МойОфис®

Комплект Средств Разработки (SDK)

Руководство программиста

MYOFFICE DOCUMENT API (Python)

ООО «НОВЫЕ ОБЛАЧНЫЕ ТЕХНОЛОГИИ»

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

«МОЙОФИС КОМПЛЕКТ СРЕДСТВ РАЗРАБОТКИ (SDK)»

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

РУКОВОДСТВО ПРОГРАММИСТА

2020.02

На 145 листах

Москва

2020

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
SDK	Software Development Kit

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Назначение программы

Программное обеспечение «МойОфис Комплект Средств Разработки (SDK)» это набор средств взаимодействия прикладных систем с приложениями МойОфис.

Библиотека MyOffice Document API для языка программирования Python, предназначена для разработки решений по автоматизации бизнес-процессов с использованием приложений (редакторов) МойОфис.

Функции «МойОфис Комплект Средств Разработки (SDK)» описаны в документе «МойОфис Комплект Средств Разработки (SDK). Функциональные возможности».

1.2 Библиотека MyOffice Document API для языка программирования Python

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

1. обработка электронных текстовых и табличных документов;
2. поддержка форматов файлов DOCX, XLSX, ODT, ODS, XODT, XODS;
3. экспорт документов в формат PDF/A-1;
4. добавление, удаление, изменение текста абзаца;
5. вставка, удаление, форматирование таблиц в текстовом документе;
6. вставка, удаление, переименование отдельных листов в табличном документе;
7. работа с ячейками электронной таблицы, установка значений, расчет формул;
8. форматирование документов с использованием различных шрифтов и цветового оформления, управление разделами, настройка ориентации страниц, управление колонтитулами документа;
9. поиск и замена фрагмента текста в документе;
10. управление закладками в текстовом документе;
11. работа с макрокомандами.

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

Библиотека MyOffice Document API для языка программирования Python предоставляет широкие возможности управления текстовыми и табличными документами в разрабатываемых приложениях и упрощает решение задачи интеграции платформы МойОфис с прикладными информационными системами.

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_2020.02_x64.zip
Linux	MyOffice_SDK_Document_API_Python_Linux_2020.02_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_2020.02_x64.zip
4. установить программный пакет MyOffice Document API с помощью системы управления пакетами, например:
   pip install MyOfficeSDKDocumentAPI-2020.02-cp27-cp27m-win_amd64.whl

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_2020.02_x64.zip
4. установить программный пакет MyOffice Document API с помощью системы управления пакетами, например:
   pip install MyOfficeSDKDocumentAPI-2020.02-cp27-cp27mu-linux_x86_64.whl

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 передачей файла, содержащего исходный код приложения.

Для запуска разработанного приложения на компьютере пользователя должны присутствовать:

1. интерпретатор Python, версии не ниже 2.7.14;
2. 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 указаны свойства создаваемого документа (см. раздел 4.59.1).

Сохранение документа осуществляется с помощью метода `saveAs`, содержащегося в объектной модели документа (см. раздел 4.51.1).

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
document = application.createDocument(sdk.DocumentType_Text)

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

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


3.1.1 Работа с текстом

3.1.1.1 Настройка свойств текста

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

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
document = application.createDocument(sdk.DocumentType_Text)


# 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")
```


3.1.2 Работа с таблицами

3.1.2.1 Вставка таблицы

Для вставки таблицы необходимо задать ее положение в документе. В примере таблица вставляется в начало документа. В случае успешного выполнения вставки возвращается уникальный идентификатор вставленной таблицы – `tableID`.

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
document = application.createDocument(sdk.DocumentType_Text)

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

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

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

3.1.2.2 Удаление таблицы

Для удаления таблицы необходим объект таблицы, который может быть получен по идентификатору `tableID`. Необходимо обратить внимание на то, что объект таблицы перестает существовать после удаления из документа.

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
document = application.createDocument(sdk.DocumentType_Text)

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

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

# Удалить таблицу
table = document.getBlocks().getTable(tableID)
table.remove()

# Сохранить документ 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.

Пример:

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

3.1.3 Работа с ячейками таблицы

3.1.3.1 Объединение ячеек таблицы

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

Пример:

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

3.1.3.2 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 эта ячейка является.

Пример:

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

3.1.4 Форматирование таблицы

3.1.4.1 Изменение ширины столбца таблицы

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

Пример:

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


3.1.4.2 Установка свойств форматирования ячеек таблицы

Установка свойств форматирования ячеек таблицы и значений ячейки.

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
document = application.createDocument(sdk.DocumentType_Text)

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

rowsCount     = 3
columnsCount  = 3
tableName     = "MyTable"
tableID = 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

tableEmployes = document.getBlocks().getTable(tableID)
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)

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

3.1.4.3 Установка границ ячеек таблицы

Установка внешних и внутренних границ ячеек таблицы.

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
document = application.createDocument(sdk.DocumentType_Text)

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

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

tableEmployees = document.getBlocks().getTable(tableID)

# 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))

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


3.1.5 Работа с закладками

3.1.5.1 Вставка закладки

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

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
document = application.createDocument(sdk.DocumentType_Text)

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

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

3.1.5.2 Изменение содержимого закладки

В документе можно изменить содержимое закладки.

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
document = application.createDocument(sdk.DocumentType_Text)

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

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


3.1.5.3 Удаление закладки

Из документа можно удалить закладку.

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
document = application.createDocument(sdk.DocumentType_Text)

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

# Удаление закладки
document.removeBookmark(bmkName)

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


3.1.6 Экспорт текстового документа

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

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
document = application.createDocument(sdk.DocumentType_Text)

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

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


3.1.7 Поиск в текстовом документе

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

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
document = application.createDocument(sdk.DocumentType_Text)

# 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)
collect = textSearch.findText("API")

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


3.1.8 Управление ориентацией и свойствами страниц раздела

Для управления ориентацией и свойствами страниц раздела, требуется выбрать раздел.

Выбор раздел возможен с использованием метода `getSectionsEnumerator`.

Пример:

Выбор первого раздела документа.

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

Выбор раздела возможен из блока.

Пример:

Выбор раздела в первом блоке документа.

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

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

Пример:

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.

Пример:

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

Получение свойств страниц раздела.

Пример:

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

Установка альбомной ориентации для раздела.

Пример:

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


Получение ориентации страниц раздела.

Пример:

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

Установка книжной ориентации для всего документа.

Пример:

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

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

Пример:

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


3.1.9 Работа с колонтитулами раздела

Получение верхнего и нижнего колонтитулов первого раздела документа.

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

try:
    application = sdk.Application()
    #Загрузка текстового документа
    document = application.loadDocument("BasicExample.docx")

    #Получаем коллекцию разделов
    section = document.getSectionsEnumerator()

    for sec in section:
        headers = sec.getHeaders()
        footer = sec.getFooters()
        for h in headers:
            #Верхний колонтитул первого раздела
            print (h.getRange().extractText())
        for f in footer:
            #Нижний колонтитул первого раздела
            print (f.getRange().extractText())

    print('OK')

except:
    print ("Error")
```

Получение типов колонтитулов разделов документа.

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

try:
    application = sdk.Application()
    #Загрузка текстового документа
    document = application.loadDocument("BasicExample.docx")

    #Получаем коллекцию разделов
    section = document.getSectionsEnumerator()
    sdk.HeaderFooterType_Footer
    for sec in section:
        headers = sec.getHeaders()
        for h in headers:
            #Получаем тип колонтитула
            print (h.getType())

    print('OK')

except:
    print ("Error")
```


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. раздел 4.59.1).

Сохранение документа осуществляется с помощью метода `saveAs`, содержащегося в объектной модели документа (см. раздел 4.51.1).

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

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

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

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

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

except:
    print ("Error")
```


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.

Пример:

```
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.

Пример:

```
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.

Пример:

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

3.2.3.2 Удаление рабочего листа табличного документа

Для удаления таблицы необходим объект таблицы, который может быть получен из объекта `tableID`. Следует обратить внимание на то, что объект таблицы перестает существовать после удаления из документа.

Пример:

```
table = document.getBlocks().getTable(table_id)
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.

Пример:

```
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.

Пример:

```
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.

Пример:

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

Пример программного кода установки выравнивания текста абзаца по ширине ячейки:

```
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"
    tableID = position.insertTable(rowsCount, columnsCount, sheetName)

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

    #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")
    print('OK')

except:
    print ("Error")
```


3.2.4.2 Объединение ячеек таблицы

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

Пример:

```
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 эта ячейка является.

Пример:

```
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"
    tableID = position.insertTable(rowsCount, columnsCount, sheetName)

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

    #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 ширин указанного столбца. Следующий пример демонстрирует настройку ширины второго столбца таблицы до 300 единиц.

Пример:

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


3.2.6 Экспорт табличного документа

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

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
document = application.createDocument(sdk.DocumentType_Workbook)

tbl_id = document.getRange().getBegin().insertTable(40,40, "Text Search")
table = document.getBlocks().getTable(tbl_id)

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

document.saveAs("wbk.xlsx")

# Экспорт в формат PDF
outputFile = "BasicExample.pdf"
document.exportTo(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`.

Пример:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()
document = application.createDocument(sdk.DocumentType_Workbook)

tbl_id = document.getRange().getBegin().insertTable(40,40, "Text Search")
table = document.getBlocks().getTable(tbl_id)

# 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")

# Установить свойства текста для найденных фрагментов
for r in collect:
    r.setTextProperties(textProperties)

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


4 СПРАВОЧНИК КЛАССОВ, СТРУКТУР И МЕТОДОВ

4.1 Поддерживаемые типы документов

Поддерживаемые типы документов представлены в Таблица 3.

Таблица 3 – Типы документов

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

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

4.2 Поддерживаемые форматы документов

Поддерживаемые форматы документов представлены в Таблица 4.

Таблица 4 – Форматы документов

Формат документа	Использование	Имя константы формата документа
PlainText	Используется для работы с файлами TXT.	`DocumentFormat_PlainText`
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_DSV`
OXML	Используется для работы с текстовыми (DOCX) или табличными (XLSX) документами в формате Open Office XML	`DocumentFormat_OXML`
ODF	Используется для работы с текстовыми (ODT) или табличными (ODS) документами формата Open Document Format (ГОСТ Р ИСО/МЭК 26300-2010)	`DocumentFormat_ODF`
HTML	Используется для работы с веб-документами в формате HTML. Работа с веб-документами в формате HTML средствами Document API в настоящий момент не поддерживается	`DocumentFormat_HTML`
PDF	Используется для работы с документами в формате Portable Document Format (PDF), версии 1.4	`DocumentFormat_PDF`
PDFA	Используется для работы с документами в формате Portable Document Format (PDF) для долгосрочного архивного хранения (PDF/A-1b)	`DocumentFormat_PDFA`

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 Таблица 5.

Таблица 5 – Форматы экспорта документов

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

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

4.4 Системы адресации ячеек

Поддерживаемые системы адресации ячеек (стили ссылок) в табличном документе представлены в Таблица 6.

Таблица 6 – Системы адресации ячеек в табличном документе

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

4.5 Кодировки документов

Поддерживаемые кодировки документов представлены в Таблица 7.

Таблица 7 – Кодировки документов

Кодировка	Имя константы кодировки документа
Невозможно определить кодировку	`Encoding_Unknown`
UTF8	`Encoding_UTF8`
UTF16BE	`Encoding_UTF16BE`
UTF16LE	`Encoding_UTF16LE`
UTF32BE	`Encoding_UTF32BE`
UTF32LE	`Encoding_UTF32LE`
Windows1250	`Encoding_Windows1250`
Windows1251	`Encoding_Windows1251`
Windows1252	`Encoding_Windows1252`
ISO8859Part5	`Encoding_ISO8859Part5`
KOI8R	`Encoding_KOI8R`
KOI8U	`Encoding_KOI8U`
CP866	`Encoding_CP866`

4.6 Типы выравнивания текста

Поддерживаемые типы выравнивания текста представлены в Таблица 8.

Таблица 8 – Типы выравнивания текста

Описание	Имя константы типа выравнивания текста
Выравнивание текста по умолчанию	`Alignment_Default`
Выравнивание текста по левому краю	`Alignment_Left`
Выравнивание текста по центру	`Alignment_Center`
Выравнивание по правому краю	`Alignment_Right`
Выравнивание по ширине	`Alignment_Justify`
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_Distributed`
Распределение текста по горизонтали – заполнение строки текстом	`Alignment_Fill`

4.7 Типы выравнивания текста, вертикально расположенного в ячейке таблицы

Поддерживаемые типы выравнивания текста, вертикально расположенного в ячейках таблицы, представлены в Таблица 9.

Таблица 9 – Типы вертикального выравнивания

Описание	Имя константы типа выравнивания текста, расположенного вертикально
Выравнивание текста, расположенного в ячейке вертикально, по правой границе ячейки	`VerticalAlignment_Bottom`
Выравнивание текста, расположенного в ячейке вертикально, по центру, симметрично относительно левой и правой границы ячейки	`VerticalAlignment_Center`
Выравнивание текста, расположенного в ячейке вертикально, по левой границе ячейки	`VerticalAlignment_Top`

4.8 Типы отображения длинного текста в ячейках таблиц

Поддерживаемые типы отображения длинного текста в ячейках таблиц представлены в Таблица 10.

Таблица 10 – Типы отображения длинного текста в ячейках таблиц

Описание типа отображения	Имя константы типа отображения длинного текста
Отображение длинного текста в одну строку с наложением на соседние ячейки	`TextLayout_SingleLine`
Перенос данных внутри ячейки по словам	`TextLayout_WrapByWords`
Текст или число отображаются так, чтобы содержимое поместилось в ячейке полностью	`TextLayout_ShrinkSizeToFitWidth`

4.9 Типы линий

Поддерживаемые типы линий представлены в Таблица 11.

Таблица 11 – Типы линий

Наименование типа линии	Имя константы типа линии
Нет границ	`NoLineProperties`
Обычная линия	`LineStyle_Solid`
Пунктирная линия	`LineStyle_Dot`
Штриховая линия	`LineStyle_Dash`
Штрихпунктирная линия	`LineStyle_DashDot`
Двойная линия	`LineStyle_Double`

4.10 Типы надстрочного/подстрочного форматирования

Поддерживаемые типы надстрочного/подстрочного форматирования представлены в Таблица 12.

Таблица 12 – Типы надстрочного/подстрочного форматирования

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

4.11 Типы форматов ячеек

Поддерживаемые форматы ячеек представлены в Таблица 13.

Таблица 13 – Форматы ячеек

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

4.12 Типы межстрочного интервала

Поддерживаемые типы межстрочного интервала представлены в Таблица 14.

Таблица 14 – Типы межстрочного интервала

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

4.13 Типы схем абзацев

Типы схем абзацев представлены в Таблица 15.

Таблица 15 – Типы типов схем абзацев

Наименование типа схемы абзаца	Имя константы типа схемы абзаца	Изображение
Начать список или нумерацию	`ListSchema_EnumFirst`	
Неизвестно	`ListSchema_Unknown`	
Список без маркера	`ListSchema_UnknownBullet`	Соответствует варианту «нет»
Нумерация без номера	`ListSchema_UnknownNumbering`	Соответствует варианту «нет»
Список с маркерами в виде круга	`ListSchema_BulletCircleSolid`	
Список с маркерами в виде окружности	`ListSchema_BulletCircleContour`	
Список с маркерами в виде квадрата	`ListSchema_BulletSquareSolid`	
Список с маркерами в виде четырех ромбов	`ListSchema_BulletDiamondDots`	
Список с маркерами в виде дефиса	`ListSchema_BulletHyphen`	
Список с маркерами в виде вогнутой стрелки	`ListSchema_BulletConcaveArrowSolid`	
Список с маркерами в виде галочки	`ListSchema_BulletCheckmark`	

Наименование типа схемы абзаца	Имя константы типа схемы абзаца	Изображение
Десятичная нумерация с точкой	`ListSchema_EnumeratorDecimalDot`	
Многоуровневая десятичная нумерация с точкой	`ListSchema_EnumeratorDecimalDotMultiLevel`	
Десятичная нумерация со скобкой	`ListSchema_EnumeratorDecimalBracket`	
Нумерация латинскими прописными буквами с точкой	`ListSchema_EnumeratorLatinUppercaseDot`	
Нумерация латинскими строчными буквами с точкой	`ListSchema_EnumeratorLatinLowercaseDot`	
Нумерация латинскими строчными буквами со скобкой	`ListSchema_EnumeratorLatinLowercaseBracket`	
Нумерация римскими прописными цифрами с точкой	`ListSchema_EnumeratorRomanUppercaseDot`	
Нумерация римскими строчными цифрами с точкой	`ListSchema_EnumeratorRomanLowercaseDot`	
Десятичная нумерация через запятую со скобкой	`ListSchema_EnumeratorDecimalRussianBracket`	
Нумерация с русскими строчными буквами со скобкой	`ListSchema_EnumeratorRussianLowercaseBracket`	

Наименование типа схемы абзаца	Имя константы типа схемы абзаца	Изображение
Завершить список или нумерацию	`ListSchema_EnumLast`	

4.14 Типы вариантов размещения знака валюты

Типы вариантов размещения знака валюты представлены в Таблица 16.

Таблица 16 – Типы вариантов размещения знака валюты

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

4.15 Типы ориентации страницы

Поддерживаемые типы ориентации страницы представлены в Таблица 17.

Таблица 17 – Типы ориентации страницы

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

4.16 Типы колонтитулов страниц документов

Поддерживаемые типы колонтитулов страниц документов представлены в **Таблица 18**.

Таблица 18 – Типы колонтитулов страниц документа

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

4.17 Класс ColorRGBA

Объявление класса:

```
class ColorRGBA(_object)
```

Подробное описание:

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.

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)
```

4.17.1 ColorRGBA.__eq__

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

Параметры:

`other - ColorRGBA` (см. раздел 4.17).

Возвращаемый тип: `bool`

Описание метода:

Позволяет использовать оператор **`__eq__`** для определения эквивалентности значений двух цветовых моделей, включая значение прозрачности.

4.18 Класс Rect

Объявление класса:

```
class Rect(_object)
```

Подробное описание:

Описывает прямоугольник в двухмерном пространстве.

Список свойств:

`x1` – положение крайней левой точки прямоугольника по горизонтали;

`y1` – положение крайней левой точки прямоугольника по вертикали;

`x2` – положение крайней правой точки прямоугольника по горизонтали;

`y2` – положение крайней правой точки прямоугольника по вертикали.

Список методов:

```
def __eq__(other)
```

4.18.1 Rect.__eq__

```
Rect.__eq__(other)
```

Параметры:

`other - Rect` (см. раздел 4.18).

Возвращаемый тип: `bool`

Описание метода:

Позволяет использовать оператор **__eq__** для определения эквивалентности положений прямоугольников в двухмерном пространстве.

4.19 Класс UserInfo

Список свойств:

`name` – наименование пользователя;

`email` – адрес электронной почты пользователя.

Список методов:

```
def __eq__(other)
```

Подробное описание:

Предоставляет информацию о пользователе.

Объявление класса:

```
class UserInfo(_object)
```

4.19.1 UserInfo.__eq__

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

Параметры:

`other` – `UserInfo` (см. раздел 4.19).

Возвращаемый тип: `bool`

Описание метода:

Позволяет использовать оператор **`__eq__`** для определения эквивалентности информации о двух пользователях.

4.20 Класс DateTime

Объявление класса:

```
class DateTime(_object)
```

Подробное описание:

Предоставляет дату и время с точностью до секунды.

Список свойств:

`year` – год;

`month` – месяц;

`day` – день;

`hour` – часы;

`minute` – минуты;

`second` – секунды.

Список методов:

```
def __eq__(other)
```

4.20.1 DateTime.__eq__

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

Параметры:

`other` – `DateTime` (см. раздел 4.20).

Возвращаемый тип: `bool`

Описание метода:

Позволяет использовать оператор **`__eq__`** для определения эквивалентности двух значений времени.

4.21 Класс PageProperties

Объявление класса:

```
class PageProperties(_object)
```

Подробное описание:

Содержит размеры страницы.

Список свойств:

`height` – высота страницы;

`width` – ширина страницы.

4.22 Класс Comment

Список методов:

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

Подробное описание:

Предоставляет доступ к свойствам комментария примененного к области текста.

Объявление класса:

```
class Comment(_object)
```

4.22.1 Comment.getRange

```
Comment.getRange()
```

Возвращаемый тип: `Range` (см. раздел 4.28).

Описание метода:

Возвращает объект `Range`, соответствующего объекта `Comment`.

4.22.2 Comment.getText

```
Comment.getText()
```

Возвращаемый тип: `str`

Описание метода:

Возвращает текст комментария.

4.22.3 Comment.getAudioUrl

```
Comment.getAudioUrl()
```

Возвращаемый тип: `str`

Описание метода:

Возвращает путь к файлу аудиокомментария.

4.22.4 Comment.getInfo

```
Comment.getInfo()
```

Возвращаемый тип: `TrackedChangeInfo` (см. раздел 4.46).

Описание метода:

Предоставляет доступ к отслеживаемой информации об изменении комментария (автор изменения, дата и т. д.).

4.23 Класс Comments

Объявление класса:

```
class Comments(_object)
```

Подробное описание:

Предоставляет интерфейс для доступа к коллекции комментариев диапазона.

Пример использования:

```
from MyOfficeSDKDocumentAPI import DocumentAPI as sdk

application = sdk.Application()

inputFile = "TextWithComments.docx"
document = application.loadDocument(inputFile)

comments = document.getComments()
for c in comments:
    dt = '{}:{} {}/{}/{}'.format(c.getInfo().timeStamp.hour,
                                 c.getInfo().timeStamp.minute,
                                 c.getInfo().timeStamp.day,
                                 c.getInfo().timeStamp.month,
                                 c.getInfo().timeStamp.year)
    trc = 'Пользователь {} ' \
            'добавил {} ' \
            'комментарий \"{}\"'.format(c.getInfo().author.name,
                                       dt,
                                       str(c.getText()).rstrip())
    print(trc)
```

4.24 Класс Position

Объявление класса:

```
class Position(_object)
```

Подробное описание:

Представляет местоположение в документе при вставке нового объекта.

Список методов:

```
def insertText(text)
def insertTable(rowsCount, columnsCount, name)
def insertPageBreak()
def insertLineBreak()
def insertBookmark(name)
def insertImage(url, size)
def insertSectionBreak()
def __eq__(other)
```

4.24.1 Position.insertText

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

Параметры:

`text` – вставляемый текст.

Описание метода:

Вставляет текст в указанную позицию.

4.24.2 Position.insertTable

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

Параметры:

`rowsCount` – количество строк в таблице;

`columnsCount` – количество столбцов в таблице;

`name` – наименование таблицы.

Возвращаемый тип: `ID` – уникальный идентификатор таблицы (см. раздел 4.47).

Описание метода:

Вставляет таблицу в указанную позицию.

4.24.3 Position.insertPageBreak

```
Position.insertPageBreak()
```

Описание метода:

Вставляет разделитель страниц в указанную позицию.

4.24.4 Position.insertLineBreak

```
Position.insertLineBreak()
```

Описание метода:

Вставляет переход на следующую строку, не разрывающий абзац, в указанную позицию.

4.24.5 Position.insertBookmark

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

Параметры:

`name` – наименование закладки.

Описание метода:

Вставляет закладку с наименованием `name` в текущую позицию.

4.24.6 Position.insertImage

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

Параметры:

`url` – полный путь к файлу рисунка;

`size` – геометрические размеры изображения для вставки.

Описание метода:

Вставляет рисунок, размещенный в файле в текущую позицию. С помощью параметра `url` задается полный путь к файлу. Параметр `size`, задает геометрические размеры изображения для вставки.

4.24.7 Position.insertSectionBreak

```
Position.insertSectionBreak()
```

Описание метода:

Вставляет разрыв раздела в указанную позицию.

4.24.8 Position.__eq__

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

Параметры:

`other` – `Position` (см. раздел 4.24).

Возвращаемый тип: `bool`

Описание метода:

Позволяет использовать оператор **`__eq__`** для определения эквивалентности значений двух местоположений в документе.

4.25 Класс Block

Объявление класса:

```
class Block(_object)
```

Подробное описание:

Является базовым классом для всех блоков документа.

Список методов:

```
def toParagraph()
def toTable()
def toShape()
def getRange()
def remove()
def getSection()
```

4.25.1 Block.toParagraph

```
Block.toParagraph()
```

Возвращаемый тип: `Paragraph` (см. раздел 4.26).

Описание метода:

Преобразует блок в абзац.

4.25.2 Block.toTable

```
Block.toTable()
```

Возвращаемый тип: `Table` (см. раздел 4.48).

Описание метода:

Преобразует блок в таблицу.

4.25.3 Block.getRange

```
Block.getRange()
```

Возвращаемый тип: `Range` (см. раздел 4.28).

Описание метода:

Возвращает диапазон, в котором содержится данный блок.

4.25.4 Block.remove

```
Block.remove()
```

Описание метода:

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

4.25.5 Block.getSection

```
Block.getSection
```

Возвращаемый тип: `Section` (см. раздел 4.29).

Описание метода:

Возвращает раздел, содержащийся в блоке.

4.26 Класс Paragraph

Объявление класса:

```
class Paragraph(Block)
```

Родительский класс:

`Block` (см. раздел 4.22).

Подробное описание:

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

Список методов:

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

4.26.1 Paragraph.getParagraphProperties

```
Paragraph.getParagraphProperties()
```

Возвращаемый тип: `ParagraphProperties` (см. раздел 4.37).

Описание метода:

Позволяет получать свойства абзаца (выравнивание, отступ, интервалы).

4.26.2 Paragraph.setParagraphProperties

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

Параметры:

`properties – ParagraphProperties` (см. раздел 4.37).

Описание метода:

Позволяет устанавливать свойства абзаца (выравнивание, отступ, интервалы).

4.26.3 Paragraph.getListSchema

```
Paragraph.getListSchema()
```

Возвращаемый тип: (см. раздел 4.13).

Описание метода:

Возвращает схему маркированного или нумерованного списка.

4.26.4 Paragraph.setListSchema

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

Параметры:

`schema` – устанавливаемая схема (см. раздел 4.13).

Описание метода:

Устанавливает схему маркированного или нумерованного списка.

4.26.5 Paragraph.getListLevel

```
Paragraph.getListLevel()
```

Возвращаемый тип: `long`

Описание метода:

Возвращает глубину вложенности элемента списка.

4.26.6 Paragraph.setListLevel

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

Параметры:

`level` – уровень вложенности элемента списка.

Описание метода:

Устанавливает глубину вложенности элемента списка.

4.26.7 Paragraph.increaseListLevel

```
Paragraph.increaseListLevel()
```

Описание метода:

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

4.26.8 Paragraph.decreaseListLevel

```
Paragraph.decreaseListLevel()
```

Описание метода:

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

4.27 Класс Paragraphs

Объявление класса:

```
class Paragraphs(_object)
```

Подробное описание:

Класс **Paragraphs** предоставляет доступ к коллекции абзацев.

Список методов:

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

4.27.1 Paragraphs.setListSchema

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

Параметры:

`schema` (см. раздел 4.13).

Описание метода:

Устанавливает тип маркированного или нумерованного списка (см. раздел 4.13).

4.27.2 Paragraphs.setListLevel

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

Параметры:

`level` - глубина вложенности элемента списка.

Описание метода:

Устанавливает глубину вложенности элемента списка.

4.27.3 Paragraphs.getEnumerator

```
Paragraphs.getEnumerator()
```

Описание метода:

Возвращает коллекцию классов `Paragraph`.

4.28 Класс Range

Объявление класса:

```
class Range(_object)
```

Подробное описание:

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

Список методов:

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

4.28.1 Range.getBegin

```
Range.getBegin()
```

Возвращаемый тип: `Position` (см. раздел 4.24).

Описание метода:

Возвращает начальную позицию диапазона.

4.28.2 Range.getEnd

```
Range.getEnd()
```

Возвращаемый тип: `Position` (см. раздел 4.24).

Описание метода:

Возвращает конечную позицию диапазона.

4.28.3 Range.extractText

```
Range.extractText()
```

Возвращаемый тип: `str`

Описание метода:

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

4.28.4 Range.removeContent

```
Range.removeContent()
```

Описание метода:

Удаляет содержимое диапазона.

4.28.5 Range.replaceText

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

Параметры:

`text` – текст, которым заменятся содержимое диапазона.

Описание метода:

Заменяет содержимое диапазона на указанный текст.

4.28.6 Range.getTextProperties

```
Range.getTextProperties()
```

Возвращаемый тип: `TextProperties` (см. раздел 4.52).

Описание метода:

Возвращает свойства оформления текста в диапазоне.

4.28.7 Range.setTextProperties

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

Параметры:

`textProperties` – свойства оформления текста в диапазоне.

Описание метода:

Настройка свойств оформления текста.

4.28.8 Range.getBlocksEnumerator

```
Range.getBlocksEnumerator()
```

Описание метода:

Возвращает коллекцию классов `Block` в диапазоне.

4.28.9 Range.getTrackedChangesEnumerator

```
Range.getTrackedChangesEnumerator()
```

Описание метода:

Возвращает коллекцию отслеживаемых изменений в диапазоне.

4.28.10 Range.getComments

```
Range.getComments()
```

Возвращаемый тип: `Comments` (см. раздел 4.23).

Описание метода:

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

4.28.11 Range.getParagraphs

```
Range.getParagraphs()
```

Возвращаемый тип: `Paragraphs` (см. раздел 4.27).

Описание метода:

Обеспечивает доступ к абзацам в диапазоне.

4.29 Класс Section

Объявление класса:

```
class Section(_object)
```

Подробное описание:

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

Список методов:

```
def setPageProperties(pageProperties)

def PageProperties getPageProperties()

def setPageOrientation(pageOrientation)

def getPageOrientation()

def getRange()

def getHeaders()

def getFooters()
```

4.29.1 Section.setPageProperties

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

Параметры:

`pageProperties` – свойства страниц в разделе (см. раздел 4.21).

Описание метода:

Устанавливает высоту и ширину страниц раздела.

4.29.2 Section.getPageProperties

```
Section.getPageProperties()
```

Возвращаемый тип: `PageProperties` (см. раздел 4.21).

Описание метода:

Возвращает размеры высоты и ширины для страниц раздела.

4.29.3 Section.setPageOrientation

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

Параметры:

`pageOrientation` – ориентация страниц в диапазоне (см. раздел 4.15).

Описание метода:

Задает ориентацию страниц раздела.

4.29.4 Section.getPageOrientation

```
Section.getPageOrientation()
```

Возвращаемый тип: `PageOrientation` (см. раздел 4.15).

Описание метода:

Возвращает ориентацию страниц раздела.

4.29.5 Section.getRange

```
Section.getRange()
```

Возвращаемый тип: `Range` (см. раздел 4.28).

Описание метода:

Возвращает диапазон в документе, соответствующий данному разделу.

4.29.6 Section.getHeaders

```
Section.getHeaders()
```

Возвращаемый тип: `HeadersFooters` (см. раздел 4.66).

Описание метода:

Предоставляет доступ к коллекции верхних колонтитулов, содержащихся в разделе.

4.29.7 Section.getFooters

```
Section.getFooters()
```

Возвращаемый тип: `HeadersFooters` (см. раздел 4.66).

Описание метода:

Предоставляет доступ к коллекции нижних колонтитулов, содержащихся в разделе.

4.30 Класс Search

Объявление класса:

```
class Search(_object)
```

Подробное описание:

Предоставляет интерфейс для выполнения поиска в документе.

Список методов:

```
def findText(*args)
```

4.30.1 Search.findText

```
Search.findText(*args)
```

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.

Описание метода:

Обеспечивает поиск текста в документе, без учета регистра.

4.31 Глобальный метод createSearch

Объявление глобального метода:

```
createSearch(document)
```

Параметры глобального метода:

`document` – экземпляр объекта `document`, в котором требуется произвести поиск.

Подробное описание:

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

4.32 Класс CellPosition

Объявление класса:

```
class CellPosition(_object)
```

Подробное описание:

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

Список свойств:

`row` – номер строки ячейки, нумерация начинается с 0;

`column` – номер столбца ячейки, нумерация начинается с 0.

Список методов:

```
def toString()
```

4.32.1 CellPosition.toString

```
CellPosition.toString()
```

Возвращаемый тип: `str`

Описание метода:

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

4.33 Класс CellRangePosition

Объявление класса:

```
class CellRangePosition(_object)
```

Подробное описание:

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.

Список свойств:

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

`bottomRight` – содержит позицию правой нижней ячейки таблицы прямоугольного диапазона.

Список методов:

```
def toString()
```

4.33.1 CellRangePosition. toString

```
CellRangePosition.toString()
```

Возвращаемый тип: `str`

Описание метода:

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

4.34 Класс TableRangeInfo

Объявление класса:

```
class TableRangeInfo(_object)
```

Подробное описание:

Предоставляет положение диапазона ячеек в таблице.

Список свойств:

`tableRange` – положение диапазона ячеек в таблице;

`tableID` – уникальный идентификатор таблицы.

4.35 Класс LineSpacing

Объявление класса:

```
class LineSpacing(_object)
```

Подробное описание:

Позволяет управлять межстрочным интервалом.

Список свойств:

`value` – значение межстрочного интервала;

`rule` – тип межстрочного интервала (см. раздел 4.12).

4.36 Класс LineProperties

Объявление класса:

```
class LineProperties(_object)
```

Подробное описание:

Содержит свойства оформления линий.

Удобная константа для невидимой линии (линия со стилем NoLineProperties).

Список свойств:

`style` – стиль линии (см. раздел 4.9);

`width` – ширина линии;

`color` – цвет линии.

4.37 Класс ParagraphProperties

Объявление класса:

```
class ParagraphProperties(_object)
```

Подробное описание:

Содержит свойства абзаца.

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.

4.38 Класс Borders

Объявление класса:

```
class Borders(_object)
```

Подробное описание:

Предоставляет методы для управления границами отдельной ячейки.

Список методов:

```
def getLeft()
def getRight()
def getTop()
def getBottom()
def getDiagonalDown()
def getDiagonalUp()
def getOuter()
def getDiagonals()
def setLeft(lineProperties)
def setRight(lineProperties)
def setTop(lineProperties)
def setBottom(lineProperties)
def setDiagonalDown(lineProperties)
def setDiagonalUp(lineProperties)
def setOuter(lineProperties)
def setDiagonals(lineProperties)
```

4.38.1 Borders.getLeft

```
Borders.getLeft()
```

Возвращаемый тип: `Borders` (см. раздел 4.38).

Описание метода:

Позволяет получить настройки свойств левой границы ячейки.

4.38.2 Borders.getRight

```
Borders.getRight()
```

Возвращаемый тип: `Borders` (см. раздел 4.38).

Описание метода:

Позволяет получить настройки свойств правой границы ячейки.

4.38.3 Borders.getTop

```
Borders.getTop()
```

Возвращаемый тип: `Borders` (см. раздел 4.38).

Описание метода:

Позволяет получить настройки свойств верхней границы ячейки.

4.38.4 Borders.getBottom

```
Borders.getBottom()
```

Возвращаемый тип: `Borders` (см. раздел 4.38).

Описание метода:

Позволяет получить настройки свойств нижней границы ячейки.

4.38.5 Borders.getDiagonalDown

```
Borders.getDiagonalDown()
```

Возвращаемый тип: `Borders` (см. раздел 4.38).

Описание метода:

Позволяет получить настройки свойств диагональной границы ячейки.

4.38.6 Borders.getDiagonalUp

```
Borders.getDiagonalUp()
```

Возвращаемый тип: `Borders` (см. раздел 4.38).

Описание метода:

Позволяет получить настройки свойств диагональной границы ячейки.

4.38.7 Borders.getOuter

```
Borders.getOuter()
```

Возвращаемый тип: `Borders` (см. раздел 4.38).

Описание метода:

Позволяет получить настройки свойств внешних границы ячейки.

4.38.8 Borders.getDiagonals

```
Borders.getDiagonals()
```

Возвращаемый тип: `Borders` (см. раздел 4.38).

Описание метода:

Позволяет получить настройки свойств диагональных границ ячейки.

4.38.9 Borders.setLeft

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

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить настройки свойств левой границы ячейки.

4.38.10 Borders.setRight

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

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить настройки свойств правой границы ячейки.

4.38.11 Borders.setTop

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

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить настройки свойств верхней границы ячейки.

4.38.12 Borders.setBottom

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

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить настройки свойств нижней границы ячейки.

4.38.13 Borders.setDiagonalDown

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

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить настройки свойств диагональной границы ячейки.

4.38.14 Borders.setDiagonalUp

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

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить настройки свойств диагональной границы ячейки.

4.38.15 Borders.setOuter

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

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

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

4.38.16 Borders.setDiagonals

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

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить настройки свойств диагональных границ ячейки.

4.39 Класс RangeBorders

Объявление класса:

```
class RangeBorders(Borders)
```

Родительский класс:

`Borders` (см. раздел 4.38).

Подробное описание:

Является наследником класса `Borders` и представляет методы для управления границами диапазона ячеек. Он позволяет установить внутренние границы другого стиля, отличающегося от внешних границ (слева, справа, сверху и снизу).

Список методов:

```
def getInnerHorizontal()
def getInnerVertical()
def getInner()
def getAll()
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)
```

4.39.1 RangeBorders.getInnerHorizontal

```
RangeBorders.getInnerHorizontal()
```

Возвращаемый тип: `RangeBorders` (см. раздел 4.39).

Описание метода:

Позволяет получить свойства горизонтальных внутренних границ диапазона ячеек.

4.39.2 RangeBorders.getInnerVertical

```
RangeBorders.getInnerVertical()
```

Возвращаемый тип: `RangeBorders` (см. раздел 4.39).

Описание метода:

Позволяет получить свойства вертикальных внутренних границ диапазона ячеек.

4.39.3 RangeBorders.getInner

```
RangeBorders.getInner()
```

Возвращаемый тип: `RangeBorders` (см. раздел 4.39).

Описание метода:

Позволяет получить свойства внутренних границ диапазона ячеек.

4.39.4 RangeBorders.getAll

```
RangeBorders.getAll()
```

Возвращаемый тип: `RangeBorders` (см. раздел 4.39).

Описание метода:

Позволяет получить свойства внутренних и внешних (но не диагональных) границ.

4.39.5 RangeBorders.setLeft

```
RangeBorders.setLeft(lineProperties)
```

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить свойства левых границ диапазона ячеек.

4.39.6 RangeBorders.setRight

```
RangeBorders.setRight(lineProperties)
```

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить свойства правых границ диапазона ячеек.

4.39.7 RangeBorders.setTop

```
RangeBorders.setTop(lineProperties)
```

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить свойства верхних границ диапазона ячеек.

4.39.8 RangeBorders.setBottom

```
RangeBorders.setBottom(lineProperties)
```

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить свойства нижних границ диапазона ячеек.

4.39.9 RangeBorders.setDiagonalDown

```
RangeBorders.setDiagonalDown(lineProperties)
```

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить свойства диагональных границ диапазона ячеек.

4.39.10 RangeBorders.setDiagonalUp

```
RangeBorders.setDiagonalUp(lineProperties)
```

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить свойства диагональных границ диапазона ячеек.

4.39.11 RangeBorders.setOuter

```
RangeBorders.setOuter(lineProperties)
```

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить свойства внешних границ диапазона ячеек.

4.39.12 RangeBorders.setDiagonals

```
RangeBorders.setDiagonals(lineProperties)
```

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить свойства диагональных границ диапазона ячеек.

4.39.13 RangeBorders.setInnerHorizontal

```
RangeBorders.setInnerHorizontal(lineProperties)
```

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить свойства внутренних горизонтальных границ диапазона ячеек.

4.39.14 RangeBorders.setInnerVertical

```
RangeBorders.setInnerVertical(lineProperties)
```

Параметры:

`lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить свойства внутренних вертикальных границ диапазона ячеек.

4.39.15 RangeBorders.setInner

```
RangeBorders.setInner(lineProperties)
```

Параметры: `lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить свойства внутренних границ диапазона ячеек.

4.39.16 RangeBorders.setAll

```
RangeBorders.setAll(lineProperties)
```

Параметры: `lineProperties` (см. раздел 4.36).

Описание метода:

Позволяет установить свойства всех границ диапазона ячеек.

4.40 Класс Cell

Объявление класса:

```
class Cell(_object)
```

Подробное описание:

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

Список методов:

```
def getRange()
def getFormat()
def setFormat(format)
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 unmerge()
```

4.40.1 Cell.getRange

```
Cell.getRange()
```

Возвращаемый тип: `Range` (см. раздел 4.28).

Описание метода:

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

4.40.2 Cell.getFormat

```
Cell.getFormat()
```

Описание метода:

Возвращает тип данных ячейки (см. раздел 4.11).

4.40.3 Cell.setFormat

```
Cell.setFormat(format)
```

Параметры:

`format` – устанавливаемый формат.

Описание метода:

Устанавливает тип данных ячейки (см. раздел 4.11).

4.40.4 Cell.getCustomFormat

```
Cell.getCustomFormat()
```

Описание метода:

Возвращает строку формата ячейки.

4.40.5 Cell.setCustomFormat

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

Параметры:

`formatString` – строка формата.

Описание метода:

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

4.40.6 Cell.setBool

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

Параметры:

`value` – устанавливаемое значение.

Описание метода:

Устанавливает значение типа `bool` для ячейки, указанное в параметре `value`.

4.40.7 Cell.setNumber

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

Параметры:

`value` – устанавливаемое значение.

Описание метода:

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

4.40.8 Cell.setText

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

Параметры:

`value` – устанавливаемое значение.

Описание метода:

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

4.40.9 Cell.setFormula

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

Параметры:

`value` – устанавливаемое значение.

Описание метода:

Используется для ввода формулы в ячейку.

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 «$».

4.40.10 Cell.getFormulaAsString

```
Cell.getFormulaAsString()
```

Возвращаемый тип: `str`

Описание метода:

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

4.40.11 Cell.getFormattedValue

```
Cell.getFormattedValue()
```

Возвращаемый тип: `str`

Описание метода:

Возвращает значение ячейки, в виде строки, форматированной в соответствии с требованиями типа формата ячейки (см. раздел 4.11).

4.40.12 Cell.getRawValue

```
Cell.getRawValue()
```

Возвращаемый тип: Общий (см. раздел 4.11).

Описание метода:

Возвращает значение ячейке в формате Общий (без форматирования).

4.40.13 Cell.setFormattedValue

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

Параметры:

`value` – устанавливаемое значение.

Описание метода:

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.

4.40.14 Cell.setContent

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

Параметры:

`value` – устанавливаемое значение.

Описание метода:

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.

4.40.15 Cell.getCellProperties

```
Cell.getCellProperties()
```

Возвращаемый тип: `CellProperties` (см. раздел 4.41).

Описание метода:

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

4.40.16 Cell.setCellProperties

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

Параметры:

`cellProperties` – устанавливаемые свойства элементов оформления ячейки (см. раздел 4.41).

Описание метода:

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

4.40.17 Cell.getBorders

```
Cell.getBorders()
```

Возвращаемый тип: `Border` (см. раздел 4.38).

Описание метода:

Возвращает границы отдельной ячейки.

4.40.18 Cell.setBorders

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

Параметры:

`borders` – устанавливаемые свойства границ ячейки (см. раздел 4.38).

Описание метода:

Устанавливает границы ячейки, заданные параметром `borders`.

4.40.19 Cell.getParagraphProperties

```
Cell.getParagraphProperties()
```

Возвращаемый тип: `ParagraphProperties` (см. раздел 4.37).

Описание метода:

Возвращает свойства абзаца, содержащего в ячейке.

4.40.20 Cell.setParagraphProperties

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

Параметры:

`ParagraphProperties` – устанавливаемые свойства абзаца, содержащего в ячейке (см. раздел 4.37).

Описание метода:

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

4.40.21 Cell.unmerge

```
Cell.unmerge()
```

Описание метода:

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.

4.41 Класс CellProperties

Объявление класса:

```
class CellProperties(_object)
```

Подробное описание:

Содержит свойства оформления ячейки.

Список свойств:

`verticalAlignment` – вертикальное выравнивание ячейки (см. раздел 4.7);

`backgroundColor` – цвет фона ячейки;

`textLayout` – свойства форматирования текста в ячейке (см. раздел 4.8);

`textOrientation` – ориентация текста в ячейке (см. раздел 4.67).

4.42 Класс CellRange

Объявление класса:

```
class CellRange(_object)
```

Подробное описание:

Представляет диапазон (коллекцию) ячеек.

Список методов:

```
def getBeginRow()
def getBeginColumn()
def getLastRow()
def getLastColumn()
def getEnumerator()
def setBorders(borders)
def setCellProperties(cellProperties)
def getCellProperties()
def merge()
```

4.42.1 CellRange.getBeginRow

```
CellRange.getBeginRow()
```

Возвращаемый тип: `long`

Описание метода:

Возвращает индекс начальной строки диапазона.

4.42.2 CellRange.getBeginColumn

```
CellRange.getBeginColumn()
```

Возвращаемый тип: `long`

Описание метода:

Возвращает индекс начального столбца диапазона.

4.42.3 CellRange.getLastRow

```
CellRange.getLastRow()
```

Возвращаемый тип: `long`

Описание метода:

Возвращает индекс последней строки диапазона.

4.42.4 CellRange.getLastColumn

```
CellRange.getLastColumn()
```

Возвращаемый тип: `long`

Описание метода:

Возвращает индекс последнего столбца диапазона.

4.42.5 CellRange.getEnumerator

```
CellRange.getEnumerator()
```

Описание метода:

Возвращает коллекцию ячеек диапазона.

4.42.6 CellRange.setBorders

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

Параметры:

`borders` – устанавливаемые свойства границ ячеек диапазона.

Описание метода:

Устанавливает границы ячеек в диапазоне.

4.42.7 CellRange.setCellProperties

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

Параметры:

`cellProperties` – устанавливаемые свойства ячеек диапазона.

Описание метода:

Устанавливает свойства ячеек диапазона.

4.42.8 CellRange.getCellProperties

```
CellRange.getCellProperties()
```

Возвращаемый тип: `CellProperties` (см. раздел 4.41).

Описание метода:

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

4.42.9 CellRange.merge

```
CellRange.merge()
```

Описание метода:

Служит для объединения ячеек в диапазоне.

4.43 Класс Script

Объявление класса:

```
class Script(_object)
```

Подробное описание:

Предоставляет интерфейс для доступа к отдельным макрокомандам документа.

Список методов:

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

4.43.1 Script.getName

```
Script.getName()
```

Возвращаемый тип: `str`

Описание метода:

Возвращает название макрокоманды.

4.43.2 Script.setName

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

Параметры:

`name` – название макрокоманды.

Описание метода:

Устанавливает название макрокоманды.

4.43.3 Script.getBody

```
Script.getBody()
```

Возвращаемый тип: `str`

Описание метода:

Возвращает текст исходного кода макрокоманды на языке программирования Lua.

4.43.4 Script.setBody

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

Параметры:

`body` - текст исходного кода макрокоманды на языке программирования Lua.

Описание метода:

Устанавливает текст исходного кода макрокоманды на языке программирования Lua.

4.44 Класс Scripts

Объявление класса:

```
class Scripts(_object)
```

Подробное описание:

Предоставляет интерфейс для доступа к коллекции макрокоманд документа.

Список методов:

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

4.44.1 Scripts.getScript

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

Параметры:

`name` – наименование макрокоманды.

Возвращаемый тип: `str`

Описание метода:

Возвращает ссылку на макрокоманду с наименованием `name`.

4.44.2 Scripts.setScript

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

Параметры:

`name` – наименование макрокоманды;

`script` – исходный код макрокоманды на языке программирования Lua.

Описание метода:

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

4.44.3 Scripts.removeScript

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

Параметры:

`name` – наименование макрокоманды.

Описание метода:

Удаляет макрокоманду с наименованием `name`.

4.44.4 Scripts.getEnumerator

```
Scripts.getEnumerator()
```

Описание метода:

Возвращает коллекцию макрокоманд.

4.45 Класс TrackedChange

Объявление класса:

```
class TrackedChange(_object)
```

Подробное описание:

Класс TrackedChange предоставляет интерфейс для отслеживания изменений в документе.

Список методов:

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

4.45.1 TrackedChange.getRange

```
TrackedChange.getRange()
```

Возвращаемый тип: `Range` (см. раздел 4.28).

Описание метода:

Возвращает объект Range, который соответствует измененному диапазону внутри абзаца.

4.45.2 TrackedChange.getType

```
TrackedChange.getType()
```

Возвращаемый тип: `int`

Описание метода:

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

4.45.3 TrackedChange.getInfo

```
TrackedChange.getInfo()
```

Возвращаемый тип: `TrackedChangeInfo` (см. раздел 4.46).

Описание метода:

Позволяет получить информацию об отслеживаемых изменениях.

4.46 Класс TrackedChangeInfo

Объявление класса:

```
class TrackedChangeInfo(_object)
```

Подробное описание:

Содержит информацию об авторе и дате отслеживаемого изменения.

Список свойств:

`author` – автор изменений;

`timeStamp` – дата и время создания изменений.

Список методов:

```
def __eq__(other)
```

4.46.1 TrackedChangeInfo.__eq__

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

Параметры:

`other` – `TrackedChangeInfo` (см. раздел 4.46).

Возвращаемый тип: `bool`

Описание метода:

Позволяет использовать оператор **__eq__** для определения эквивалентности двух отслеживаемых изменений.

4.47 Класс ID

Объявление класса:

```
class ID(_object)
```

Подробное описание:

Представляет уникальный идентификатор объекта в документе.

Список методов:

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

4.47.1 ID.__eq__

```
ID.__eq__(other)
```

Параметры:

`other` – `ID` (см. раздел 4.47).

Возвращаемый тип: `bool`

Описание метода:

Позволяет использовать оператор **__eq__** для определения эквивалентности двух идентификаторов объекта.

4.47.2 ID.__ne__

```
ID.__ne__(other)
```

Параметры:

`other` – `ID` (см. раздел 4.47).

Возвращаемый тип: `bool`

Описание метода:

Позволяет использовать оператор **__ne__** для определения неэквивалентности двух идентификаторов объекта.

4.48 Класс Table

Объявление класса:

```
class Table(Block)
```

Родительский класс:

`Block` (см. раздел 4.25).

Подробное описание:

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

Список методов:

```
def getName()
def setName(newName)
def getID()
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 getCellRange(*args)
def getCell(*args)
```

4.48.1 Table.getName

```
Table.getName()
```

Возвращаемый тип: `str`

Описание метода:

Позволяет получить наименование листа табличного документа.

4.48.2 Table.setName

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

Параметры:

`newName` – наименование листа табличного документа.

Описание метода:

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

4.48.3 Table.getID

```
Table.getID()
```

Возвращаемый тип: `ID` (см. раздел 4.47).

Описание метода:

Позволяет получить уникальный идентификатор таблицы в электронном документе.

4.48.4 Table.getRowsCount

```
Table.getRowsCount()
```

Возвращаемый тип: `long`

Описание метода:

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

4.48.5 Table.getColumnsCount

```
Table.getColumnsCount()
```

Возвращаемый тип: `long`

Описание метода:

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

4.48.6 Table.insertColumnAfter

```
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 единице.

Описание метода:

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

4.48.7 Table.insertColumnBefore

```
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 единице.

Описание метода:

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

4.48.8 Table.insertRowAfter

```
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 единице.

Описание метода:

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

Table.insertRowBefore

```
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 единице.

Описание метода:

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

4.48.9 Table.removeColumn

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

Параметры:

`columnIndex` – индекс столбца, начиная с которого удаляются столбцы;

`columnsCount` – количество удаляемых столбцов, по умолчанию равно единице.

Описание метода:

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

4.48.10 Table.removeRow

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

Параметры:

`rowIndex` – индекс строки, начиная с которой удаляются строки;

`rowsCount` – количество удаляемых строк, по умолчанию равно единице.

Описание метода:

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

4.48.11 Table.setColumnWidth

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

Параметры:

`columnIndex` – номер столбца;

`width` – ширина столбца.

Описание метода:

Позволяет установить значение `width` в качестве ширины столбца с индексом `columnIndex`. Индексация столбцов начинается с ноля.

4.48.12 Table.getCellRange

```
Table.getCellRange(*args)
```

Параметры:

В качестве параметра может использоваться:

- адрес диапазона ячеек с использованием наиболее распространенной системы адресации ячеек, при которой столбцы задаются буквами, а строки – числами, например, «A1:C4»;
- положение диапазона ячеек с помощью объекта класса `CellRangePosition`, который позволяет указать строки и столбцы по индексу. Например, диапазон «A1:C4» задается как `CellRangePosition(0,0,2,3)`.

Возвращаемый тип: `CellRange` (см. раздел 4.42).

Описание метода:

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

4.48.13 Table.getCell

```
Table.getCell(*args)
```

Параметры:

В качестве параметра может использоваться:

- адрес ячейки с использованием наиболее распространенной системы адресации ячеек, при которой столбцы задаются буквами, а строки – числами, например, «A1»;
- положение ячейки с помощью объекта класса `CellPosition`, который позволяет указать строки и столбцы по индексу. Например, положение ячейки «A1» задается как `CellPosition(0,0)`.

Возвращаемый тип: `Cell` (см. раздел 4.40).

Описание метода:

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

4.49 Класс Blocks

Объявление класса:

```
class Blocks(_object)
```

Подробное описание:

Предоставляет интерфейс для доступа к блокам документа.

Список методов:

```
def getBlock(blockIndex)
def getParagraph(paragraphIndex)
def getTable(*args)
def getEnumerator()
def getParagraphsEnumerator()
def getTablesEnumerator()
```

4.49.1 Blocks.getBlock

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

Параметры:

`blockIndex` – индекс блока.

Возвращаемый тип: `Block` (см. раздел 4.25).

Описание метода:

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

4.49.2 Blocks.getParagraph

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

Параметры:

`paragraphIndex` – индекс абзаца.

Возвращаемый тип: `Paragraph` (см. раздел 4.26).

Описание метода:

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

4.49.3 Blocks.getTable

```
Blocks.getTable(*args)
```

Параметры:

`*args`

Возвращаемый тип: `Table` (см. раздел 4.48).

Описание метода:

Возвращает блок с таблицей или `none`, если таблица с таким индексом не существует в документе.

4.49.4 Blocks.getEnumerator

```
Blocks.getEnumerator()
```

Описание метода:

Возвращает коллекцию классов `Blocks`.

4.49.5 Blocks.getParagraphsEnumerator

```
Blocks.getParagraphsEnumerator()
```

Описание метода:

Возвращает коллекцию классов `Paragraphs`.

4.49.6 Blocks.getTablesEnumerator

```
Blocks.getTablesEnumerator()
```

Описание метода:

Возвращает коллекцию классов `Tables`.

4.50 Класс Bookmarks

Объявление класса:

```
class Bookmarks(_object)
```

Подробное описание:

Предоставляет интерфейс для доступа к закладкам документа.

Список методов:

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

4.50.1 Bookmarks.getBookmarkRange

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

Параметры:

`bookmarkName` – наименование закладки.

Возвращаемый тип: `Range` (см. раздел 4.28).

Описание метода:

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

4.50.2 Bookmarks.removeBookmark

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

Параметры:

`bookmarkName` – наименование закладки.

Описание метода:

Удаляет закладку с наименованием `bookmarkName`.

4.51 Класс Document

Объявление класса:

```
class Document(_object)
```

Подробное описание:

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

Список методов:

```
def saveAs(*args)
def exportTo(filePath, format)
def getBlocks()
def getBookmarks()
def getComments()
def getParagraphs()
def getScripts()
def getRange()
def merge(other)
def setChangesTrackingEnabled(value)
def isChangesTrackingEnabled()
def setPageProperties(properties)
def setPageOrientation(orientation)
def getSectionsEnumerator()
```

4.51.1 Document.saveAs

```
Document.saveAs(*args)
```

Параметры:

В качестве аргумента может использоваться путь сохранения файла с указанием имени файла и расширения.

При необходимости, в качестве второго аргумента можно использовать объект типа `SaveDocumentSettings` (см раздел 4.58), который содержит формат документа, тип документа и пароль для защиты документа от несанкционированного доступа.

Описание метода:

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

4.51.2 Document.exportTo

```
Document.exportTo(filePath, format)
```

Параметры:

`filePath` – путь размещения экспортируемого файла;

`format` – формат экспорта файла (см. раздел 4.3).

Описание метода:

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

4.51.3 Document.getBlocks

```
Document.getBlocks()
```

Возвращаемый тип: `Blocks` (см. раздел 4.49).

Описание метода:

Обеспечивает доступ к блокам данных, содержащихся в документе.

4.51.4 Document.getBookmarks

```
Document.getBookmarks()
```

Возвращаемый тип: `Bookmarks` (см. раздел 4.50).

Описание метода:

Обеспечивает доступ к коллекции закладок (`bookmark`) в документе.

4.51.5 Document.getComments

```
Document.getComments()
```

Возвращаемый тип:

Описание метода: `Comments` (см. раздел 4.23).

Обеспечивает доступ к коллекции комментариев, которые хранятся в документе.

4.51.6 Document.getParagraphs

```
Document.getParagraphs()
```

Возвращаемый тип: `Paragraphs` (см. раздел 4.27).

Описание метода:

Обеспечивает доступ к коллекции абзацев, которые хранятся в документе.

4.51.7 Document.getScripts

```
Document.getScripts()
```

Возвращаемый тип: `Scripts` (см. раздел 4.44).

Описание метода:

Обеспечивает доступ к коллекции макрокоманд в документе.

4.51.8 Document.getRange

```
Document.getRange()
```

Возвращаемый тип: `Range` (см. раздел 4.28).

Описание метода:

Возвращает диапазон, содержащий весь документ.

4.51.9 Document.merge

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

Параметры:

`other` – (см. раздел 4.51).

Описание метода:

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

4.51.10 Document.setChangesTrackingEnabled

```
Document.setChangesTrackingEnabled(value)
```

Параметры:

`value` – имеет значение `true`, если отслеживание изменений в документе необходимо включить, `false`, если отслеживание изменений в документе необходимо выключить.

Описание метода:

Устанавливает включение/выключение отслеживания изменений в документе.

4.51.11 Document.isChangesTrackingEnabled

```
Document.isChangesTrackingEnabled()
```

Возвращаемый тип: `Bool`

Описание метода:

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

4.51.12 Document.setPageProperties

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

Параметры:

`properties` – свойства страницы (см. раздел 4.21).

Описание метода:

Устанавливает ширину и высоту страниц в документе.

4.51.13 Document.setPageOrientation

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

Параметры:

`orientation` – ориентация страницы (см. раздел 4.15).

Описание метода:

Устанавливает альбомную, либо книжную ориентацию страниц в документе.

4.51.14 Document.getSectionsEnumerator

```
Document.getSectionsEnumerator()
```

Возвращаемый тип: `Section` (см. раздел 4.29).

Описание метода:

Возвращает коллекцию классов `Section`.

4.52 Класс TextProperties

Объявление класса:

```
class TextProperties(_object)
```

Подробное описание:

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

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 оформления текста;

`backgroundColor` – цветовая модель фона текста;

`characterSpacing` – межзнаковый интервал.

4.53 Класс LocaleInfo

Объявление класса:

```
class LocaleInfo(_object)
```

Подробное описание:

Предоставляет информацию о локализации.

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).

4.54 Класс TimeZone

Объявление класса:

```
class TimeZone(_object)
```

Подробное описание:

Предоставляет информацию о часовом поясе.

4.55 Класс DocumentSettings

Объявление класса:

```
class DocumentSettings(_object)
```

Подробное описание:

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

Список свойств:

`documentType` – тип документа (см. раздел 4.1);

`userInfo` – информация о пользователе (см. раздел 4.19);

`localeInfo` – информация о локализации (см. раздел 4.53);

`timeZone` – информация о временной зоне (см. раздел 4.54);

`formulaType` – система адресации ячеек (см. раздел 4.4).

4.56 Класс DSVSettings

Объявление класса:

```
class DSVSettings(_object)
```

Подробное описание:

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

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.

4.57 Класс LoadDocumentSettings

Объявление класса:

```
class LoadDocumentSettings(_object)
```

Подробное описание:

Предоставляет дополнительные настройки, необходимые для загрузки документов из файла.

Список свойств:

`commonDocumentSettings` – экземпляр класса `DocumentSettings`, общие настройки документа (см. раздел 4.55);

`encoding` – кодировка документа (см. раздел 4.5);

`dsvSettings` – экземпляр класса `dsvSettings`, настройки, необходимые для работы с файлами CSV и DSV (см. раздел 4.56);

`documentPassword` – пароль для защиты электронного документа от несанкционированного доступа. Механизм парольной защиты поддерживается только для семейства ОС Microsoft Windows.

4.58 Класс SaveDocumentSettings

Объявление класса:

```
class SaveDocumentSettings(_object)
```

Подробное описание:

Предоставляет дополнительные настройки, необходимые для сохранения документа в файл.

Список свойств:

`documentFormat` – формат документа (см. раздел 4.2);

`documentType` – тип документа (см. раздел 4.1);

`documentPassword` – пароль для защиты электронного документа от несанкционированного доступа. Механизм парольной защиты поддерживается только для семейства ОС Microsoft Windows.

4.59 Класс Application

Объявление класса:

```
class Application(_object)
```

Подробное описание:

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

Список методов:

```
def createDocument(*args)
def loadDocument(*args)
```

4.59.1 Application.createDocument

```
Application.createDocument(*args)
```

Параметры:

В качестве аргумента может быть указан тип документа, как объект класса `DocumentType` (см. раздел 4.1), в этом случае создается новый документ указанного типа с настройками по умолчанию.

Если в качестве аргумента указывается объект класса `DocumentSettings` (см. раздел 4.55), то в этом случае создается новый документ с настройками, указанными в объекте.

Возвращаемый тип: `document` (см. раздел 4.51).

Описание метода:

Создает новый документ указанного в параметрах типа с настройками по умолчанию или с указанными в параметрах. Список поддерживаемых типов документов приведен в разделе 4.1. Настройки документа приведены в разделе 4.55.

4.59.2 Application.loadDocument

```
Application.loadDocument(*args)
```

Параметры:

Аргумент может содержать путь к текстовому или табличному документу, в этом случае формат и тип документа определятся из расширения файла.

Если аргумент содержит объект класса `LoadDocumentSettings` (см. раздел 4.57), то в этом случае откроется документ с характеристиками, указанными в объекте.

Возвращаемый тип: `document` (см. раздел 4.51).

Описание метода:

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` (см. раздел 4.57).

4.60 Класс Scripting

Объявление класса:

```
class Scripting(_object)
```

Подробное описание:

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

Список методов:

```
def runScript(name)
```

4.60.1 Scripting.runScript

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

Параметры:

`name` – названием макрокоманды.

Описание метода:

Запускает макрокоманду с названием `name`.

4.61 Глобальная функция createScripting

Объявление глобальной функции:

```
createScripting(document)
```

Параметры глобальной функции:

`document` (см. раздел 4.51).

Подробное описание:

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

4.62 Класс PointU

Объявление класса:

```
class PointU(_object)
```

Подробное описание:

Представляет собой точку в двухмерном пространстве.

Список свойств:

x – координата точки по оси x;

y – координата точки по оси y.

Список методов:

```
def toString()
```

4.62.1 PointU.toString

```
PointU.toString()
```

Возвращаемый тип: `str`

Описание метода:

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

4.63 Класс RectU

Объявление класса:

```
class RectU(_object)
```

Подробное описание:

Описывает прямоугольник в двухмерном пространстве.

Список свойств:

`topLeft` – координата левой верхней вершины прямоугольника;

`bottomRight` – координата правой нижней вершины прямоугольника.

Список методов:

```
def toString()
```

4.63.1 RectU.toString

```
RectU.toString()
```

Возвращаемый тип: `str`

Описание метода:

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

4.64 Класс SizeU

Объявление класса:

```
class SizeU(_object)
```

Подробное описание:

Описывает размер объекта в двухмерном пространстве.

Список свойств:

`width` – ширина объекта в двухмерном пространстве;

`height` – высота объекта в двухмерном пространстве.

Список методов:

```
def toString()
```

4.64.1 SizeU.toString

```
SizeU.toString()
```

Возвращаемый тип: `str`

Описание метода:

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

4.65 Класс HeaderFooter

Объявление класса:

```
class HeaderFooter(_object)
```

Подробное описание:

Определяет отдельный колонтитул в документе.

Список методов:

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

4.65.1 HeaderFooter.getType

```
HeaderFooter.getType()
```

Возвращаемый тип: `long` (см. раздел 4.16).

Описание метода:

Предоставляет информацию о типе колонтитула.

4.65.2 HeaderFooter.getBlocks

```
HeaderFooter.getBlocks()
```

Возвращаемый тип: `Blocks` (см. раздел 4.49).

Описание метода:

Предоставляет доступ к блокам, которые содержатся в колонтитуле.

4.65.3 HeaderFooter.getRange

```
HeaderFooter.getRange()
```

Возвращаемый тип: `Range` (см. раздел 4.28).

Описание метода:

Предоставляет диапазон с содержанием верхнего или нижнего колонтитулов.

4.66 Класс HeadersFooters

Объявление класса:

```
class HeadersFooters(_object)
```

Подробное описание:

Представляет коллекцию верхних и нижних колонтитулов в разделе (Section) документа. См. описание класса Section в разделе 4.29.

Список методов:

```
def getEnumerator()
```

4.66.1 HeadersFooters.getEnumerator

```
HeadersFooters.getEnumerator()
```

Описание метода:

Возвращает коллекцию колонтитулов.

4.67 Класс TextOrientation

Объявление класса:

```
class TextOrientation(_object)
```

Подробное описание:

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

Список методов:

```
def getAngle()
```

4.67.1 TextOrientation.getAngle

```
TextOrientation.getAngle()
```

Возвращаемый тип: `long`.

Описание метода:

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

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 5.2 указывается какие были 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:

```
from MyOfficeSDKDocumentAPI import DocumentAPI

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.

