен в `true`, если необходимо включить зеркальные поля в документе и в `false`, если зеркальные поля в документе необходимо выключить.

```
Document.setMirroredMarginsEnabled(enabled)
```

4.75.16 Document.areMirroredMarginsEnabled

Возвращает значение `true`, если включены зеркальные поля в документе, `false`, если зеркальные поля в документе выключены.

```
Document.areMirroredMarginsEnabled()
```

4.75.17 Document.getNamedExpressions

Используется для получения списка именованных выражений, возвращает объект типа `NamedExpressions`.

```
Document.getNamedExpressions()
```

4.75.18 Document.getSections

Используется для доступа к разделам документа, возвращает `Sections`.

```
Document.getSections()
```

4.75.19 Document.getPivotTablesManager

Возвращает `PivotTablesManager`, который используется для создания сводных таблиц.

```
Document.getPivotTablesManager()
```

4.76 Класс TextProperties

Содержит свойства оформления фрагмента текста (диапазона). Это может быть либо весь абзац, либо его часть, либо даже текст в нескольких ячейках таблицы.

```
class TextProperties(object)
```

Список методов

```
def __eq__(other)
def __ne__(other)
```

Список свойств:

- `fontName` – наименование шрифта;
- `fontSize` – размер шрифта;
- `bold` – содержит значение `true`, если шрифт текста «полужирный»;
- `italic` – содержит значение `true`, если шрифт текста «курсив»;
- `underline` – содержит значение `true`, если шрифт текста «подчеркнутый»;
- `strikethtough` – содержит значение `true`, если шрифт текста «зачеркнутый»;
- `allCapitals` – содержит значение `true`, если в тексте все символы прописные;
- `scriptPosition` – тип надстрочного/подстрочного форматирования;
- `textColor` – цветовая модель текста `Color`;
- `backgroundColor` – цветовая модель фона текста `ColorRGBA`;
- `characterSpacing` – межзнаковый интервал.

4.76.1 TextProperties.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `TextProperties`.

```
TextProperties.__eq__(other)
```

4.76.2 TextProperties.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `TextProperties`.

```
TextProperties.__ne__(other)
```

4.77 Класс LocaleInfo

Предоставляет информацию о локализации.

```
class LocaleInfo(object)
```

Список свойств:

- `localName` – название локализации, представлено в формате <language> <REGION>, где языковой код соответствует стандарту ISO-639, а код региона стандарту ISO-3166;
- `decimalSeparator` – десятичный разделитель, отделяет целые и дробные части чисел;
- `thousandSeparator` – символ, разделяющий группы цифр в числовых значениях;
- `listSeparator` – символ, отделяющий элементы в списке;
- `currencySymbol` – символ валюты, используемой в текущей стране или регионе;
- `currencyFormat` – расположение знака валюты в текущем регионе;
- `shortDatePattern` – заданный «короткий» формат отображения даты в текущем регионе (например, 'm/d/yy' для en_US);
- `longDatePattern` – заданный «длинный» формат отображения даты в текущем регионе (например, 'dddd, mmmm d, yyyy' for en_US);
- `timePattern` – заданный формат отображения времени в текущем регионе (например, 'h:mm AM/PM' для en_US).

4.78 Класс TimeZone

Предоставляет информацию о часовом поясе.

```
class TimeZone(object)
```

4.79 Класс DocumentSettings

Предоставляет настройки, необходимые на протяжении всего периода работы с документом.

```
class DocumentSettings(object)
```

Список свойств:

- `documentType` – тип документа `DocumentType`;
- `userInfo` – информация о пользователе `UserInfo`;
- `localeInfo` – информация о локализации `LocaleInfo`;
- `timeZone` – информация о временной зоне `TimeZone`;
- `formulaType` – система адресации ячеек `FormulaType`.

4.80 Класс DSVSettings

Предоставляет настройки, необходимые для работы с файлами CSV (comma-separated value) и DSV (delimiter-separated value).

```
class DSVSettings(object)
```

Список свойств:

- `separator` – символ-разделитель полей данных, значение по умолчанию – запятая;
- `escapeChar` – символ-разделитель для текстовых строк, значение по умолчанию – двойная кавычка;
- `autofit` – признак необходимости автоматического подстраивания ширины столбца под размер данных в ячейке;
- `startBlockIndex` – поле содержит индекс блока документа для начала сохранения;
- `startBlockIndex` – поле содержит индекс блока документа для окончания сохранения.

4.81 Класс LoadDocumentSettings

Предоставляет дополнительные настройки, необходимые для загрузки документов из файла.

```
class LoadDocumentSettings(object)
```

Список свойств:

- `commonDocumentSettings` – экземпляр класса `DocumentSettings`, общие настройки документа;
- `encoding` – кодировка документа `Encoding`;
- `dsvSettings` – экземпляр класса `DSVSettings`, настройки, необходимые для работы с файлами CSV и DSV;
- `documentPassword` – пароль для защиты электронного документа от несанкционированного доступа. Механизм парольной защиты поддерживается только для семейства ОС Microsoft Windows.

4.82 Класс SaveDocumentSettings

Предоставляет дополнительные настройки, необходимые для сохранения документа в файл.

```
class SaveDocumentSettings(object)
```

Список свойств:

- `documentFormat` – формат документа `DocumentFormat`;
- `documentType` – тип документа `DocumentType`;
- `documentPassword` – пароль для защиты электронного документа от несанкционированного доступа. Механизм парольной защиты поддерживается только для семейства ОС Microsoft Windows;
- `isTemplate` – флаг, обозначающий, что документ должен быть сохранен как шаблон;
- `dsvSettings` – структура `DSVSettings`, необходимая для сохранения в формате DSV.

4.83 Класс Application

Управляет параметрами и объектами приложения. Предоставляет интерфейс для создания и загрузки документов. Как правило, создается только один объект класса для всего сеанса работы с документом.

```
class Application(object)
```

Список методов:

```
def getMessenger()
def createDocument(documentType)
def createDocument(settings)
def loadDocument(filePath)
def loadDocument(filePath, settings)
```

4.83.1 Application.getMessenger

Метод `getMessenger` возвращает объект `Messenger`, реализующий логирование событий.

```
Application.getMessenger()
```

4.83.2 Application.createDocument

Создает новый документ с заданным типом `documentType`, либо с настройками `settings`, указанными в параметрах.

```
Application.createDocument(documentType)
Application.createDocument(settings)
```

Параметры:

- `documentType` – тип документа `DocumentType`, создается новый документ указанного типа с настройками по умолчанию.

Если в качестве аргумента указывается объект `settings` класса `DocumentSettings`, то в этом случае создается новый документ с настройками, указанными в объекте.

4.83.3 Application.loadDocument

Загружает существующий текстовый или табличный документ из файла, находящего по указанному в аргументе пути. Формат и тип документа определяются из расширения файла, если не указаны явно с помощью класса `LoadDocumentSettings`. Метод возвращает экземпляр `Document` созданного документа.

```
Application.loadDocument(filePath)
Application.loadDocument(filePath, settings)
```

Параметры:

- `filePath` – содержит путь к текстовому или табличному документу, в этом случае формат и тип документа определятся из расширения файла.

Если аргумент содержит объект класса LoadDocumentSettings, то в этом случае откроется документ с характеристиками, указанными в объекте.

4.84 Класс Messenger

Класс Messenger предоставляет возможность работы по логированию событий.

```
class Messenger(object)
```

Список методов:

```
def subscribe(handler)
def notify(message)
```

4.84.1 Messenger.subscribe

Метод subscribe служит для подписки на события лога. В качестве параметра передается обработчик событий MessageHandler. Метод возвращает соединение Connection.

```
Messenger.subscribe(handler)
```

4.84.2 Messenger.notify

Метод notify используется для создания события лога, в качестве параметра передается Message.

```
Messenger.notify(message)
```

4.85 Класс Message

Класс Message предназначен для формирования событий лога.

```
class Message(object)
```

Список методов:

```
def getSeverity()
def getText()
def makeInfo(text)
def makeWarning(text)
def makeError(text)
def __eq__(other)
def __ne__(other)
```

4.85.1 Message.getSeverity

Метод возвращает уровень лога Severity.

```
Message.getSeverity()
```

4.85.2 Message.getText

Метод возвращает текст сообщения.

```
Message.getText()
```

4.85.3 Message.makeInfo

Метод создает сообщение уровня `Info` с заданным текстом.

```
Message.makeInfo(text)
```

4.85.4 Message.makeWarning

Метод создает сообщение уровня `Warninig` с заданным текстом.

```
Message.makeWarning(text)
```

4.85.5 Message.makeError

Метод создает сообщение уровня `Error` с заданным текстом.

```
Message.makeError(text)
```

4.85.6 Message.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `Message`.

```
Message.__eq__(other)
```

4.85.7 Message.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `Message`.

```
Message.__ne__(other)
```

4.86 Уровни сообщений лога Severity

Поддерживаемые уровни сообщений лога представлены в Таблице 50.

Таблица 50 - Уровни сообщений лога

Имя константы	Уровень сообщения
`Severity_Info`	Информация
`Severity_Warning`	Предупреждение.
`Severity_Error`	Ошибка.

4.87 Класс MessageHandler

Класс предназначен для обработки сообщений лога, метод `handle` вызывается при возникновении сообщения.

```
class MessageHandler(object)
```

4.88 Класс Connection

Класс `Connection` реализует соединение между `Messenger` и клиентом. Содержит один метод `unsubscribe`, служащий для разрыва соединения.

```
class Connection(object)
```

4.89 Класс Scripting

Управляет виртуальной машиной Lua. Класс предоставляет интерфейс для выполнения макрокоманд Lua, которые хранятся в электронном документе.

```
class Scripting(object)
```

Список методов:

```
def runScript(name)
```

4.89.1 Scripting.runScript

```
Scripting.runScript(name)
```

Параметры:

- `name` – название макрокоманды.

Описание метода:

Запускает макрокоманду с названием `name`.

4.90 Глобальная функция createScripting

Создает объект `Scripting` для доступа к макрокомандам в электронном документе.

```
createScripting(document)
```

4.91 Класс PointU

Представляет собой точку в двухмерном пространстве.

```
class PointU(object)
```

Список свойств:

- x – координата точки по оси x;
- y – координата точки по оси y.

Список методов:

```
def toString()
```

4.91.1 PointU.toString

Возвращает информацию о координатах точки в виде строкового значения формата `(x: <value>, y: <value>)`.

```
PointU.toString()
```

4.92 Класс RectU

Описывает прямоугольник в двухмерном пространстве.

```
class RectU(object)
```

Список свойств:

- `topLeft` – координата левой верхней вершины прямоугольника;
- `bottomRight` – координата правой нижней вершины прямоугольника.

Список методов:

```
def toString()
```

4.92.1 RectU.toString

Возвращает информацию о положении прямоугольника в виде строкового значения формата `(topLeft: <value>, bottomRight: <value>)`.

```
RectU.toString()
```

4.93 Класс SizeU

Описывает размер объекта в двухмерном пространстве.

```
class SizeU(object)
```

Список свойств:

- `width` – ширина объекта в двухмерном пространстве;
- `height` – высота объекта в двухмерном пространстве.

Список методов:

```
def toString()
```

4.93.1 SizeU.toString

Возвращает информацию о размерах объекта в виде строкового значения формата `(width: <value>, height: <value>)`.

```
SizeU.toString()
```

4.94 Класс HeaderFooter

Определяет отдельный колонтитул в документе.

```
class HeaderFooter(object)
```

Список методов:

```
def getType()
def getBlocks()
def getRange()
```

4.94.1 HeaderFooter.getType

Возвращает тип колонтитула `HeaderFooterType`.

```
HeaderFooter.getType()
```

4.94.2 HeaderFooter.getBlocks

Предоставляет доступ к блокам `Blocks`, которые содержатся в колонтитуле.

```
HeaderFooter.getBlocks()
```

4.94.3 HeaderFooter.getRange

Предоставляет диапазон `Range` с содержанием верхнего или нижнего колонтитулов.

```
HeaderFooter.getRange()
```

4.95 Класс HeadersFooters

Представляет коллекцию верхних и нижних колонтитулов в разделе документа. См. описание класса `Section`.

```
class HeadersFooters(object)
```

Список методов:

```
def getEnumerator()
```

4.95.1 HeadersFooters.getEnumerator

Возвращает коллекцию колонтитулов.

```
HeadersFooters.getEnumerator()
```

4.96 Класс TextOrientation

Управляет свойством ориентации текста в ячейке, фигуре и т.д.

```
class TextOrientation(object)
```

Список методов:

```
def getAngle()
def isStackedChars()
def __eq__(other)
def __ne__(other)
```

4.96.1 TextOrientation.getAngle

Возвращает угол ориентации текста в ячейке. Значение угла указывается в градусах.

```
TextOrientation.getAngle()
```

4.96.2 TextOrientation.isStackedChars

Возвращает `true`, если ориентация текста - вертикальный столбец.

```
TextOrientation.isStackedChars()
```

4.96.3 TextOrientation.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `TextOrientation`.

```
TextOrientation.__eq__(other)
```

4.96.4 TextOrientation.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `TextOrientation`.

```
TextOrientation.__ne__(other)
```

4.97 Класс TextExportSettings

Предоставляет настройки, необходимые для экспорта текстовых документов.

```
class TextExportSettings(object)
```

Свойство:

- `pageNumbers` – содержит настройки страниц `PageNumbers` для экспорта текстовых документов.

4.98 Класс WorkbookExportSettings

Предоставляет настройки, необходимые для экспорта табличных документов.

```
class WorkbookExportSettings(object)
```

Список свойств:

- `sheetNames` – представляет коллекцию имен листов `SheetNames` для экспорта. Если коллекция пуста, экспортируются все листы;
- `printingScope` – представляет область печати `PrintingScope` (весь документ, область печати, пользовательский диапазон и т.д.);
- `pageProperties` – представляет свойства страницы для выходного документа `PageProperties` (высота и ширина страницы в пунктах (pt));
- `scale` – представляет масштаб экспорта выходного документа в процентах (например, 50,0%, 150,63%, 400,0% и т. д.).

Для присвоения значения коллекции имен листов `sheetNames` необходимо использовать преобразование `VectorString`

Пример:

```
export_doc_settings = DocumentAPI.WorkbookExportSettings()

export_doc_settings.sheetNames = DocumentAPI.VectorString(["Sheet1", "Лист1"])
```

4.99 Класс PrintingScope

Содержит настройки для экспорта табличных документов. Позволяет создать области печати следующих типов:

- выбранная область печати `PrintingScopeType`;
- указанный диапазон ячеек `CellRangePosition`.

```
class PrintingScope(object)
```

Список методов:

```
def usePrintArea()
def getCellRange()
```

4.99.1 PrintingScope.usePrintArea

Возвращает `true`, если область печати должна использоваться во время печати, экспорта и т. д.

```
PrintingScope.usePrintArea()
```

4.99.2 PrintingScope.getCellRange

Возвращает диапазон ячеек `CellRangePosition` таблицы.

```
PrintingScope.getCellRange()
```

4.100 Класс PageNumbers

Представляет собой коллекцию страниц для экспорта. Позволяет установить следующие типы страниц для экспорта:

- нечетные, четные страницы `PageParity`;
- список номеров страниц;
- диапазон страниц, с указанием начальной и конечной страницы.

```
class PageNumbers(object)
```

Список методов:

```
def contains()
def getLast()
```

4.100.1 PageNumbers.contains

Возвращает `true`, если указанное в параметре `pageNumber` значение входит в коллекцию номеров страниц.

```
PageNumbers.contains(pageNumber)
```

Параметр:

- `pageNumber` – номер страницы.

4.100.2 PageNumbers.getLast

Возвращает последний номер страницы или `none` в случае нечетно-четных или всех страниц.

```
PageNumbers.getLast()
```

4.101 Класс HorizontalTextAnchoredPosition

Предназначен для управления относительным положением объекта со смещением или выравниванием по горизонтали.

```
class HorizontalTextAnchoredPosition()
```

Список свойств:

- `relativeTo` – тип размещения объекта относительно закрепленной позиции по горизонтали `HorizontalRelativeTo`;
- `offset` – смещение объекта;
- `aligment` – тип выравнивания объекта относительно закрепленной позиции по горизонтали `HorizontalAnchorAlignment`.

Список методов:

```
def __eq__(other)
def __ne__(other)
```

4.101.1 HorizontalTextAnchoredPosition.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `HorizontalTextAnchoredPosition`.

```
HorizontalTextAnchoredPosition.__eq__(other)
```

4.101.2 HorizontalTextAnchoredPosition.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `HorizontalTextAnchoredPosition`.

```
HorizontalTextAnchoredPosition.__ne__(other)
```

4.102 Класс VerticalTextAnchoredPosition

Предназначен для управления относительным положением объекта со смещением или выравниванием по вертикали.

```
class VerticalTextAnchoredPosition()
```

Список свойств:

- `relativeTo` – тип размещения объекта относительно закрепленной позиции по вертикали `VerticalRelativeTo`;
- `offset` – смещение объекта;
- `aligment` – тип выравнивания объекта относительно закрепленной позиции по вертикали `VerticalAnchorAlignment`.

Список методов:

```
def __eq__(other)
def __ne__(other)
```

4.102.1 VerticalTextAnchoredPosition.__eq__

Метод __eq__ используется для определения эквивалентности двух объектов типа VerticalTextAnchoredPosition.

```
VerticalTextAnchoredPosition.__eq__(other)
```

4.102.2 VerticalTextAnchoredPosition.__ne__

Метод __ne__ используется для определения неэквивалентности двух объектов типа VerticalTextAnchoredPosition.

```
VerticalTextAnchoredPosition.__ne__(other)
```

4.103 Класс TextAnchoredPosition

Представляет позицию объекта на странице текстового документа.

```
class TextAnchoredPosition()
```

Список свойств:

- horizontal – позиция по горизонтали HorizontalRelativeTo;
- vertical – позиция по вертикали VerticalRelativeTo.

Список методов:

```
def __eq__(other)
def __ne__(other)
```

4.103.1 TextAnchoredPosition.__eq__

Метод __eq__ используется для определения эквивалентности двух объектов типа TextAnchoredPosition.

```
TextAnchoredPosition.__eq__(other)
```

4.103.2 TextAnchoredPosition.__ne__

Метод __ne__ используется для определения неэквивалентности двух объектов типа TextAnchoredPosition.

```
TextAnchoredPosition.__ne__(other)
```

4.104 Класс AnchoredPosition

Представляет позицию с относительными смещениями на странице текстового документа.

```
class AnchoredPosition()
```

Список свойств:

- `textPosition` – позиция на странице текстового документа `AnchoredPosition`).

Список методов:

```
def __eq__(other)
def __ne__(other)
```

4.104.1 AnchoredPosition.__eq__

Метод `__eq__` используется для определения эквивалентности двух объектов типа `AnchoredPosition`.

```
AnchoredPosition.__eq__(other)
```

4.104.2 AnchoredPosition.__ne__

Метод `__ne__` используется для определения неэквивалентности двух объектов типа `AnchoredPosition`.

```
AnchoredPosition.__ne__(other)
```

4.105 Класс Frame

Представляет прямоугольную область встроенного объекта. Класс предназначен для изменения положения и геометрии встроенного объекта, управления расположением и переносом текста вокруг объекта.

```
class Frame(object)
```

Список методов:

```
def setPosition(position)
def getPosition()
def setDimensions(size)
def getDimensions()
def setWrapType(type)
def getWrapType()
```

4.105.1 InlineFrame.setPosition

Метод задает положение встроенного объекта. Для текстовых документов позиция может быть установлена только для встроенных объектов, тип переноса текста которых не соответствует `TextWrapType_Inline`.

```
InlineFrame.setPosition(position)
```

Параметры:

- `position` – положение встроенного объекта `AnchoredPosition`.

4.105.2 InlineFrame.getPosition

Метод возвращает позицию встроенного объекта на странице `AnchoredPosition`. Если позиция не определена, то возвращается `none`.

```
InlineFrame.getPosition()
```

4.105.3 InlineFrame.setDimensions

Метод задает размеры (изменяет размер) встроенного объекта. Размеры считаются неопределенными, если их значение равно `none`.

```
InlineFrame.setDimensions(size)
```

Параметры:

- `size` – размеры встроенного объекта `SizeU`.

4.105.4 InlineFrame.getDimensions

Возвращает размеры встроенного объекта `SizeU` или `none`, если размеры не определены.

```
InlineFrame.getDimensions()
```

4.105.5 InlineFrame.setWrapType

Устанавливает вариант обтекания текстом встроенного объекта.

```
InlineFrame.setWrapType(type)
```

Параметр:

- `type` – вариант обтекания текстом встроенного объекта `TextWrapType`.

4.105.6 InlineFrame.getWrapType

Возвращает вариант обтекания текстом встроенного объекта `TextWrapType`.

```
InlineFrame.getWrapType()
```

4.106 Класс Image

Представляет собой изображение, как встроенный объект.

```
class Image(object)
```

Список методов:

```
def getFrame()
```

4.106.1 Image.getFrame

Метод возвращает прямоугольную область встроенного объекта `InlineFrame`.

```
Image.getFrame()
```

4.107 Класс Images

Предоставляет интерфейс для доступа к коллекции изображений.

```
class Images(object)
```

Список методов:

```
def getEnumerator()
```

4.107.1 Images.getEnumerator

Возвращает коллекцию объектов `Image`.

```
Images.getEnumerator()
```

4.108 Класс InlineObject

Представляет собой встроенный объект, который позиционируется как символ в строке текста.

```
class MediaObject(object)
```

Список методов:

```
def toImage()
def remove()
```

4.108.1 MediaObject.toImage

Метод возвращает изображение `Image`, если встроенный объект является экземпляром класса `Image`. В случае ошибки возвращается `none`.

```
MediaObject.toImage()
```

4.109 Класс InlineObjects

Предоставляет интерфейс для доступа к коллекции встроенных объектов.

```
class MediaObjects(object)
```

Список методов:

```
def getEnumerator()
```

4.109.1 MediaObjects.getEnumerator

Возвращает коллекцию классов MediaObject.

```
MediaObjects.getEnumerator()
```

4.110 Класс ColorTransforms

Класс ColorTransforms предназначен для конвертации цвета и содержит метод apply, конвертирующий цвет.

```
class ColorTransforms(object)
```

Список методов:

```
def apply(colorRGBA)
```

4.110.1 ColorTransforms.apply

Метод apply конвертирует цвет ColorRGBA в соответствии с алгоритмом, определенным реализацией, и возвращает модифицированный цвет ColorRGBA.

```
ColorTransforms.apply(colorRGBA)
```

4.111 Класс Color

Представляет либо цветовой объект RGBA, либо заданные цвета идентификатора темы.

```
class Color(object)
```

Список методов:

```
def getRGBAColor()
def getThemeColorID()
def setTransforms(transforms)
def getTransforms()

def __eq__(other)
def __ne__(other)
```

4.111.1 Color.getRGBAColor

Метод возвращает цвет ColorRGBA.

```
Color.getRGBAColor()
```

4.111.2 Color.getThemeColorID

Метод возвращает цвет идентификатора темы ThemeColorID.

```
Color.getThemeColorID()
```

4.111.3 Color.setTransforms

Метод устанавливает правило трансформации цвета, реализованное объектом ColorTransforms.

```
Color.setTransforms(transforms)
```

4.111.4 Color.getTransforms

Метод возвращает объект ColorTransforms, реализующий трансформацию цвета.

```
Color.getTransforms()
```

4.111.5 Color.__eq__

Метод __eq__ используется для определения эквивалентности двух объектов типа Color.

```
Color.__eq__(other)
```

4.111.6 Color.__ne__

Метод __ne__ используется для определения неэквивалентности двух объектов типа Color.

```
Color.__ne__(other)
```

4.112 Исключения

4.112.1 Класс BaseError

Класс BaseError является базовым классом для всех исключений SDK.

```
class BaseError(Exception)
```

4.112.2 Класс ApplicationCreateError

Исключение ApplicationCreateError вызывается в случае, когда объект Application не может быть создан.

```
class ApplicationCreateError(BaseError)
```

4.112.3 Класс IncorrectArgumentError

Исключение IncorrectArgumentError вызывается в случае, когда один из аргументов метода или функции имеет недействительное значение.

```
class IncorrectArgumentError(BaseError)
```

4.112.4 Класс InvalidObjectError

Исключение `InvalidObjectError` вызывается в случае, когда объект больше не может быть использован.

```
class InvalidObjectError(BaseError)
```

4.112.5 Класс DocumentCreateError

Исключение `DocumentCreateError` вызывается в случае, когда документ не может быть создан.

```
class DocumentCreateError(BaseError)
```

4.112.6 Класс DocumentLoadError

Исключение `DocumentLoadError` вызывается в случае, когда документ не может быть загружен.

```
class DocumentLoadError(BaseError)
```

4.112.7 Класс DocumentSaveError

Исключение `DocumentSaveError` вызывается в случае, когда документ не может быть сохранен.

```
class DocumentSaveError(BaseError)
```

4.112.8 Класс DocumentExportError

Исключение `DocumentExportError` вызывается в случае, когда документ не может быть экспортирован.

```
class DocumentExportError(BaseError)
```

4.112.9 Класс NoSuchElementError

Исключение `NoSuchElementError` вызывается в случае, когда элемент не существует.

```
class NoSuchElementError(BaseError)
```

4.112.10 Класс NotImplementedError

Исключение `NotImplementedError` вызывается в случае, если обнаружена нереализованная функциональность.

```
class NotImplementedError(BaseError)
```

4.112.11 Класс OutOfRangeError

Исключение `OutOfRangeError` вызывается в случае обнаружения выхода значения за пределы диапазона.

```
class OutOfRangeError(BaseError)
```

4.112.12 Класс ParseError

Исключение `ParseError` вызывается в случае ошибки синтаксического разбора текста.

```
class ParseError(BaseError)
```

4.112.13 Класс UnknownError

Исключение `UnknownError` вызывается в случае, когда критическое исключение возникло по неизвестной причине. Приложение должно быть завершено, поскольку возникло неопределенное состояние ядра Document API.

```
class UnknownError(BaseError)
```

4.112.14 Класс ForbiddenActionError

Исключение `ForbiddenActionError` вызывается в случае выполнения запрещенной операции.

```
class ForbiddenActionError(BaseError)
```

4.112.15 Класс DocumentModificationError

Исключение `DocumentModificationError` вызывается, когда невозможно выполнить операцию по изменению документа.

```
class DocumentModificationError(BaseError)
```

4.112.16 Класс PivotTableError

Исключение `PivotTableError` вызывается в случае ошибки при работе со сводными таблицами. Например, использование фильтра, который не может быть применен к сводной таблице.

```
class PivotTableError(BaseError)
```

4.112.17 Класс PositionDocumentMismatchError

Исключение `PositionDocumentsMismatchError` вызывается в случае, когда несколько позиций относятся к различным документам и не могут быть использованы в одной операции. Например, при попытке пользователя создать диапазон `Range`,

включающий позиции `Position`, принадлежащие нескольким различным документам, и выполнить операцию для такого диапазона.

```
class PositionDocumentMismatchError(BaseError)
```

4.112.18 Класс ScriptExecutionError

Исключение `ScriptExecutionError` вызывается в случае, когда сценарий не удается выполнить.

```
class ScriptExecutionError(BaseError)
```

5 ВЕРСИИ DOCUMENT API

5.1 Механизм контроля версий

Константы версии Document API `Major` и `Minor` позволяют проверить совместимость предыдущей и текущей версии Document API.

Если была изменена константа `Major` версии Document API, т. е. в Document API произошли обратно несовместимые изменения, то программный код должен быть пересмотрен и обновлен. Обратно несовместимыми изменениями считаются: переименование, удаление или несовместимое изменение подписи существующих классов или методов, а также добавление новых методов, типов и членов класса.

В разделе Изменения указывается появились ли несовместимые изменения в Document API.

Если была изменена константа `Minor` версии Document API, то в Document API произошли только обратно совместимые изменения, и нет необходимости менять программный код, чтобы он работал с более новой версией Document API. Но гарантируется совместимость только на уровне исходного кода, поэтому необходимо перекомпилировать программный код приложения с более новой версией библиотеки Document API.

Рекомендуется проверить версию Document API до инициализации, как указано ниже:

```
if __name__ == '__main__':
    expected_major_api_version = 1
    expected_minor_api_version = 0

    if not DocumentAPI.isAPIVersionCompatible(expected_major_api_version,
expected_minor_api_version):

        # Вывод сообщения о серьезной ошибке несовместимости версии библиотеки
Document API и выход из программы

        pass

    # Работа с библиотекой Document API (создание объекта Application и т. д.)
```

5.2 Изменения

Обратно несовместимые изменения в Document API отсутствуют.