openpyxl Documentation

发布 *3.0.7*

See AUTHORS

2021 年 08 月 19 日

Contents

Author Eric Gazoni, Charlie Clark

Source code http://bitbucket.org/openpyxl/openpyxl/src

Issues http://bitbucket.org/openpyxl/openpyxl/issues

Generated 2021 年 08 月 19 日

Version 3.0.7

CHAPTER 1

支持

这是一个由志愿者在业余时间维护的开源项目。这很可能意味着会缺少你想要的特定的功能。But things don’t have to stay that way. 你可以对这个项目进行贡献*Development* 或者和开发者联系来开发特定的功能。

可以向 ‘Clark Consulting & Research <http://www.clark-consulting.eu/>‘_ 和 Adimian 寻求专业支持。欢迎为该项目捐款以支持进一步的开发和维护。

错误报告和功能请求可以使用 issue tracker 来提交。请提供错误的完整最终，并尽可能提交示例文件。如果出于保密原因您无法公开提供文件，请与开发人员联系。

CHAPTER 2

如何贡献

只要遵从了以下步骤，我们欢迎任何帮助:

1. 为了每一个独立的功能开了新的 fork (https://bitbucket.org/openpyxl/openpyxl/fork) ，也不要想着同时解决所有的问题，这也能使为 review 和 merge 你的 changes 的人更加方便;-)

2. Hack hack hack

3. 不要忘了为你的修改添加单元测试！（是的，即使只有一行代码，没有单元测试也是不会被接受的哦。）如果不知道怎么做，可以参考源代码中大量的例子

4. 如果添加了一个完整的功能或者对某个功能做出了改进，你可以自豪地把自己加入作者文件中:-)

5. 为了让大家知道你刚提交的功能是多么的棒，务必更新一下文档！

6. 当以上步骤都完成之后，提一个 pull request (在 **你**的 repository 页点击大大的 “pull request” 按钮）然后等你的代码被 review。如果以上步骤都完成了，那么就会合并到主 repository 。

更多信息请查询*Development*

2.1 其他提供帮助的方式

即使你不会写代码（或者代码写得不是很好），也有多种方式来作出贡献

- 为 bug 追踪器（bug tracker）进行分流：关闭已经解决的，无关的，不能复现的 bug
- 对几乎每个方面的文档进行更新：增加了大量大型的特性（主要是图表和图像）但是没有文档，因此很难用新特性来做点什么

- proposing compatibility fixes for different versions of Python: 我们支持 2.7, 3.4, 3.5, 3.6 和 3.7

CHAPTER 3

安装

使用 pip 安装 openpyxl。建议在不带系统软件包的 Python virtualenv 中执行此操作:

```
$ pip install openpyxl
```

注解： 支持流行的 lxml 库，在创建大量文件的时候特别有用。

警告： 为了在 openpyxl 文件中包含（jpeg, png, bmp,⋯）等图片，你还需要安装 *pillow*:

```
$ pip install pillow
```

或者你也可以浏览 https://pypi.python.org/pypi/Pillow/，选择最新版本或下拉到页面最后选择 Windows 二进制版

3.1 Working with a checkout

Sometimes you might want to work with the checkout of a particular version. This may be the case if bugs have been fixed but a release has not yet been made.

```
$ pip install -e hg+https://bitbucket.org/openpyxl/openpyxl@3.0#egg=openpyxl
```

CHAPTER 4

Usage examples

4.1 教程

4.1.1 教程

新建工作表

无须在文件系统中创建文件即可开始使用 openpyxl。只要导入 *Workbook* 类就可以开始工作了:

```
>>> from openpyxl import Workbook
>>> wb = Workbook()
```

一个工作表至少有一个工作簿. 你可以通过 *Workbook.active* 来获取这个属性:

```
>>> ws = wb.active
```

注解: 这个值默认为 0。除非你修改了这个值，不然这个方法会一直获取第一个工作表。

你可以使用 *Workbook.create_sheet* 方法来创建新的工作簿:

```
>>> ws1 = wb.create_sheet("Mysheet") # insert at the end (default)
# or
```

(下页继续)

(续上页)

```
>>> ws2 = wb.create_sheet("Mysheet", 0) # insert at first position
# or
>>> ws3 = wb.create_sheet("Mysheet", -1) # insert at the penultimate position
```

工作薄在创建时会自动生成一个名字，以 (Sheet，Sheet1，Sheet2，…) 来进行命名。你也可以通过 *Worksheet.title* 属性来修改命名:

```
ws.title = "New Title"
```

默认情况下，包含该标题的选项卡的背景颜色为白色。你也可以使用 `RRGGBB` 颜色来改变 *Worksheet.sheet_properties.tabColor* 属性:

```
ws.sheet_properties.tabColor = "1072BA"
```

给工作表命名后，就可以将其作为工作簿的键:

```
>>> ws3 = wb["New Title"]
```

你可以使用 *Workbook.sheetname* 属性查看工作簿中所有工作表的名称:

```
>>> print(wb.sheetnames)
['Sheet2', 'New Title', 'Sheet1']
```

你可以遍历工作表:

```
>>> for sheet in wb:
...     print(sheet.title)
```

你可以在 ** 一个工作表 ** 中创建一个工作簿的复制:

Workbook.copy_worksheet method:

```
>>> source = wb.active
>>> target = wb.copy_worksheet(source)
```

注解: 只有单元格（包含值、样式、超链接和注释）以及确定的工作簿属性（包含尺寸、格式和属性）会被复制。其余的工作表/工作簿属性都不会被复制，例如：文件、图表。

你也 ** 不能 ** 跨工作表复制工作簿。工作表以 *read-only* 或 *write_only* 模式打开时也无法复制。

Playing with data

访问单元格

现在我们已经知道如何创建工作表，接下来可以开始修改单元格内容了。可以直接通过工作表的键来访问单元格:

```
>>> c = ws['A4']
```

此处会返回 A4 单元格，如果不存在不将会进行创建可以直接分配值:

```
>>> ws['A4'] = 4
```

这里是 *Worksheet.cell* 方法.

也可以通过行列符号访问单元格:

```
>>> d = ws.cell(row=4, column=2, value=10)
```

注解: 当工作薄在内存中被创建之后并没有单元格 *cells* ，单元格只有在被第一次访问 (access) 的时候才会创建

警告: 由于这个特性，即使你未对单元格赋值，滚动浏览而非直接访问时也会在内存中直接创建。

Something like

```
>>> for x in range(1,101):
...        for y in range(1,101):
...            ws.cell(row=x, column=y)
```

will create 100x100 cells in memory, for nothing.

访问大量单元格

可以使用切片来访问一系列单元格:

```
>>> cell_range = ws['A1':'C2']
```

一系列的行和列也可以通过类似的方法获取:

```
>>> colC = ws['C']
>>> col_range = ws['C:D']
>>> row10 = ws[10]
>>> row_range = ws[5:10]
```

你也使用 *Worksheet.iter_rows* 方法:

```
>>> for row in ws.iter_rows(min_row=1, max_col=3, max_row=2):
...     for cell in row:
...         print(cell)
<Cell Sheet1.A1>
<Cell Sheet1.B1>
<Cell Sheet1.C1>
<Cell Sheet1.A2>
<Cell Sheet1.B2>
<Cell Sheet1.C2>
```

同样 *Worksheet.iter_cols* 方法会返回列:

```
>>> for col in ws.iter_cols(min_row=1, max_col=3, max_row=2):
...     for cell in col:
...         print(cell)
<Cell Sheet1.A1>
<Cell Sheet1.A2>
<Cell Sheet1.B1>
<Cell Sheet1.B2>
<Cell Sheet1.C1>
<Cell Sheet1.C2>
```

注解: 由于性能原因 *Worksheet.iter_cols()* 方法在只读模式下不可用。

如果需要遍历文件中的所有行和列，可以使用 *Worksheet.rows* 属性

```
>>> ws = wb.active
>>> ws['C9'] = 'hello world'
>>> tuple(ws.rows)
((<Cell Sheet.A1>, <Cell Sheet.B1>, <Cell Sheet.C1>),
(<Cell Sheet.A2>, <Cell Sheet.B2>, <Cell Sheet.C2>),
(<Cell Sheet.A3>, <Cell Sheet.B3>, <Cell Sheet.C3>),
(<Cell Sheet.A4>, <Cell Sheet.B4>, <Cell Sheet.C4>),
(<Cell Sheet.A5>, <Cell Sheet.B5>, <Cell Sheet.C5>),
(<Cell Sheet.A6>, <Cell Sheet.B6>, <Cell Sheet.C6>),
(<Cell Sheet.A7>, <Cell Sheet.B7>, <Cell Sheet.C7>),
(<Cell Sheet.A8>, <Cell Sheet.B8>, <Cell Sheet.C8>),
(<Cell Sheet.A9>, <Cell Sheet.B9>, <Cell Sheet.C9>))
```

或者 *Worksheet.columns* 属性:

```
>>> tuple(ws.columns)
((<Cell Sheet.A1>,
<Cell Sheet.A2>,
<Cell Sheet.A3>,
<Cell Sheet.A4>,
<Cell Sheet.A5>,
<Cell Sheet.A6>,
...
<Cell Sheet.B7>,
<Cell Sheet.B8>,
<Cell Sheet.B9>),
(<Cell Sheet.C1>,
<Cell Sheet.C2>,
<Cell Sheet.C3>,
<Cell Sheet.C4>,
<Cell Sheet.C5>,
<Cell Sheet.C6>,
<Cell Sheet.C7>,
<Cell Sheet.C8>,
<Cell Sheet.C9>))
```

注解: 由于性能原因 *Worksheet.columns* 方法在只读模式下不可用。

Values only

如果你只想要工作薄的值，你可以使用 *Worksheet.values* 属性。这会遍历工作簿中所有的行但只返回单元格值:

```
for row in ws.values:
   for value in row:
     print(value)
```

Worksheet.iter_rows 和 *Worksheet.iter_cols* 可以用 `values_only` 参数来返回单元格值:

```
>>> for row in ws.iter_rows(min_row=1, max_col=3, max_row=2, values_only=True):
...   print(row)
```

(下页继续)

(续上页)

```
(None, None, None)
(None, None, None)
```

数据存储

一旦有了 `Cell`, 我们可以为其分配一个值:

```
>>> c.value = 'hello, world'
>>> print(c.value)
'hello, world'

>>> d.value = 3.14
>>> print(d.value)
3.14
```

保存至文件

保存工作表最简单和安全的方法就是使用 `Workbook` 类的 `Workbook.save()` 方法:

```
>>> wb = Workbook()
>>> wb.save('balances.xlsx')
```

警告: 这个操作将会无警告直接覆盖已有文件

注解: 文件名后缀并不强制为 xlsx 或 xlsm，但是如果没使用官方后缀名，会在用其他应用打开时遇到一些麻烦。

由于 OOXML 文件基本上都是 ZIP 文件，你也可以用你喜欢的 ZIP 压缩管理器打开

你可以指定属性 *template=True* 将工作表保存为模板:

```
>>> wb = load_workbook('document.xlsx')
>>> wb.template = True
>>> wb.save('document_template.xltx')
```

或者设置属性为 *False* (默认) 将其保存为一个文档:

```
>>> wb = load_workbook('document_template.xltx')
>>> wb.template = False
>>> wb.save('document.xlsx', as_template=False)
```

警告： 你应当在保存模板文档时监视数据的属性和文档拓展名，否则引擎可能会无法打开文档。

注解： 以下操作将会失败:

```
>>> wb = load_workbook('document.xlsx')
>>> # 需要保存为 *.xlsx 拓展名
>>> wb.save('new_document.xlsm')
>>> # 微软 Excel 无法打开这个文档
>>>
>>> # or
>>>
>>> # 需要执行 keep_vba=True
>>> wb = load_workbook('document.xlsm')
>>> wb.save('new_document.xlsm')
>>> # 微软 Excel 将不会打开这个文档
>>>
>>> # or
>>>
>>> wb = load_workbook('document.xltm', keep_vba=True)
>>> # 如果需要一个模板文档，需要将拓展名指定为 *.xltm.
>>> wb.save('new_document.xlsm')
>>> # 微软 Excel 将不会打开这个文档
```

保存成流 (stream)

如果想把文件保存成流。例如当使用 Pyramid, Flask 或 Django 等 web 应用程序时，可以提供 NamedTemporaryFile():

```
>>> from tempfile import NamedTemporaryFile
>>> from openpyxl import Workbook
>>> wb = Workbook()
>>> with NamedTemporaryFile() as tmp:
        wb.save(tmp.name)
```

(下页继续)

(续上页)

```
        tmp.seek(0)
        stream = tmp.read()
```

从文件加载

可以使用 `openpyxl.load_workbook()` 方法来打开一个已存在的工作表:

```
>>> from openpyxl import load_workbook
>>> wb2 = load_workbook('test.xlsx')
>>> print wb2.sheetnames
['Sheet2', 'New Title', 'Sheet1']
```

教程到这里就结束了, 你可以继续简单用法 部分

4.2 Cookbook

4.2.1 简单用法

写入工作表

```
>>> from openpyxl import Workbook
>>> from openpyxl.utils import get_column_letter
>>>
>>> wb = Workbook()
>>>
>>> dest_filename = 'empty_book.xlsx'
>>>
>>> ws1 = wb.active
>>> ws1.title = "range names"
>>>
>>> for row in range(1, 40):
...     ws1.append(range(600))
>>>
>>> ws2 = wb.create_sheet(title="Pi")
>>>
>>> ws2['F5'] = 3.14
>>>
>>> ws3 = wb.create_sheet(title="Data")
```

(下页继续)

(续上页)

```
>>> for row in range(10, 20):
...     for col in range(27, 54):
...         _ = ws3.cell(column=col, row=row, value="{0}".format(get_column_letter(col)))
>>> print(ws3['AA10'].value)
AA
>>> wb.save(filename = dest_filename)
```

读取已有的工作表

```
>>> from openpyxl import load_workbook
>>> wb = load_workbook(filename = 'empty_book.xlsx')
>>> sheet_ranges = wb['range names']
>>> print(sheet_ranges['D18'].value)
3
```

注解: 在使用 *load_workbook* 函数时有几个可供选择。

- *data_only* controls whether cells with formulae have either the formula (default) or the value stored the last time Excel read the sheet.
- *keep_vba* controls whether any Visual Basic elements are preserved or not (default). If they are preserved they are still not editable.

警告: 用 openpyxl 打开文件并进行保存会导致图片和图表的丢失，因为 openpyxl 无法读取 Excel 文件所有可能的项。

使用数字格式

```
>>> import datetime
>>> from openpyxl import Workbook
>>> wb = Workbook()
>>> ws = wb.active
>>> # set date using a Python datetime
>>> ws['A1'] = datetime.datetime(2010, 7, 21)
>>>
>>> ws['A1'].number_format
'yyyy-mm-dd h:mm:ss'
```

使用公式

```
>>> from openpyxl import Workbook
>>> wb = Workbook()
>>> ws = wb.active
>>> # add a simple formula
>>> ws["A1"] = "=SUM(1, 1)"
>>> wb.save("formula.xlsx")
```

警告： 您必须为函数使用英文名称，并且函数参数必须用逗号分隔，而不能使用其他标点符号，例如分号。

openpyxl 不会检查公式但可以检查公式的名称:

```
>>> from openpyxl.utils import FORMULAE
>>> "HEX2DEC" in FORMULAE
True
```

如果你正在尝试使用一个未知的公式，可能是因为这公式未被包含在最初的规范中。这样的公式只有以 *_xlfn* 为前缀才能起作用。

合并 / 拆分单元格

When you merge cells all cells but the top-left one are **removed** from the worksheet. To carry the border-information of the merged cell, the boundary cells of the merged cell are created as MergeCells which always have the value None. See 合并单元格的样式 for information on formatting merged cells.

```
>>> from openpyxl.workbook import Workbook
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> ws.merge_cells('A2:D2')
>>> ws.unmerge_cells('A2:D2')
>>>
>>> # or equivalently
>>> ws.merge_cells(start_row=2, start_column=1, end_row=4, end_column=4)
>>> ws.unmerge_cells(start_row=2, start_column=1, end_row=4, end_column=4)
```

插入图像

```
>>> from openpyxl import Workbook
>>> from openpyxl.drawing.image import Image
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>> ws['A1'] = 'You should see three logos below'
```

```
>>> # create an image
>>> img = Image('logo.png')
```

```
>>> # add to worksheet and anchor next to cells
>>> ws.add_image(img, 'A1')
>>> wb.save('logo.xlsx')
```

隐藏

```
>>> import openpyxl
>>> wb = openpyxl.Workbook()
>>> ws = wb.create_sheet()
>>> ws.column_dimensions.group('A','D', hidden=True)
>>> ws.row_dimensions.group(1,10, hidden=True)
>>> wb.save('group.xlsx')
```

CHAPTER 5

性能

5.1 性能

openpyxl 尝试来平衡功能与性能。如果有疑问，我们把重点放在功能而非性能上：一旦建立了 API，性能调整将变得更简单。与其他库和应用程序相比，内存使用率很高，约为原始文件大小的 50 倍，例如 50 MB 的 Excel 文件为内存使用约为 2.5 GB。由于许多用例只涉及读取或写入文件，the 优化模式 modes mean this is less of a problem.

5.1.1 基准测试

所有基准都是综合性的，并且高度依赖于硬件，但是它们仍然可以提供说明（indication）。

写入性能

benchmark code 可以调整使用更多的工作表以及数据中字符串的比例。由于不同版本的 Python 也会对性能有着显著影响，所以使用了 driver script 对 tox 环境下不同的版本 Python 进行测试。

性能与出色的替代库 xlsxwriter 进行了比较

```
Versions:
python: 3.6.9
openpyxl: 3.0.1
xlsxwriter: 1.2.5
```

(下页继续)

(续上页)

```
Dimensions:
    Rows = 1000
    Cols = 50
    Sheets = 1
    Proportion text = 0.10

Times:
    xlsxwriter            :   0.59
    xlsxwriter (optimised):   0.54
    openpyxl              :   0.73
    openpyxl (optimised)  :   0.61


Versions:
python: 3.7.5
openpyxl: 3.0.1
xlsxwriter: 1.2.5

Dimensions:
    Rows = 1000
    Cols = 50
    Sheets = 1
    Proportion text = 0.10

Times:
    xlsxwriter            :   0.65
    xlsxwriter (optimised):   0.53
    openpyxl              :   0.70
    openpyxl (optimised)  :   0.63


Versions:
python: 3.8.0
openpyxl: 3.0.1
xlsxwriter: 1.2.5

Dimensions:
    Rows = 1000
    Cols = 50
```

(下页继续)

(续上页)

```
    Sheets = 1
    Proportion text = 0.10

Times:
    xlsxwriter             :   0.54
    xlsxwriter (optimised):   0.50
    openpyxl               :   1.10
    openpyxl (optimised)  :   0.57
```

读取性能

读取性能测试使用了 bug report 提供的文件，和早期的 xlrd 库进行比较。xlrd 主要用于.XLS 文件较旧的 BIFF 文件格式，它对 XLSX 文件支持有限。

基准测试 代码显示了处理文件时正确选项的重要性。在这种情况下，禁用外部链接将让 openpyxl 停止打开链接工作表的缓存副本。

两个库的一个主要区别是 openpyxl 的只读模式可以快速打开工作簿，使其适用于多进程，这也大大减少了内存的使用。xlrd 也不会自动将日期和时间转换为 Python 的 datetime，尽管它会相应地注释单元格 (annotate cells)，但是在客户端代码中这样做会大大降低性能。

```
Versions:
python: 3.6.9
xlread: 1.2.0
openpyxl: 3.0.1

openpyxl, read-only
    Workbook loaded 1.14s
    OptimizationData 23.17s
    Output Model 0.00s
    >>DATA>> 0.00s
    Store days 0% 23.92s
    Store days 100% 17.35s
    Total time 65.59s
    0 cells in total

Versions:
python: 3.7.5
xlread: 1.2.0
openpyxl: 3.0.1

```

(下页继续)

(续上页)

```
openpyxl, read-only
    Workbook loaded 0.98s
    OptimizationData 21.35s
    Output Model 0.00s
    >>DATA>> 0.00s
    Store days 0% 20.70s
    Store days 100% 16.16s
    Total time 59.19s
    0 cells in total

Versions:
python: 3.8.0
xlread: 1.2.0
openpyxl: 3.0.1

openpyxl, read-only
    Workbook loaded 0.90s
    OptimizationData 19.58s
    Output Model 0.00s
    >>DATA>> 0.00s
    Store days 0% 19.35s
    Store days 100% 15.02s
    Total time 54.85s
    0 cells in total
```

并行

读取工作表会占用大量 CPU 从而限制了从并行中获取好处。但是，如果你主要对 dump 工作表内容感兴趣，你可以使用 openpyxl 的只读模式打开复数工作表来利用多核 CPU。

Sample code using the same source file as for read performance shows that performance scales reasonably with only a slight overhead due to creating additional Python processes.

CHAPTER 6

其他主题

6.1 优化模式

6.1.1 只读模式

有时，你可能需要打开或写入极端大的 XLSX 文件，但通用的 openpyxl 程序无法处理这么大的负载。幸运的是，有两种模式可以使你在（几乎）恒定的内存消耗下读写无限量的数据。

介绍 `openpyxl.worksheet._read_only.ReadOnlyWorksheet`:

```
from openpyxl import load_workbook
wb = load_workbook(filename='large_file.xlsx', read_only=True)
ws = wb['big_data']

for row in ws.rows:
    for cell in row:
        print(cell.value)
```

警告:

- `openpyxl.worksheet._read_only.ReadOnlyWorksheet` 是只读的

单元格的返回值不是*openpyxl.cell.cell.Cell* 而是 `openpyxl.cell._read_only.`

ReadOnlyCell.

工作表尺寸（dimensions）

只读模式依赖创建文件的应用以及库提供工作表的正确信息，尤其是文件的已使用部分，称之为尺寸（dimensions）。一些应用汇进行设置错误。可以使用 *ws.calculate_dimension()* 函数来检查工作表的尺寸（dimensions）。如果返回和范围和你知道的不一样，比如说 *A1:A1* ，你可以简单重置 *max_row* 和 *max_column* 属性，即可使用该文件:

```
ws.reset_dimensions()
```

6.1.2 只写模式

常规的*openpyxl.worksheet.worksheet.Worksheet* 被替代成更快的 openpyxl.worksheet._write_only.WriteOnlyWorksheet 。当你想导出大量数据的时候请确保安装了 *lxml* 库.

```
>>> from openpyxl import Workbook
>>> wb = Workbook(write_only=True)
>>> ws = wb.create_sheet()
>>>
>>> # now we'll fill it with 100 rows x 200 columns
>>>
>>> for irow in range(100):
...     ws.append(['%d' % i for i in range(200)])
>>> # save the file
>>> wb.save('new_big_file.xlsx') # doctest: +SKIP
```

如果你想要带有样式或者注释的单元格可以使用 openpyxl.cell.WriteOnlyCell()

```
>>> from openpyxl import Workbook
>>> wb = Workbook(write_only = True)
>>> ws = wb.create_sheet()
>>> from openpyxl.cell import WriteOnlyCell
>>> from openpyxl.comments import Comment
>>> from openpyxl.styles import Font
>>> cell = WriteOnlyCell(ws, value="hello world")
>>> cell.font = Font(name='Courier', size=36)
>>> cell.comment = Comment(text="A comment", author="Author's Name")
>>> ws.append([cell, 3.14, None])
>>> wb.save('write_only_file.xlsx')
```

以上会创建只有一张工作表的只写工作簿，一行写入（append）三个单元格：一个带有自定义字体和注释的文字单元格，一个浮点数单元格和一个空单元格（一定会被丢弃）。

警告:

- 和普通工作簿不同的是，新创建的只写工作簿没有任何工作表；工作表只能由 `create_sheet()` 方法进行创建。
- 在只读工作簿中，只能由 `append()` 来添加行。无法使用 `cell()` 或 `iter_rows()` 对任意位置的单元进行读取或写入。
- 可以导出不限量的数据（即使超过 Excel 的处理上限），同时内存使用量小于 10Mb。
- 一个只写工作簿只能保存一次。之后如果任何尝试保存和添加数据（append()）的操作都会会引发*openpyxl.utils.exceptions.WorkbookAlreadySaved* 错误。
- Everything that appears in the file before the actual cell data must be created before cells are added because it must written to the file before then. For example, *freeze_panes* should be set before cells are added.

6.2 插入删除行或列，移动范围单元格

6.2.1 插入行和列

你可以使用工作表相关的方法来插入行和列:

- *openpyxl.worksheet.worksheet.Worksheet.insert_rows()*
- *openpyxl.worksheet.worksheet.Worksheet.insert_cols()*
- *openpyxl.worksheet.worksheet.Worksheet.delete_rows()*
- *openpyxl.worksheet.worksheet.Worksheet.delete_cols()*

默认是一行或一列。例如在第七行插入一行 (存在第七行):

```
>>> ws.insert_rows(7)
```

6.2.2 删除多行或多列

删除 `F:H` 列:

```
>>> ws.delete_cols(6, 3)
```

6.2.3 Moving ranges of cells

你也可以在一个工作表内移动范围单元格:

```
>>> ws.move_range("D4:F10", rows=-1, cols=2)
```

这会将 D4:F10 单元格向上移动一行向右移动两列，已存在的单元格将会被覆盖

如果单元格包含公式，你可以让 openpyxl 帮你进行 translate，但也并非总是你想要的结果，因此默认是禁用的。同时，只有单元格本身的公式将会被 translate。其他单元格对该单元格的引用或 defined name 将不会被更新。你可以使用 *Parsing Formulas* 来做这件事:

```
>>> ws.move_range("G4:H10", rows=1, cols=1, translate=True)
```

公式中的相对引用移动一行和一列

6.3 与 Pandas 和 NumPy 一起使用

openpyxl 可以与流行的 Pandas 和 NumPy 一起使用

6.3.1 NumPy 支持

openpyxl 内置支持 NumPy 的 float，integer 和 boolean 类型。DateTimes are supported using the Pandas’ Timestamp type.

6.3.2 和 Pandas Dataframes 一起使用

openpyxl.utils.dataframe.dataframe_to_rows() 提供了一种使用 Pandas Dataframes 的简单方法:

```
from openpyxl.utils.dataframe import dataframe_to_rows
wb = Workbook()
ws = wb.active

for r in dataframe_to_rows(df, index=True, header=True):
    ws.append(r)
```

虽然 Pandas 本身支持对 Excel 的转换，但这为客户端代码提供了更多的灵活性，包括直接将数据帧（stream dataframes）流传输到文件的能力。

将 dataframe 转换为工作簿时高亮表头和索引:

```
wb = Workbook()
ws = wb.active

for r in dataframe_to_rows(df, index=True, header=True):
    ws.append(r)

for cell in ws['A'] + ws[1]:
    cell.style = 'Pandas'

wb.save("pandas_openpyxl.xlsx")
```

另外，如果你只想转换数据，你可以使用只写模式:

```
from openpyxl.cell.cell import WriteOnlyCell
wb = Workbook(write_only=True)
ws = wb.create_sheet()

cell = WriteOnlyCell(ws)
cell.style = 'Pandas'

 def format_first_row(row, cell):

    for c in row:
        cell.value = c
        yield cell

rows = dataframe_to_rows(df)
first_row = format_first_row(next(rows), cell)
ws.append(first_row)

for row in rows:
    row = list(row)
    cell.value = row[0]
    row[0] = cell
    ws.append(row)

wb.save("openpyxl_stream.xlsx")
```

此代码和标准工作簿一起起作用。

6.3.3 将工作簿转换为 Dataframe（PS：样例文件可以参考 df.to_excel() 的文件）

如果工作簿没有表头和索引很容易用 *values* 属性将一个工作簿转换为 Dataframe:

```
df = DataFrame(ws.values)
```

如果工作簿确实有表头和索引，例如 Pandas 创建的文件，那还要做更多的一些工作:

```
from itertools import islice
data = ws.values
cols = next(data)[1:]
data = list(data)
idx = [r[0] for r in data]
data = (islice(r, 1, None) for r in data)
df = DataFrame(data, index=idx, columns=cols)
```

6.4 图表

6.4.1 图标类型

以下图表是可用的:

面积图

二维面积图

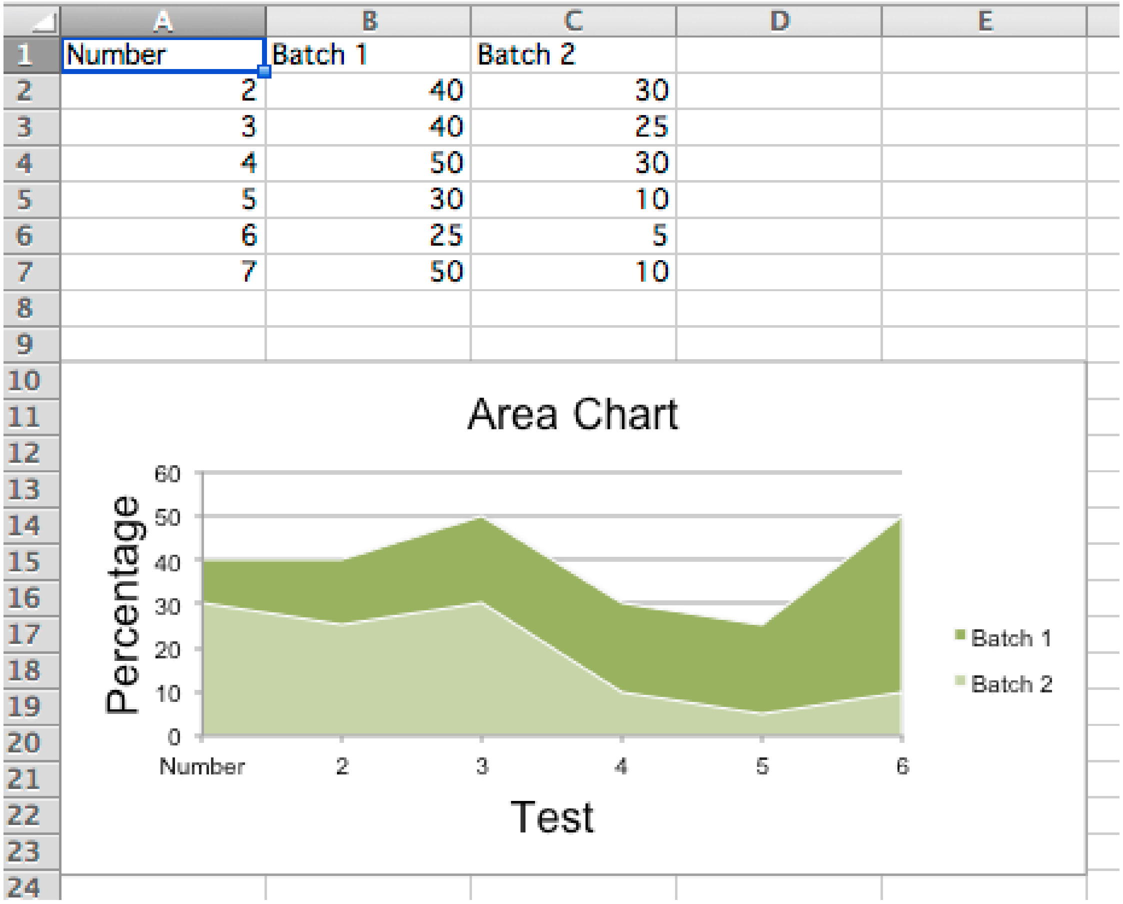
面积图类似于折线图，不同之处在于填充了绘制线下方的区域。通过将分组设置为“标准”，“堆叠”或“百分比堆叠”，可以使用不同的变体。默认为“标准”。

```
from openpyxl import Workbook
from openpyxl.chart import (
    AreaChart,
    Reference,
    Series,
)

wb = Workbook()
ws = wb.active

```

（下页继续）

(续上页)

```
rows = [
    ['Number', 'Batch 1', 'Batch 2'],
    [2, 40, 30],
    [3, 40, 25],
    [4, 50, 30],
    [5, 30, 10],
    [6, 25, 5],
    [7, 50, 10],
]

for row in rows:
    ws.append(row)

chart = AreaChart()
chart.title = "Area Chart"
chart.style = 13
chart.x_axis.title = 'Test'
chart.y_axis.title = 'Percentage'

cats = Reference(ws, min_col=1, min_row=1, max_row=7)
data = Reference(ws, min_col=2, min_row=1, max_col=3, max_row=7)
chart.add_data(data, titles_from_data=True)
chart.set_categories(cats)

ws.add_chart(chart, "A10")

wb.save("area.xlsx")
```



Number	Batch 1	Batch 2
2	40	30
3	40	25
4	50	30
5	30	10
6	25	5
7	50	10

三维面积图

你也可以创建三维面积图

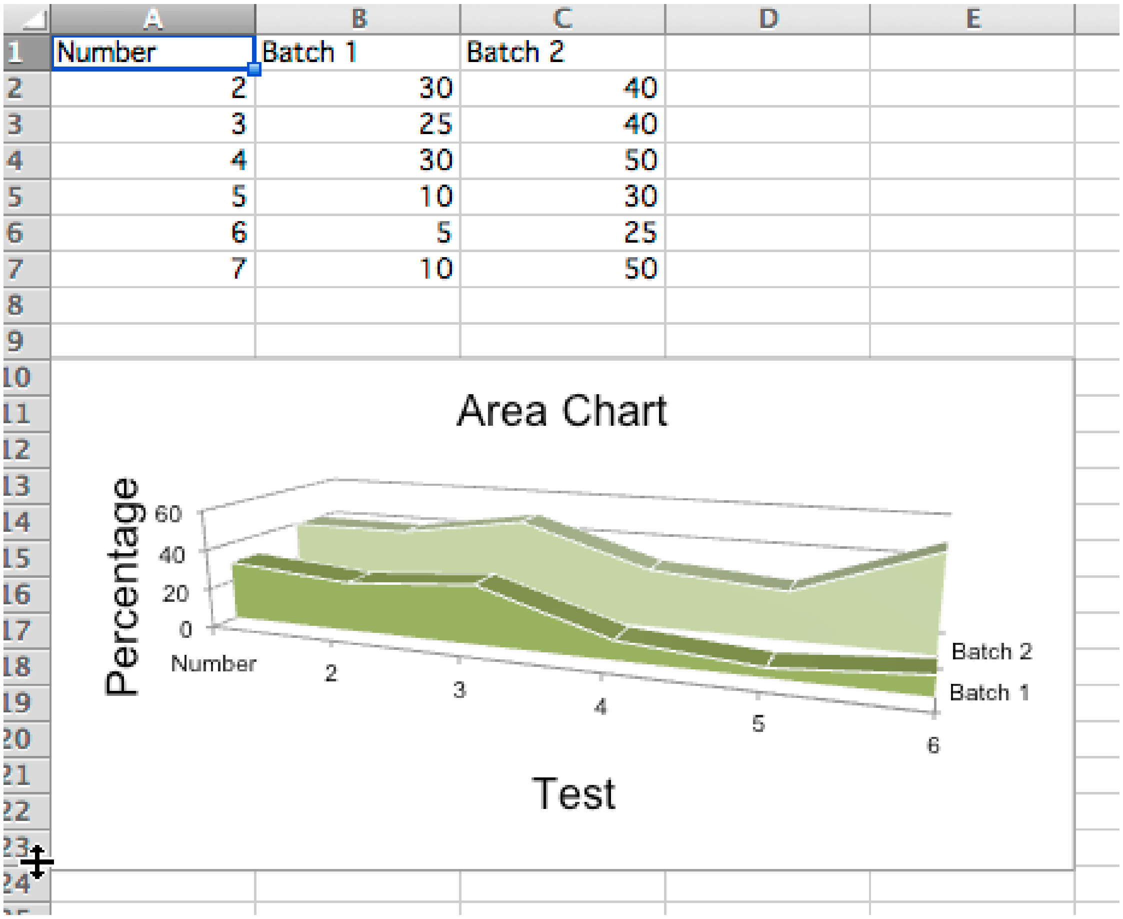
```
from openpyxl import Workbook
from openpyxl.chart import (
    AreaChart3D,
    Reference,
    Series,
)

wb = Workbook()
ws = wb.active

rows = [
    ['Number', 'Batch 1', 'Batch 2'],
    [2, 30, 40],
    [3, 25, 40],
```

（下页继续）

(续上页)

```
    [4 ,30, 50],
    [5 ,10, 30],
    [6,  5, 25],
    [7 ,10, 50],
]

for row in rows:
    ws.append(row)

chart = AreaChart3D()
chart.title = "Area Chart"
chart.style = 13
chart.x_axis.title = 'Test'
chart.y_axis.title = 'Percentage'
chart.legend = None

cats = Reference(ws, min_col=1, min_row=1, max_row=7)
data = Reference(ws, min_col=2, min_row=1, max_col=3, max_row=7)
chart.add_data(data, titles_from_data=True)
chart.set_categories(cats)

ws.add_chart(chart, "A10")

wb.save("area3D.xlsx")
```

这将生成一个简单的三维面积图，其中第三个轴可用于替换图例：



	A	B	C	D	E
1	Number	Batch 1	Batch 2		
2	2	30	40		
3	3	25	40		
4	4	30	50		
5	5	10	30		
6	6	5	25		
7	7	10	50		

条形图和柱状图

在条形图中，值被绘制为水平条或垂直列。(In bar charts values are plotted as either horizontal bars or vertical columns.)

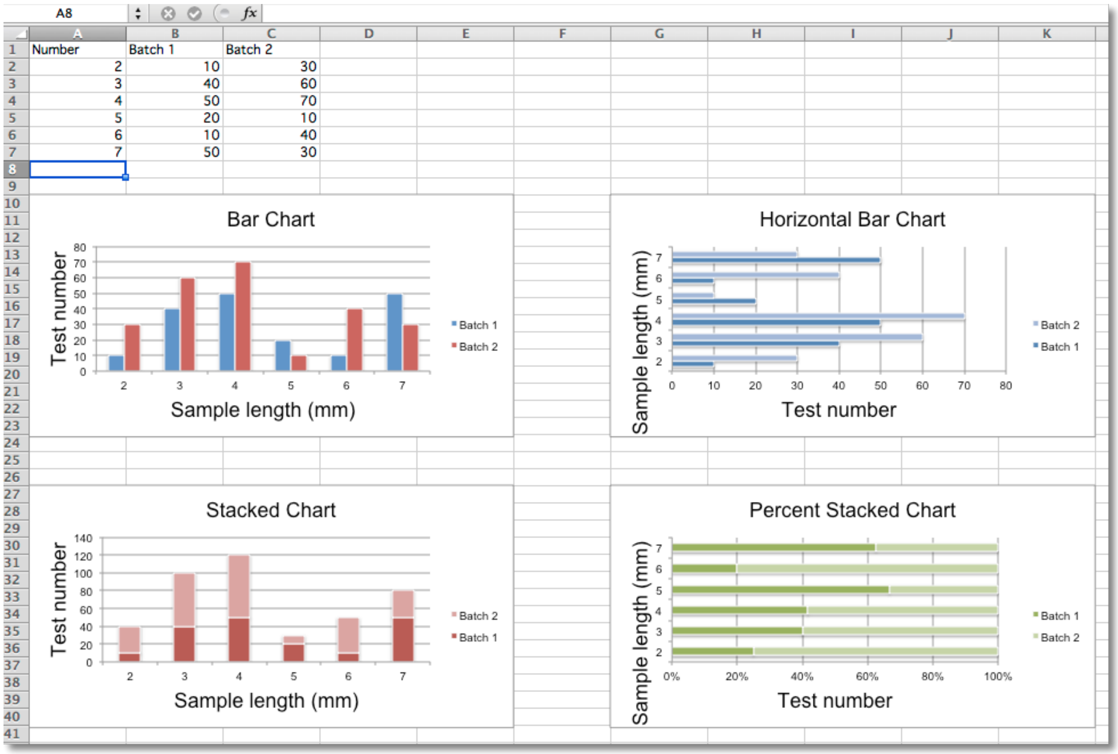
垂直水平和堆叠条形图

注解： 以下设置会影响不同的图表类型。

通过分别将 *type* 设置为 *col* 或 *bar*，可以柱状和水平条形图之间切换。

使用堆叠图表时，需要将 *overlap* 属性设置为 100。

如果条是水平的，则 x 和 y 轴将反转。



A8

	A	B	C
1	Number	Batch 1	Batch 2
2	2	10	30
3	3	40	60
4	4	50	70
5	5	20	10
6	6	10	40
7	7	50	30

```
from openpyxl import Workbook
from openpyxl.chart import BarChart, Series, Reference

wb = Workbook(write_only=True)
ws = wb.create_sheet()

rows = [
    ('Number', 'Batch 1', 'Batch 2'),
    (2, 10, 30),
    (3, 40, 60),
    (4, 50, 70),
    (5, 20, 10),
    (6, 10, 40),
    (7, 50, 30),
]


for row in rows:
    ws.append(row)


chart1 = BarChart()
```

(下页继续)

(续上页)

```
chart1.type = "col"
chart1.style = 10
chart1.title = "Bar Chart"
chart1.y_axis.title = 'Test number'
chart1.x_axis.title = 'Sample length (mm)'

data = Reference(ws, min_col=2, min_row=1, max_row=7, max_col=3)
cats = Reference(ws, min_col=1, min_row=2, max_row=7)
chart1.add_data(data, titles_from_data=True)
chart1.set_categories(cats)
chart1.shape = 4
ws.add_chart(chart1, "A10")

from copy import deepcopy

chart2 = deepcopy(chart1)
chart2.style = 11
chart2.type = "bar"
chart2.title = "Horizontal Bar Chart"

ws.add_chart(chart2, "G10")


chart3 = deepcopy(chart1)
chart3.type = "col"
chart3.style = 12
chart3.grouping = "stacked"
chart3.overlap = 100
chart3.title = 'Stacked Chart'

ws.add_chart(chart3, "A27")


chart4 = deepcopy(chart1)
chart4.type = "bar"
chart4.style = 13
chart4.grouping = "percentStacked"
chart4.overlap = 100
chart4.title = 'Percent Stacked Chart'
```

(下页继续)

(续上页)

```
ws.add_chart(chart4, "G27")

wb.save("bar.xlsx")
```

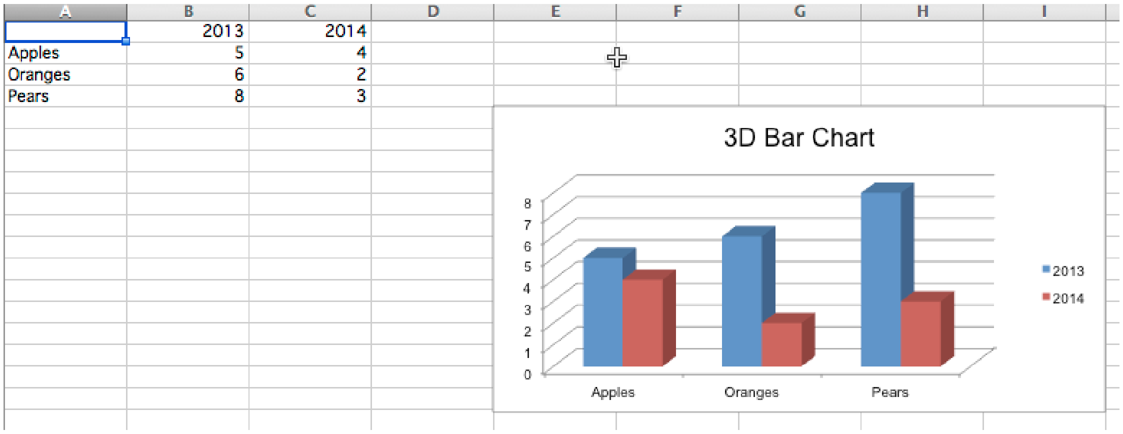
以上创建了四个图表，展示了各种可能性。

三维条形图

你也能创建三维条形图

```
from openpyxl import Workbook
from openpyxl.chart import (
    Reference,
    Series,
    BarChart3D,
)

wb = Workbook()
ws = wb.active

rows = [
    (None, 2013, 2014),
    ("Apples", 5, 4),
    ("Oranges", 6, 2),
    ("Pears", 8, 3)
]

for row in rows:
    ws.append(row)

data = Reference(ws, min_col=2, min_row=1, max_col=3, max_row=4)
titles = Reference(ws, min_col=1, min_row=2, max_row=4)
chart = BarChart3D()
chart.title = "3D Bar Chart"
chart.add_data(data=data, titles_from_data=True)
chart.set_categories(titles)

ws.add_chart(chart, "E5")
wb.save("bar3d.xlsx")
```

这样能创建一个简单的三维条形图



A	B	C
	2013	2014
Apples	5	4
Oranges	6	2
Pears	8	3

气泡图

气泡图类似于散点图但使用第三维来决定气泡的大小，可以包含多个图例。

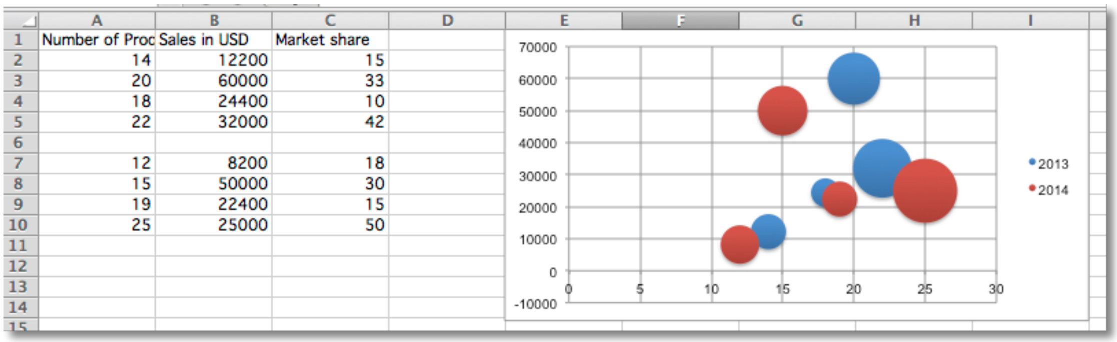
```
"""
Sample bubble chart
"""

from openpyxl import Workbook
from openpyxl.chart import Series, Reference, BubbleChart

wb = Workbook()
ws = wb.active

rows = [
    ("Number of Products", "Sales in USD", "Market share"),
    (14, 12200, 15),
    (20, 60000, 33),
    (18, 24400, 10),
    (22, 32000, 42),
    (),
    (12, 8200, 18),
    (15, 50000, 30),
    (19, 22400, 15),
    (25, 25000, 50),
]

for row in rows:
    ws.append(row)
```

(下页继续)

(续上页)

```

chart = BubbleChart()
chart.style = 18 # use a preset style

# add the first series of data
xvalues = Reference(ws, min_col=1, min_row=2, max_row=5)
yvalues = Reference(ws, min_col=2, min_row=2, max_row=5)
size = Reference(ws, min_col=3, min_row=2, max_row=5)
series = Series(values=yvalues, xvalues=xvalues, zvalues=size, title="2013")
chart.series.append(series)

# add the second
xvalues = Reference(ws, min_col=1, min_row=7, max_row=10)
yvalues = Reference(ws, min_col=2, min_row=7, max_row=10)
size = Reference(ws, min_col=3, min_row=7, max_row=10)
series = Series(values=yvalues, xvalues=xvalues, zvalues=size, title="2014")
chart.series.append(series)

# place the chart starting in cell E1
ws.add_chart(chart, "E1")
wb.save("bubble.xlsx")
```

这会产生一个有两个图例的气泡图，并且样式如下



	A	B	C	D
1	Number of Proc	Sales in USD	Market share	
2	14	12200	15	
3	20	60000	33	
4	18	24400	10	
5	22	32000	42	
6				
7	12	8200	18	
8	15	50000	30	
9	19	22400	15	
10	25	25000	50	
11				
12				
13				
14				
15				

Line Charts

Line Charts

Line charts allow data to be plotted against a fixed axis. They are similar to scatter charts, the main difference is that with line charts each data series is plotted against the same values. Different kinds of axes can be used for the secondary axes.

Similar to bar charts there are three kinds of line charts: standard, stacked and percentStacked.

```
from datetime import date

from openpyxl import Workbook
from openpyxl.chart import (
    LineChart,
    Reference,
)
from openpyxl.chart.axis import DateAxis

wb = Workbook()
ws = wb.active

rows = [
    ['Date', 'Batch 1', 'Batch 2', 'Batch 3'],
    [date(2015,9, 1), 40, 30, 25],
    [date(2015,9, 2), 40, 25, 30],
    [date(2015,9, 3), 50, 30, 45],
    [date(2015,9, 4), 30, 25, 40],
    [date(2015,9, 5), 25, 35, 30],
    [date(2015,9, 6), 20, 40, 35],
]

for row in rows:
    ws.append(row)

c1 = LineChart()
c1.title = "Line Chart"
c1.style = 13
c1.y_axis.title = 'Size'
c1.x_axis.title = 'Test Number'

data = Reference(ws, min_col=2, min_row=1, max_col=4, max_row=7)
c1.add_data(data, titles_from_data=True)

# Style the lines
s1 = c1.series[0]
s1.marker.symbol = "triangle"
s1.marker.graphicalProperties.solidFill = "FF0000" # Marker filling
s1.marker.graphicalProperties.line.solidFill = "FF0000" # Marker outline
```

(下页继续)

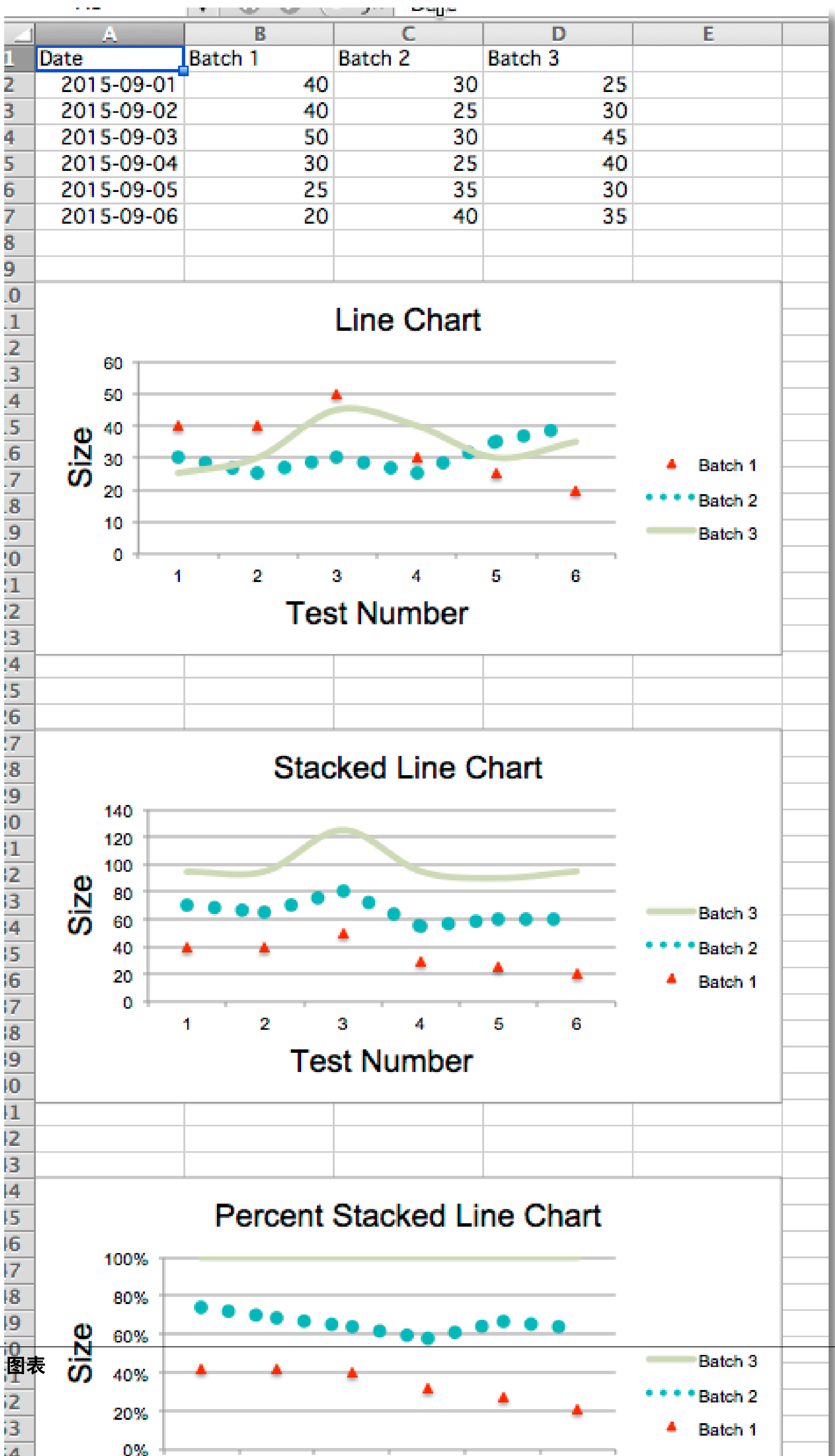
(续上页)

```
s1.graphicalProperties.line.noFill = True

s2 = c1.series[1]
s2.graphicalProperties.line.solidFill = "00AAAA"
s2.graphicalProperties.line.dashStyle = "sysDot"
s2.graphicalProperties.line.width = 100050 # width in EMUs

s2 = c1.series[2]
s2.smooth = True # Make the line smooth

ws.add_chart(c1, "A10")

from copy import deepcopy
stacked = deepcopy(c1)
stacked.grouping = "stacked"
stacked.title = "Stacked Line Chart"
ws.add_chart(stacked, "A27")

percent_stacked = deepcopy(c1)
percent_stacked.grouping = "percentStacked"
percent_stacked.title = "Percent Stacked Line Chart"
ws.add_chart(percent_stacked, "A44")

# Chart with date axis
c2 = LineChart()
c2.title = "Date Axis"
c2.style = 12
c2.y_axis.title = "Size"
c2.y_axis.crossAx = 500
c2.x_axis = DateAxis(crossAx=100)
c2.x_axis.number_format = 'd-mmm'
c2.x_axis.majorTimeUnit = "days"
c2.x_axis.title = "Date"

c2.add_data(data, titles_from_data=True)
dates = Reference(ws, min_col=1, min_row=2, max_row=7)
c2.set_categories(dates)

ws.add_chart(c2, "A61")
```

(下页继续)

(续上页)

```


wb.save("line.xlsx")
```



Date	Batch 1	Batch 2	Batch 3
2015-09-01	40	30	25
2015-09-02	40	25	30
2015-09-03	50	30	45
2015-09-04	30	25	40
2015-09-05	25	35	30
2015-09-06	20	40	35

3D Line Charts

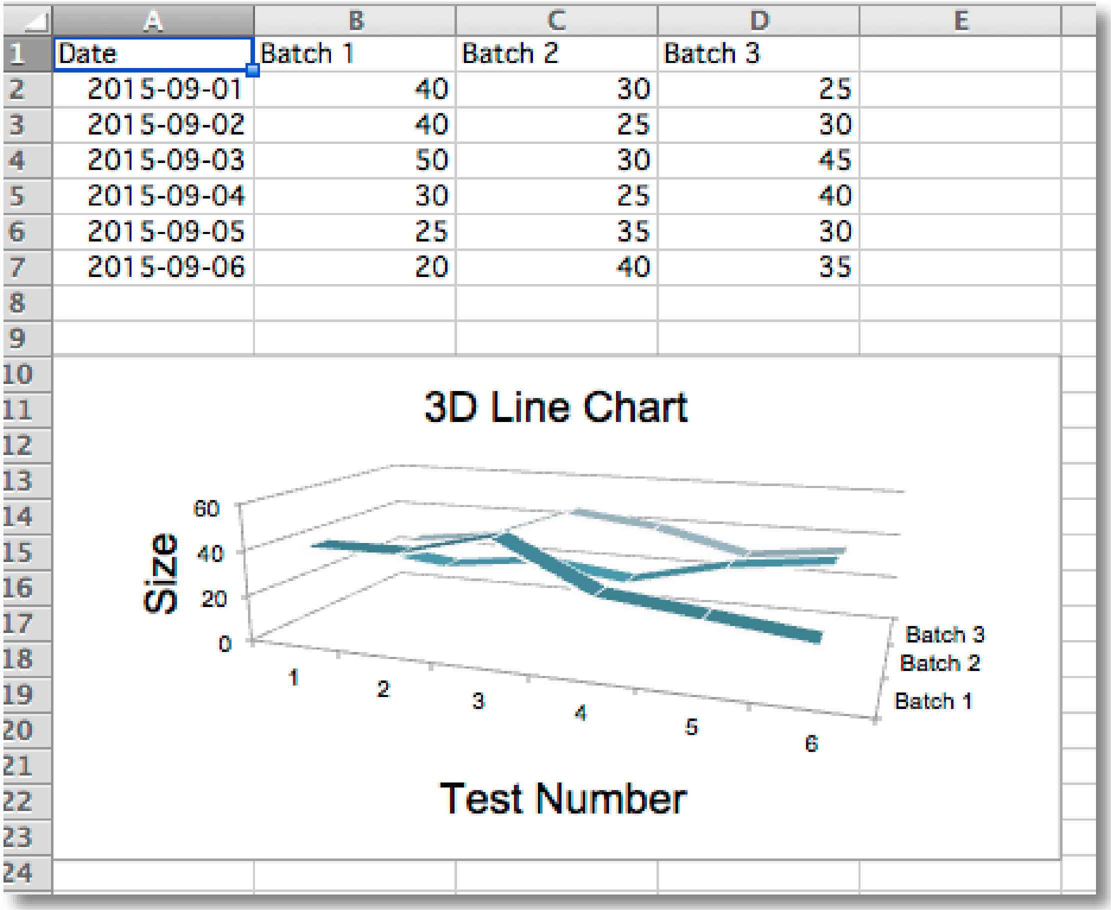
In 3D line charts the third axis is the same as the legend for the series.

```
from datetime import date

from openpyxl import Workbook
from openpyxl.chart import (
    LineChart3D,
    Reference,
)
from openpyxl.chart.axis import DateAxis

wb = Workbook()
ws = wb.active

rows = [
    ['Date', 'Batch 1', 'Batch 2', 'Batch 3'],
    [date(2015,9, 1), 40, 30, 25],
    [date(2015,9, 2), 40, 25, 30],
    [date(2015,9, 3), 50, 30, 45],
    [date(2015,9, 4), 30, 25, 40],
    [date(2015,9, 5), 25, 35, 30],
    [date(2015,9, 6), 20, 40, 35],
]

for row in rows:
    ws.append(row)

c1 = LineChart3D()
c1.title = "3D Line Chart"
c1.legend = None
c1.style = 15
c1.y_axis.title = 'Size'
c1.x_axis.title = 'Test Number'

data = Reference(ws, min_col=2, min_row=1, max_col=4, max_row=7)
c1.add_data(data, titles_from_data=True)

ws.add_chart(c1, "A10")
```

(下页继续)

(续上页)

```
wb.save("line3D.xlsx")
```



	A	B	C	D	E
1	Date	Batch 1	Batch 2	Batch 3	
2	2015-09-01	40	30	25	
3	2015-09-02	40	25	30	
4	2015-09-03	50	30	45	
5	2015-09-04	30	25	40	
6	2015-09-05	25	35	30	
7	2015-09-06	20	40	35	

Scatter Charts

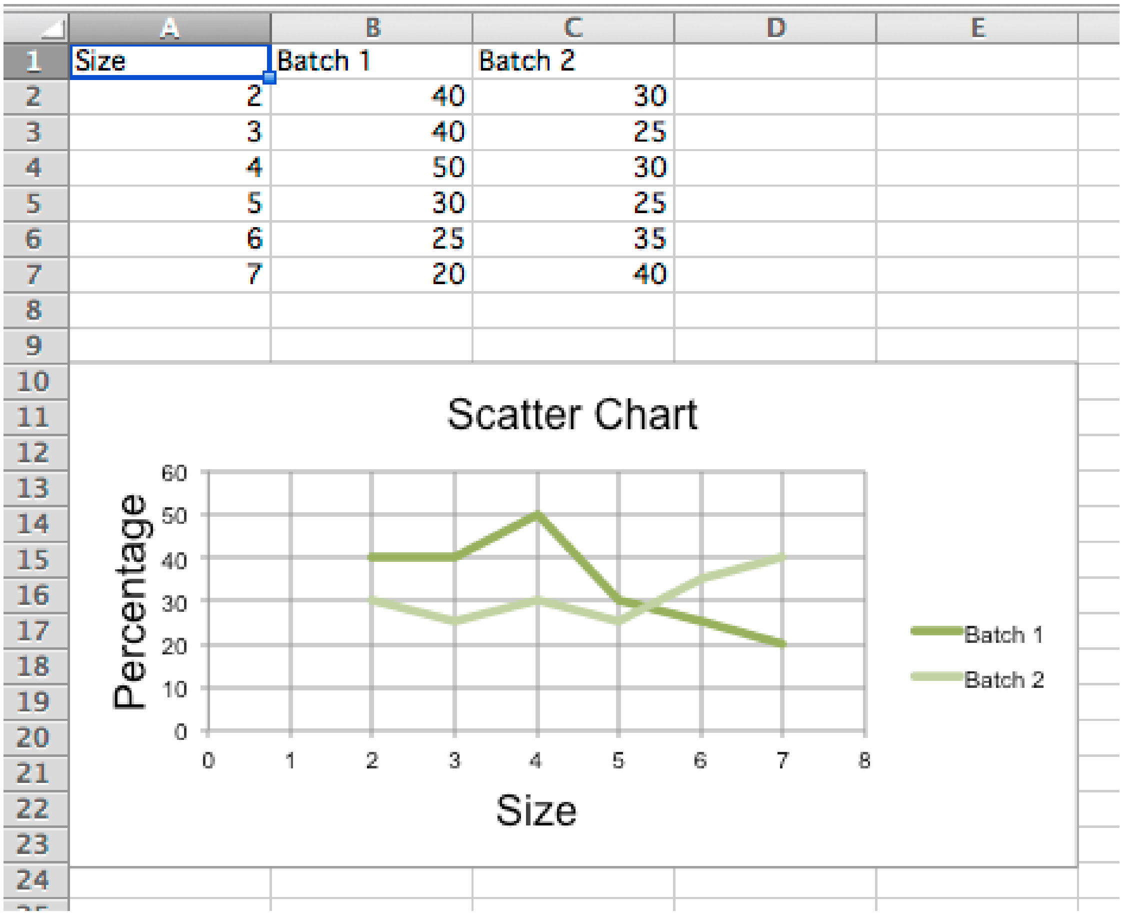
Scatter, or xy, charts are similar to some line charts. The main difference is that one series of values is plotted against another. This is useful where values are unordered.

```
from openpyxl import Workbook
from openpyxl.chart import (
    ScatterChart,
    Reference,
    Series,
)

wb = Workbook()
ws = wb.active
```

(下页继续)

(续上页)

```
rows = [
    ['Size', 'Batch 1', 'Batch 2'],
    [2, 40, 30],
    [3, 40, 25],
    [4, 50, 30],
    [5, 30, 25],
    [6, 25, 35],
    [7, 20, 40],
]

for row in rows:
    ws.append(row)

chart = ScatterChart()
chart.title = "Scatter Chart"
chart.style = 13
chart.x_axis.title = 'Size'
chart.y_axis.title = 'Percentage'

xvalues = Reference(ws, min_col=1, min_row=2, max_row=7)
for i in range(2, 4):
    values = Reference(ws, min_col=i, min_row=1, max_row=7)
    series = Series(values, xvalues, title_from_data=True)
    chart.series.append(series)

ws.add_chart(chart, "A10")

wb.save("scatter.xlsx")
```



	A	B	C	D	E
1	Size	Batch 1	Batch 2		
2	2	40	30		
3	3	40	25		
4	4	50	30		
5	5	30	25		
6	6	25	35		
7	7	20	40		

注解: The specification says that there are the following types of scatter charts: ‘line’ , ‘lineMarker’ , ‘marker’ , ‘smooth’ , ‘smoothMarker’ . However, at least in Microsoft Excel, this is just a shortcut for other settings that otherwise have no effect. For consistency with line charts, the style for each series should be set manually.

Pie Charts

Pie Charts

Pie charts plot data as slices of a circle with each slice representing the percentage of the whole. Slices are plotted in a clockwise direction with 0° being at the top of the circle. Pie charts can only take a single series of data. The title of the chart will default to being the title of the series.

```
from openpyxl import Workbook

from openpyxl.chart import (
```

(下页继续)

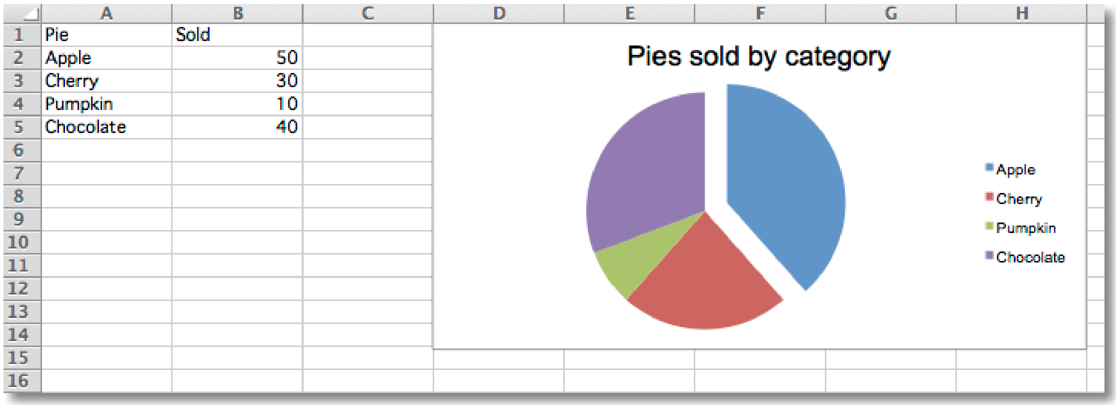
(续上页)

```
    PieChart,
    ProjectedPieChart,
    Reference
)
from openpyxl.chart.series import DataPoint

data = [
    ['Pie', 'Sold'],
    ['Apple', 50],
    ['Cherry', 30],
    ['Pumpkin', 10],
    ['Chocolate', 40],
]

wb = Workbook()
ws = wb.active

for row in data:
    ws.append(row)

pie = PieChart()
labels = Reference(ws, min_col=1, min_row=2, max_row=5)
data = Reference(ws, min_col=2, min_row=1, max_row=5)
pie.add_data(data, titles_from_data=True)
pie.set_categories(labels)
pie.title = "Pies sold by category"

# Cut the first slice out of the pie
slice = DataPoint(idx=0, explosion=20)
pie.series[0].data_points = [slice]

ws.add_chart(pie, "D1")


ws = wb.create_sheet(title="Projection")

data = [
    ['Page', 'Views'],
    ['Search', 95],
    ['Products', 4],
```

(下页继续)

(续上页)

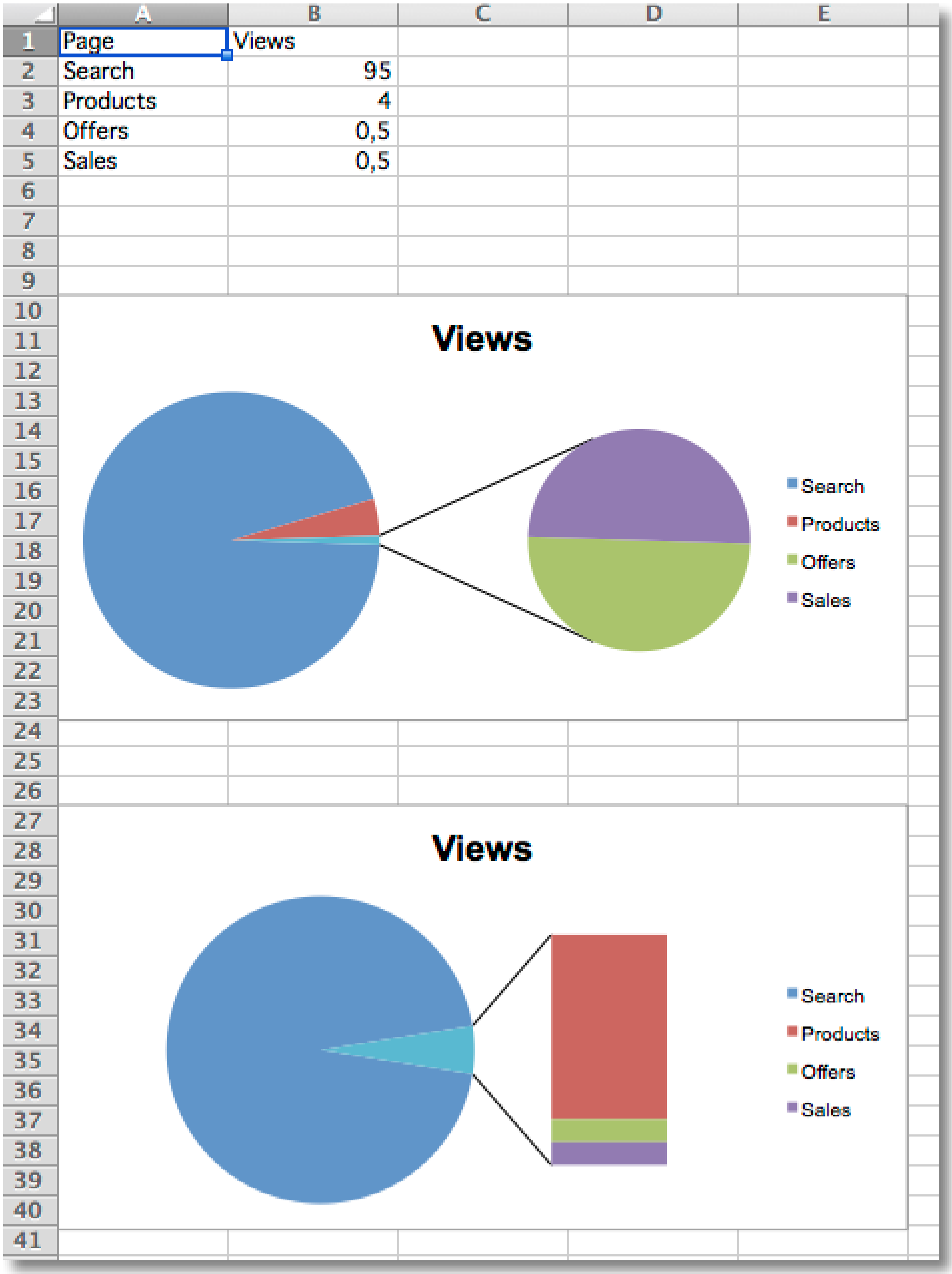
```
    ['Offers', 0.5],
    ['Sales', 0.5],
]

for row in data:
    ws.append(row)

projected_pie = ProjectedPieChart()
projected_pie.type = "pie"
projected_pie.splitType = "val" # split by value
labels = Reference(ws, min_col=1, min_row=2, max_row=5)
data = Reference(ws, min_col=2, min_row=1, max_row=5)
projected_pie.add_data(data, titles_from_data=True)
projected_pie.set_categories(labels)

ws.add_chart(projected_pie, "A10")

from copy import deepcopy
projected_bar = deepcopy(projected_pie)
projected_bar.type = "bar"
projected_bar.splitType = 'pos' # split by position

ws.add_chart(projected_bar, "A27")

wb.save("pie.xlsx")
```



	A	B	C
1	Pie	Sold	
2	Apple	50	
3	Cherry	30	
4	Pumpkin	10	
5	Chocolate	40	

Projected Pie Charts

Projected pie charts extract some slices from a pie chart and project them into a second pie or bar chart. This is useful when there are several smaller items in the data series. The chart can be split according to percent, val(ue) or pos(ition). If nothing is set then the application decides which to use. In addition custom splits can be defined.

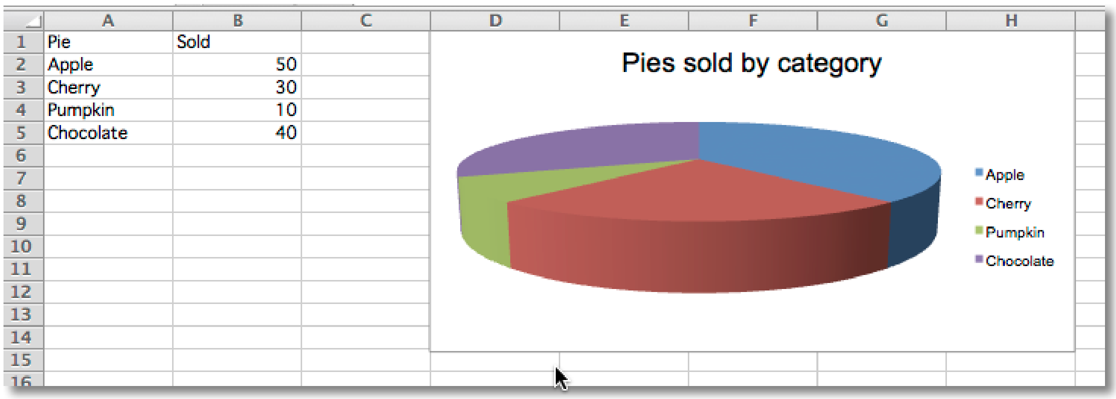


Page	Views
Search	95
Products	4
Offers	0,5
Sales	0,5

3D Pie Charts

Pie charts can also be created with a 3D effect.

```python
from openpyxl import Workbook

from openpyxl.chart import (
    PieChart3D,
    Reference
)

data = [
    ['Pie', 'Sold'],
    ['Apple', 50],
    ['Cherry', 30],
    ['Pumpkin', 10],
    ['Chocolate', 40],
]

wb = Workbook()
ws = wb.active

for row in data:
    ws.append(row)

pie = PieChart3D()
labels = Reference(ws, min_col=1, min_row=2, max_row=5)
data = Reference(ws, min_col=2, min_row=1, max_row=5)
pie.add_data(data, titles_from_data=True)
pie.set_categories(labels)
pie.title = "Pies sold by category"


ws.add_chart(pie, "D1")

wb.save("pie3D.xlsx")
```

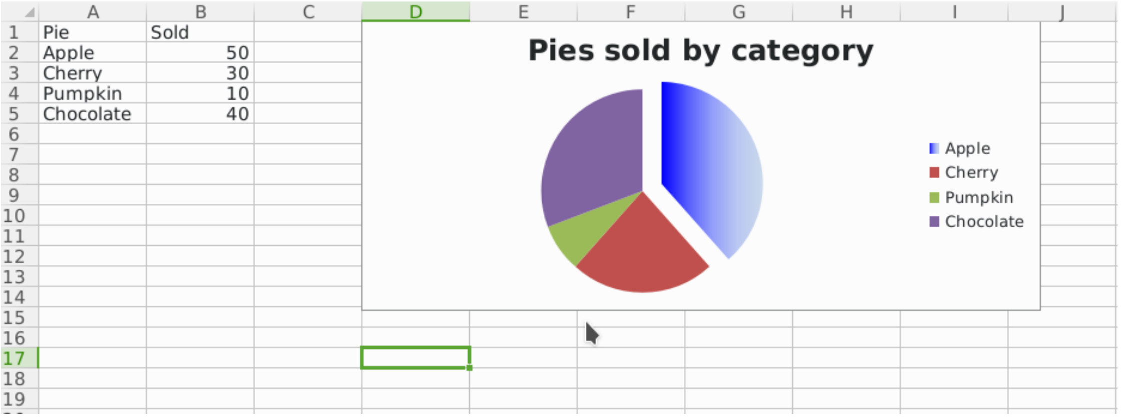


Pie	Sold
Apple	50
Cherry	30
Pumpkin	10
Chocolate	40

Gradient Pie Charts

Pie charts can also be created with gradient series.

..literalinclude:: pie-gradient.py



Pie	Sold
Apple	50
Cherry	30
Pumpkin	10
Chocolate	40

旭日图

旭日图和饼图相似，不同之处在于使用的是圆形还是圆环。他们还可以将多个系列的数据绘制为同心环。

```
from openpyxl import Workbook

from openpyxl.chart import (
    DoughnutChart,
    Reference,
    Series,
)
from openpyxl.chart.series import DataPoint
```

（下页继续）

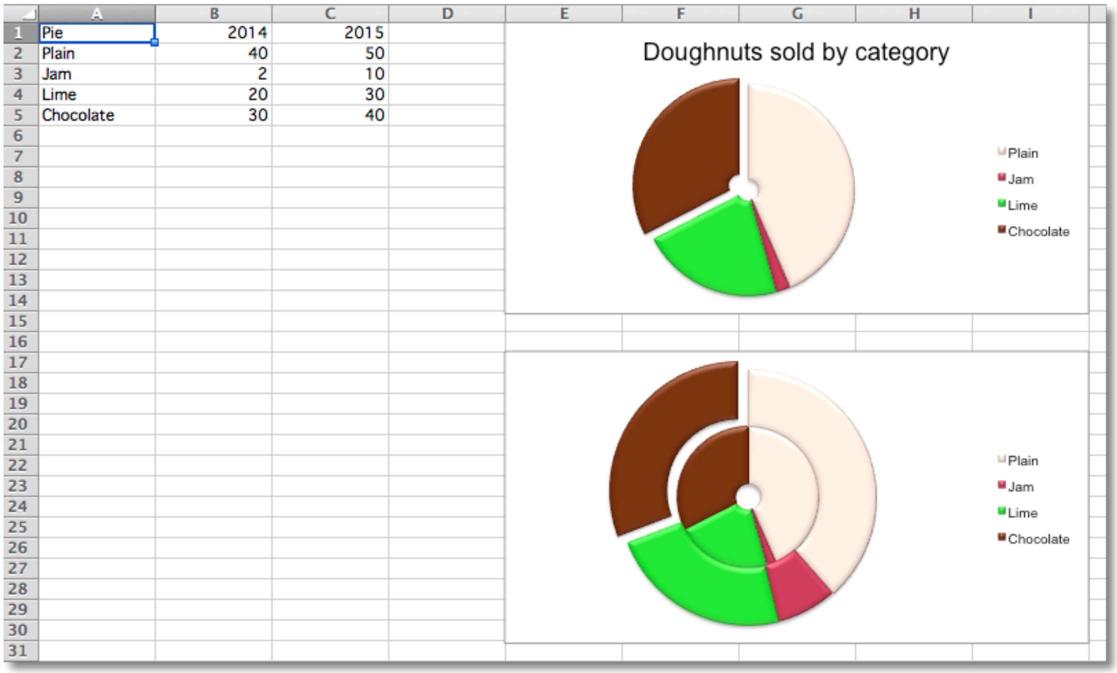
(续上页)

```
data = [
    ['Pie', 2014, 2015],
    ['Plain', 40, 50],
    ['Jam', 2, 10],
    ['Lime', 20, 30],
    ['Chocolate', 30, 40],
]

wb = Workbook()
ws = wb.active

for row in data:
    ws.append(row)

chart = DoughnutChart()
labels = Reference(ws, min_col=1, min_row=2, max_row=5)
data = Reference(ws, min_col=2, min_row=1, max_row=5)
chart.add_data(data, titles_from_data=True)
chart.set_categories(labels)
chart.title = "Doughnuts sold by category"
chart.style = 26

# Cut the first slice out of the doughnut
slices = [DataPoint(idx=i) for i in range(4)]
plain, jam, lime, chocolate = slices
chart.series[0].data_points = slices
plain.graphicalProperties.solidFill = "FAE1D0"
jam.graphicalProperties.solidFill = "BB2244"
lime.graphicalProperties.solidFill = "22DD22"
chocolate.graphicalProperties.solidFill = "61210B"
chocolate.explosion = 10

ws.add_chart(chart, "E1")

from copy import deepcopy

chart2 = deepcopy(chart)
chart2.title = None
data = Reference(ws, min_col=3, min_row=1, max_row=5)
series2 = Series(data, title_from_data=True)
```

(下页继续)

(续上页)

```
series2.data_points = slices
chart2.series.append(series2)

ws.add_chart(chart2, "E17")

wb.save("doughnut.xlsx")
```



	A	B	C
1	Pie	2014	2015
2	Plain	40	50
3	Jam	2	10
4	Lime	20	30
5	Chocolate	30	40

Radar Charts

Data that is arranged in columns or rows on a worksheet can be plotted in a radar chart. Radar charts compare the aggregate values of multiple data series. It is effectively a projection of an area chart on a circular x-axis.

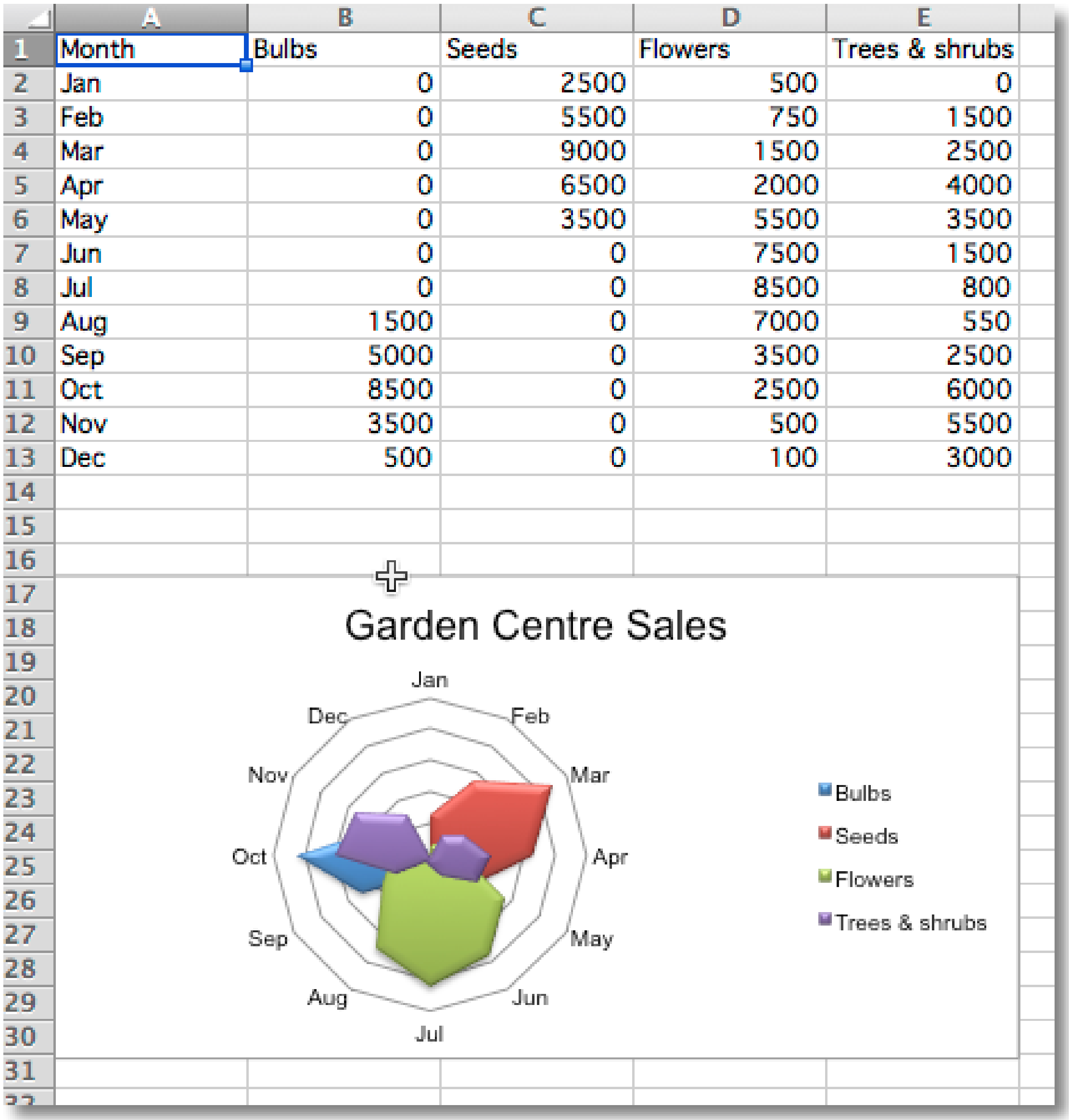
There are two types of radar chart: standard, where the area is marked with a line; and filled, where the whole area is filled. The additional type "marker" has no effect. If markers are desired these can be set for the relevant series.

```
from openpyxl import Workbook
from openpyxl.chart import (
    RadarChart,
    Reference,
)

```

(下页继续)

(续上页)

```
wb = Workbook()
ws = wb.active

rows = [
    ['Month', "Bulbs", "Seeds", "Flowers", "Trees & shrubs"],
    ['Jan', 0, 2500, 500, 0,],
    ['Feb', 0, 5500, 750, 1500],
    ['Mar', 0, 9000, 1500, 2500],
    ['Apr', 0, 6500, 2000, 4000],
    ['May', 0, 3500, 5500, 3500],
    ['Jun', 0, 0, 7500, 1500],
    ['Jul', 0, 0, 8500, 800],
    ['Aug', 1500, 0, 7000, 550],
    ['Sep', 5000, 0, 3500, 2500],
    ['Oct', 8500, 0, 2500, 6000],
    ['Nov', 3500, 0, 500, 5500],
    ['Dec', 500, 0, 100, 3000 ],
]

for row in rows:
    ws.append(row)

chart = RadarChart()
chart.type = "filled"
labels = Reference(ws, min_col=1, min_row=2, max_row=13)
data = Reference(ws, min_col=2, max_col=5, min_row=1, max_row=13)
chart.add_data(data, titles_from_data=True)
chart.set_categories(labels)
chart.style = 26
chart.title = "Garden Centre Sales"
chart.y_axis.delete = True

ws.add_chart(chart, "A17")

wb.save("radar.xlsx")
```



Month	Bulbs	Seeds	Flowers	Trees & shrubs
Jan	0	2500	500	0
Feb	0	5500	750	1500
Mar	0	9000	1500	2500
Apr	0	6500	2000	4000
May	0	3500	5500	3500
Jun	0	0	7500	1500
Jul	0	0	8500	800
Aug	1500	0	7000	550
Sep	5000	0	3500	2500
Oct	8500	0	2500	6000
Nov	3500	0	500	5500
Dec	500	0	100	3000

Stock Charts

Data that is arranged in columns or rows in a specific order on a worksheet can be plotted in a stock chart. As its name implies, a stock chart is most often used to illustrate the fluctuation of stock prices. However, this chart may also be used for scientific data. For example, you could use a stock chart to indicate the fluctuation of daily or annual temperatures. You must organize your data in the correct order to create stock charts.

The way stock chart data is organized in the worksheet is very important. For example, to create a simple high-low-close stock chart, you should arrange your data with High, Low, and Close entered as column headings, in that order.

Although stock charts are a distinct type, the various types are just shortcuts for particular formatting options:

- high-low-close is essentially a line chart with no lines and the marker set to XYZ. It also sets hiLoLines to True
- open-high-low-close is the same as a high-low-close chart with the marker for each data point set to XZZ and upDownLines.

Volume can be added by combining the stock chart with a bar chart for the volume.

```
from datetime import date

from openpyxl import Workbook

from openpyxl.chart import (
    BarChart,
    StockChart,
    Reference,
    Series,
)
from openpyxl.chart.axis import DateAxis, ChartLines
from openpyxl.chart.updown_bars import UpDownBars

wb = Workbook()
ws = wb.active

rows = [
   ['Date',      'Volume','Open', 'High', 'Low', 'Close'],
   ['2015-01-01', 20000,    26.2, 27.20, 23.49, 25.45,  ],
   ['2015-01-02', 10000,    25.45, 25.03, 19.55, 23.05, ],
   ['2015-01-03', 15000,    23.05, 24.46, 20.03, 22.42, ],
   ['2015-01-04', 2000,     22.42, 23.97, 20.07, 21.90, ],
   ['2015-01-05', 12000,    21.9, 23.65, 19.50, 21.51,  ],
]

for row in rows:
    ws.append(row)

# High-low-close
c1 = StockChart()
labels = Reference(ws, min_col=1, min_row=2, max_row=6)
data = Reference(ws, min_col=4, max_col=6, min_row=1, max_row=6)
c1.add_data(data, titles_from_data=True)
```

(下页继续)

(续上页)

```
c1.set_categories(labels)
for s in c1.series:
    s.graphicalProperties.line.noFill = True
# marker for close
s.marker.symbol = "dot"
s.marker.size = 5
c1.title = "High-low-close"
c1.hiLowLines = ChartLines()

# Excel is broken and needs a cache of values in order to display hiLoLines :-/
from openpyxl.chart.data_source import NumData, NumVal
pts = [NumVal(idx=i) for i in range(len(data) - 1)]
cache = NumData(pt=pts)
c1.series[-1].val.numRef.numCache = cache

ws.add_chart(c1, "A10")

# Open-high-low-close
c2 = StockChart()
data = Reference(ws, min_col=3, max_col=6, min_row=1, max_row=6)
c2.add_data(data, titles_from_data=True)
c2.set_categories(labels)
for s in c2.series:
    s.graphicalProperties.line.noFill = True
c2.hiLowLines = ChartLines()
c2.upDownBars = UpDownBars()
c2.title = "Open-high-low-close"

# add dummy cache
c2.series[-1].val.numRef.numCache = cache

ws.add_chart(c2, "G10")

# Create bar chart for volume

bar = BarChart()
data =  Reference(ws, min_col=2, min_row=1, max_row=6)
bar.add_data(data, titles_from_data=True)
bar.set_categories(labels)
```

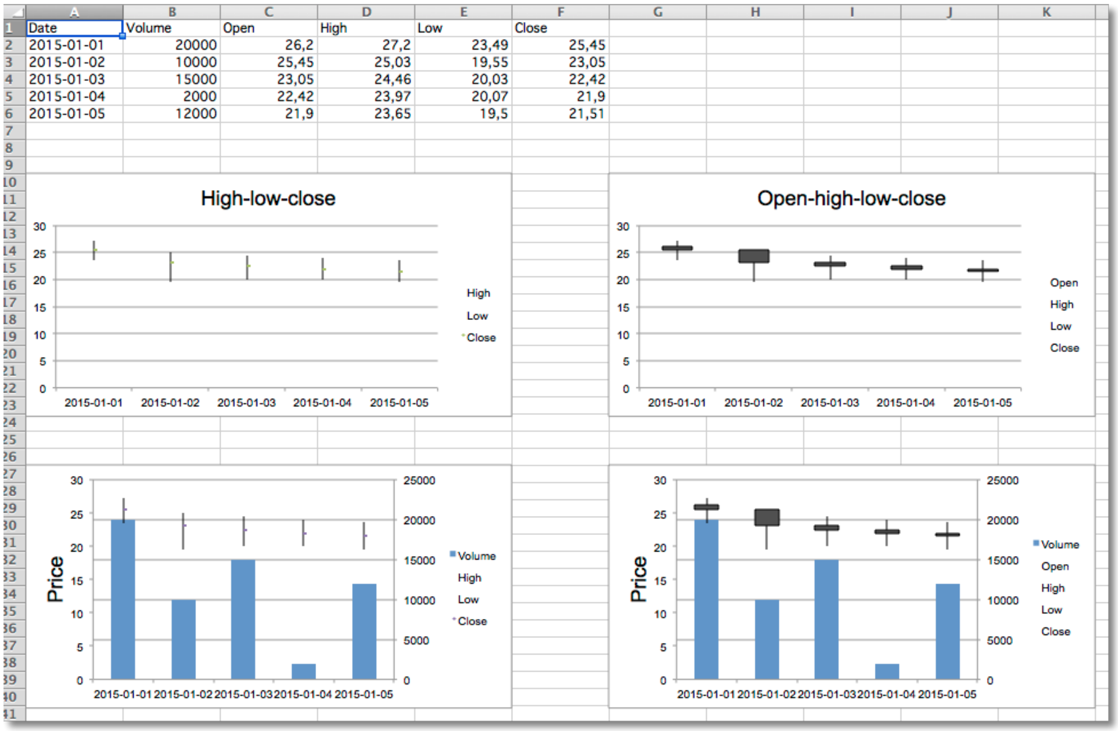
(下页继续)

(续上页)

```
from copy import deepcopy

# Volume-high-low-close
b1 = deepcopy(bar)
c3 = deepcopy(c1)
c3.y_axis.majorGridlines = None
c3.y_axis.title = "Price"
b1.y_axis.axId = 20
b1.z_axis = c3.y_axis
b1.y_axis.crosses = "max"
b1 += c3

c3.title = "High low close volume"

ws.add_chart(b1, "A27")

## Volume-open-high-low-close
b2 = deepcopy(bar)
c4 = deepcopy(c2)
c4.y_axis.majorGridlines = None
c4.y_axis.title = "Price"
b2.y_axis.axId = 20
b2.z_axis = c4.y_axis
b2.y_axis.crosses = "max"
b2 += c4

ws.add_chart(b2, "G27")

wb.save("stock.xlsx")
```

警告: Due to a bug in Excel high-low lines will only be shown if at least one of the data series has some dummy values. This can be done with the following hack:

```
from openpyxl.chart.data_source import NumData, NumVal
pts = [NumVal(idx=i) for i in range(len(data) - 1)]
cache = NumData(pt=pts)
c1.series[-1].val.numRef.numCache = cache
```



Date	Volume	Open	High	Low	Close
2015-01-01	20000	26,2	27,2	23,49	25,45
2015-01-02	10000	25,45	25,03	19,55	23,05
2015-01-03	15000	23,05	24,46	20,03	22,42
2015-01-04	2000	22,42	23,97	20,07	21,9
2015-01-05	12000	21,9	23,65	19,5	21,51

Surface charts

Data that is arranged in columns or rows on a worksheet can be plotted in a surface chart. A surface chart is useful when you want to find optimum combinations between two sets of data. As in a topographic map, colors and patterns indicate areas that are in the same range of values.

By default all surface charts are 3D. 2D wireframe and contour charts are created by setting the rotation and perspective.

```
from openpyxl import Workbook
from openpyxl.chart import (
    SurfaceChart,
    SurfaceChart3D,
    Reference,
    Series,
)
from openpyxl.chart.axis import SeriesAxis

wb = Workbook()
ws = wb.active

data = [
    [None, 10, 20, 30, 40, 50,],
```

(下页继续)

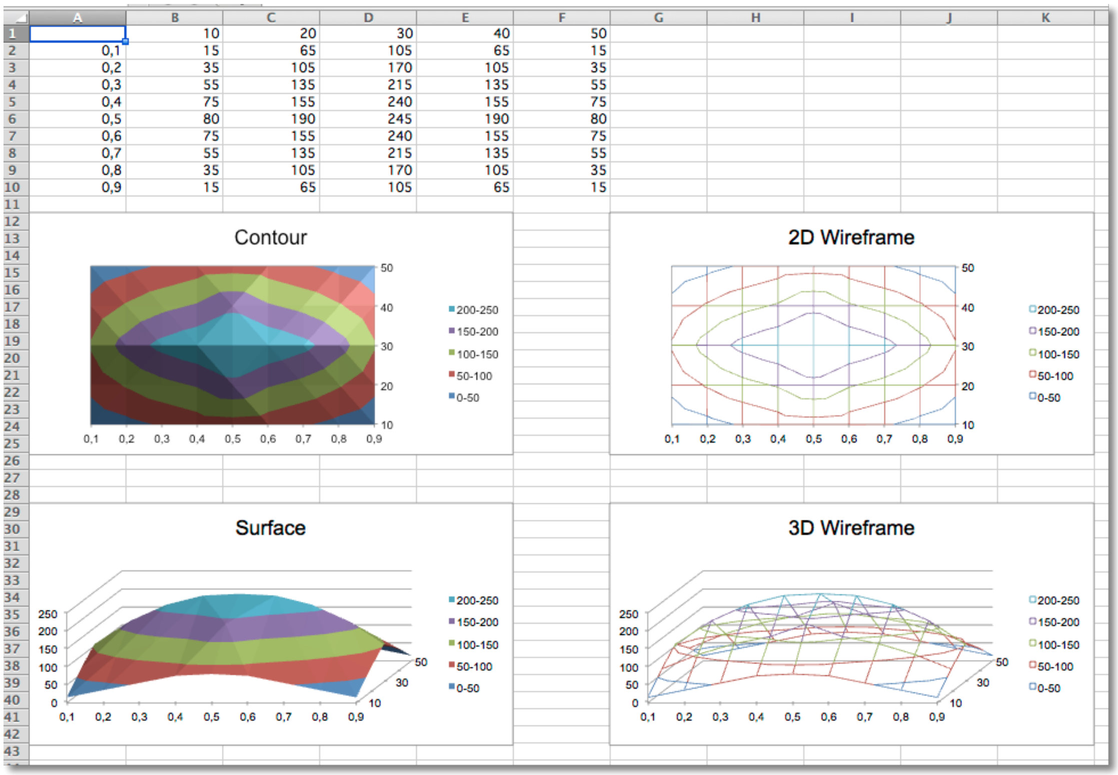
(续上页)

```
    [0.1, 15, 65, 105, 65, 15,],
    [0.2, 35, 105, 170, 105, 35,],
    [0.3, 55, 135, 215, 135, 55,],
    [0.4, 75, 155, 240, 155, 75,],
    [0.5, 80, 190, 245, 190, 80,],
    [0.6, 75, 155, 240, 155, 75,],
    [0.7, 55, 135, 215, 135, 55,],
    [0.8, 35, 105, 170, 105, 35,],
    [0.9, 15, 65, 105, 65, 15],
]

for row in data:
    ws.append(row)


c1 = SurfaceChart()
ref = Reference(ws, min_col=2, max_col=6, min_row=1, max_row=10)
labels = Reference(ws, min_col=1, min_row=2, max_row=10)
c1.add_data(ref, titles_from_data=True)
c1.set_categories(labels)
c1.title = "Contour"

ws.add_chart(c1, "A12")

from copy import deepcopy

# wireframe
c2 = deepcopy(c1)
c2.wireframe = True
c2.title = "2D Wireframe"

ws.add_chart(c2, "G12")

# 3D Surface
c3 = SurfaceChart3D()
c3.add_data(ref, titles_from_data=True)
c3.set_categories(labels)
c3.title = "Surface"

ws.add_chart(c3, "A29")
```

(下页继续)

(续上页)

```
c4 = deepcopy(c3)
c4.wireframe = True
c4.title = "3D Wireframe"

ws.add_chart(c4, "G29")

wb.save("surface.xlsx")
```



	A	B	C	D	E	F
1		10	20	30	40	50
2	0,1	15	65	105	65	15
3	0,2	35	105	170	105	35
4	0,3	55	135	215	135	55
5	0,4	75	155	240	155	75
6	0,5	80	190	245	190	80
7	0,6	75	155	240	155	75
8	0,7	55	135	215	135	55
9	0,8	35	105	170	105	35
10	0,9	15	65	105	65	15

6.4.2 创建图表

图表由至少一个系列的一个或多个数据点组成。系列由单元格范围的引用组成。

```
>>> from openpyxl import Workbook
>>> wb = Workbook()
>>> ws = wb.active
>>> for i in range(10):
...     ws.append([i])
>>>
>>> from openpyxl.chart import BarChart, Reference, Series
```

(下页继续)

(续上页)

```
>>> values = Reference(ws, min_col=1, min_row=1, max_col=1, max_row=10)
>>> chart = BarChart()
>>> chart.add_data(values)
>>> ws.add_chart(chart, "E15")
>>> wb.save("SampleChart.xlsx")
```

默认情况下，图表的左上角固定在单元格 E15 上，大小为 15 x 7.5 厘米（大约 5 列乘 14 行）。可以通过设置图表的 *anchor*，*width* 和 *height* 属性来更改。实际大小将取决于操作系统和设备。其他锚点（anchors ）也是有可能的。更多资料请参考 *openpyxl.drawing.spreadsheet_drawing*。

6.4.3 使用轴

Axis Limits and Scale

Minima and Maxima

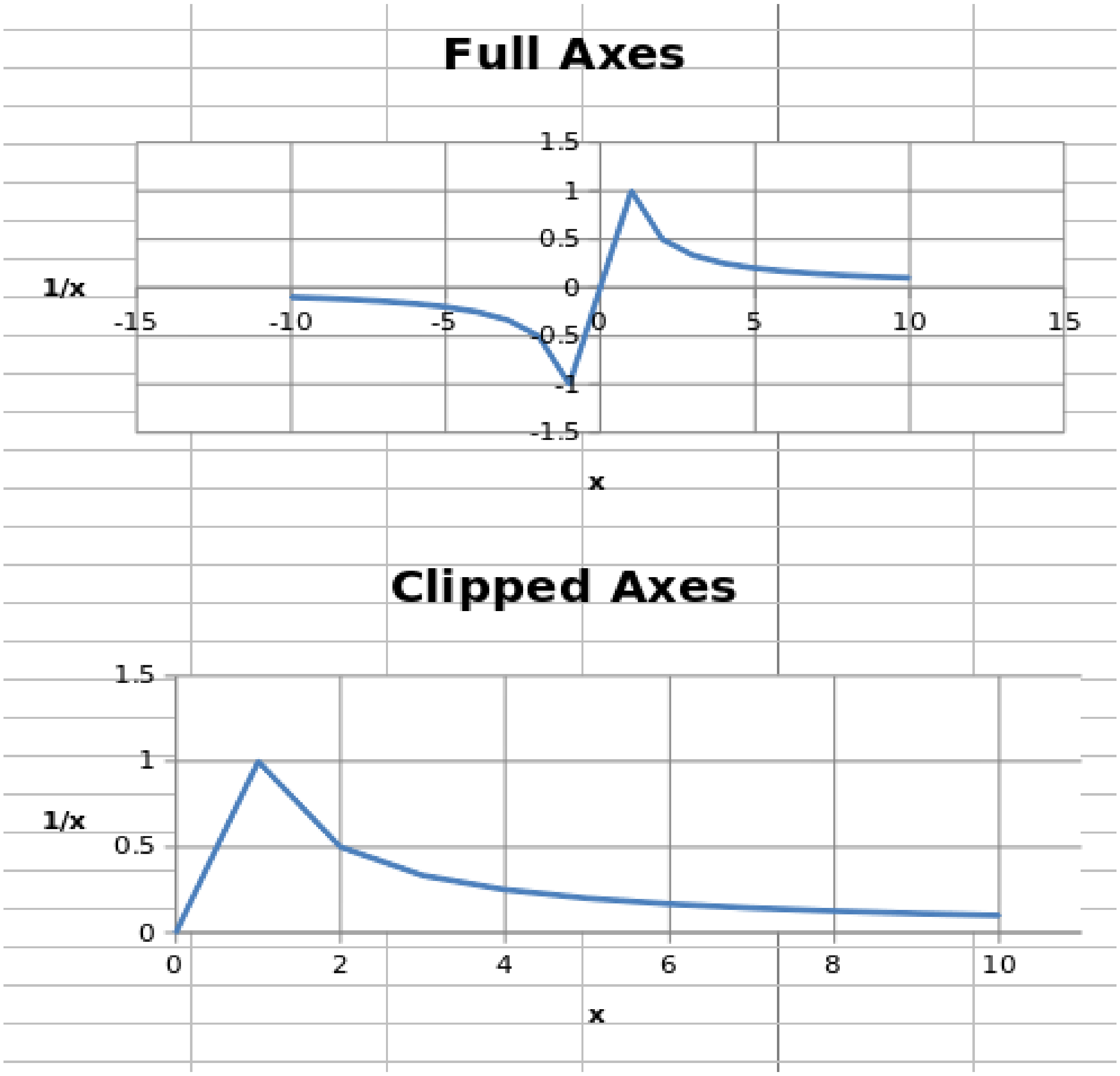
Axis minimum and maximum values can be set manually to display specific regions on a chart.

```
from openpyxl import Workbook
from openpyxl.chart import (
    ScatterChart,
    Reference,
    Series,
)

wb = Workbook()
ws = wb.active

ws.append(['X', '1/X'])
for x in range(-10, 11):
    if x:
        ws.append([x, 1.0 / x])

chart1 = ScatterChart()
chart1.title = "Full Axes"
chart1.x_axis.title = 'x'
chart1.y_axis.title = '1/x'
chart1.legend = None

chart2 = ScatterChart()
```

(下页继续)

(续上页)

```
chart2.title = "Clipped Axes"
chart2.x_axis.title = 'x'
chart2.y_axis.title = '1/x'
chart2.legend = None

chart2.x_axis.scaling.min = 0
chart2.y_axis.scaling.min = 0
chart2.x_axis.scaling.max = 11
chart2.y_axis.scaling.max = 1.5

x = Reference(ws, min_col=1, min_row=2, max_row=22)
y = Reference(ws, min_col=2, min_row=2, max_row=22)
s = Series(y, xvalues=x)
chart1.append(s)
chart2.append(s)

ws.add_chart(chart1, "C1")
ws.add_chart(chart2, "C15")

wb.save("minmax.xlsx")
```



注解: In some cases such as the one shown, setting the axis limits is effectively equivalent to displaying a sub-range of the data. For large datasets, rendering of scatter plots (and possibly others) will be much faster when using subsets of the data rather than axis limits in both Excel and Open/Libre Office.

Logarithmic Scaling

Both the x- and y-axes can be scaled logarithmically. The base of the logarithm can be set to any valid float. If the x-axis is scaled logarithmically, negative values in the domain will be discarded.

```
from openpyxl import Workbook
from openpyxl.chart import (
```

(下页继续)

(续上页)

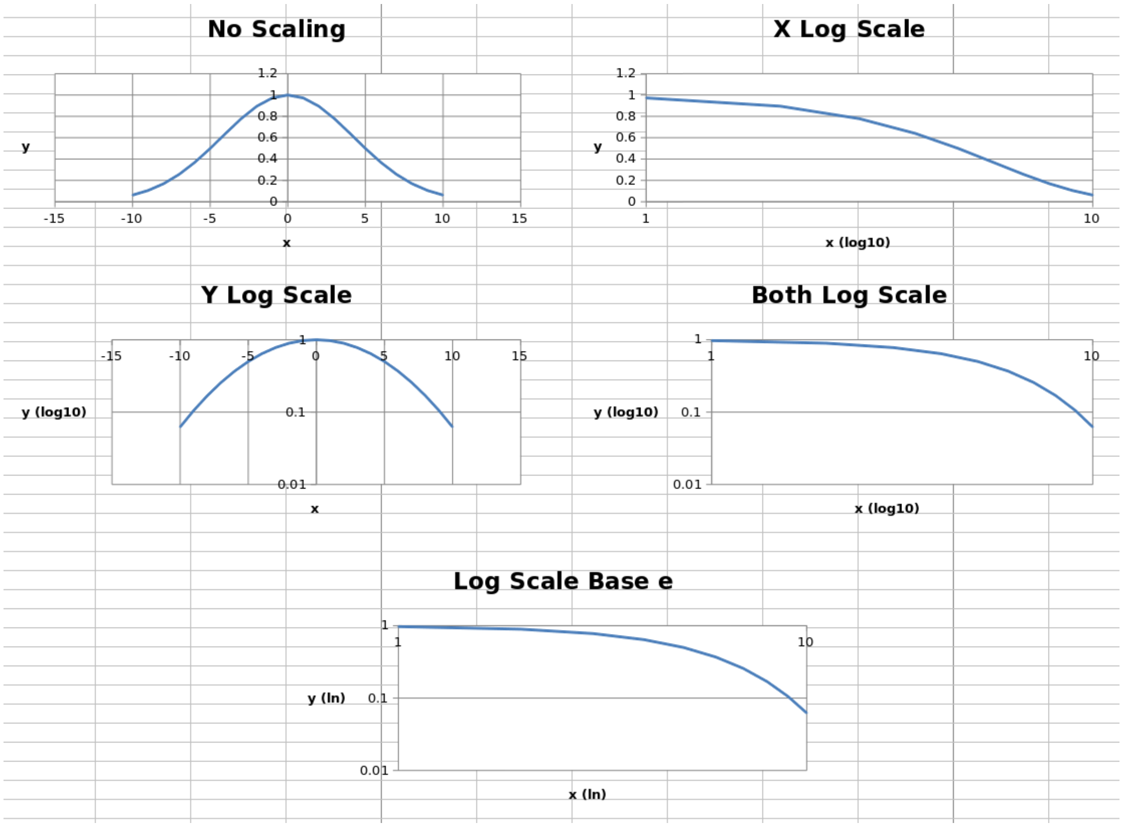
```
    ScatterChart,
    Reference,
    Series,
)
import math

wb = Workbook()
ws = wb.active

ws.append(['X', 'Gaussian'])
for i, x in enumerate(range(-10, 11)):
    ws.append([x, "=EXP(-(($A${row}/6)^2))".format(row = i + 2)])

chart1 = ScatterChart()
chart1.title = "No Scaling"
chart1.x_axis.title = 'x'
chart1.y_axis.title = 'y'
chart1.legend = None

chart2 = ScatterChart()
chart2.title = "X Log Scale"
chart2.x_axis.title = 'x (log10)'
chart2.y_axis.title = 'y'
chart2.legend = None
chart2.x_axis.scaling.logBase = 10

chart3 = ScatterChart()
chart3.title = "Y Log Scale"
chart3.x_axis.title = 'x'
chart3.y_axis.title = 'y (log10)'
chart3.legend = None
chart3.y_axis.scaling.logBase = 10

chart4 = ScatterChart()
chart4.title = "Both Log Scale"
chart4.x_axis.title = 'x (log10)'
chart4.y_axis.title = 'y (log10)'
chart4.legend = None
chart4.x_axis.scaling.logBase = 10
chart4.y_axis.scaling.logBase = 10
```

(下页继续)

(续上页)

```

chart5 = ScatterChart()
chart5.title = "Log Scale Base e"
chart5.x_axis.title = 'x (ln)'
chart5.y_axis.title = 'y (ln)'
chart5.legend = None
chart5.x_axis.scaling.logBase = math.e
chart5.y_axis.scaling.logBase = math.e

x = Reference(ws, min_col=1, min_row=2, max_row=22)
y = Reference(ws, min_col=2, min_row=2, max_row=22)
s = Series(y, xvalues=x)
chart1.append(s)
chart2.append(s)
chart3.append(s)
chart4.append(s)
chart5.append(s)

ws.add_chart(chart1, "C1")
ws.add_chart(chart2, "I1")
ws.add_chart(chart3, "C15")
ws.add_chart(chart4, "I15")
ws.add_chart(chart5, "F30")

wb.save("log.xlsx")
```

This produces five charts that look something like this:



The first four charts show the same data unscaled, scaled logarithmically in each axis and in both axes, with the logarithm base set to 10. The final chart shows the same data with both axes scaled, but the base of the logarithm set to `e`.

Axis Orientation

Axes can be displayed "normally" or in reverse. Axis orientation is controlled by the scaling `orientation` property, which can have a value of either `'minMax'` for normal orientation or `'maxMin'` for reversed.

```
from openpyxl import Workbook
from openpyxl.chart import (
    ScatterChart,
    Reference,
    Series,
)

wb = Workbook()
ws = wb.active
```

(下页继续)

(续上页)

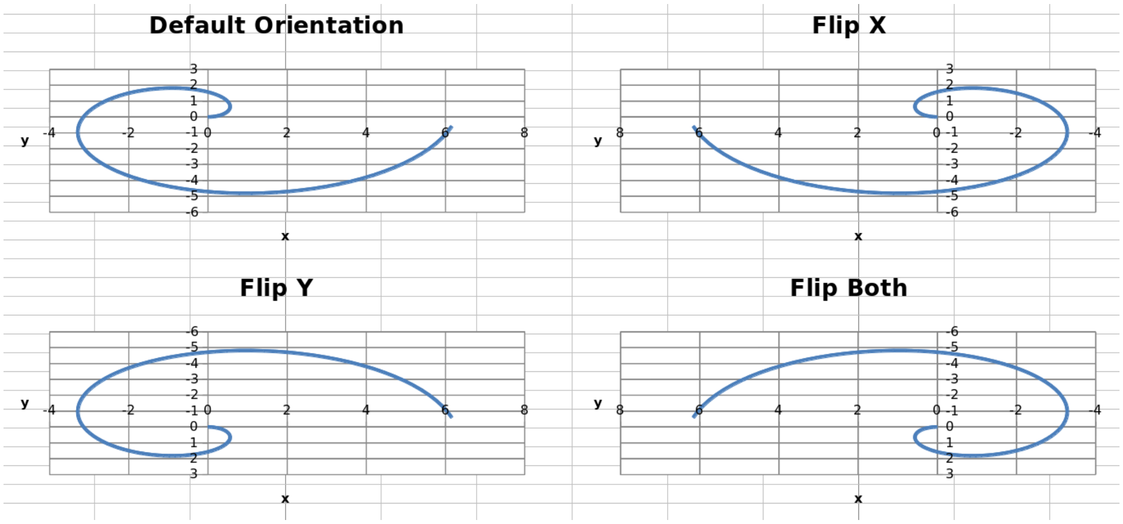
```
ws["A1"] = "Archimedean Spiral"
ws.append(["T", "X", "Y"])
for i, t in enumerate(range(100)):
    ws.append([t / 16.0, "=$A${row}*COS($A${row})".format(row = i + 3),
                         "=$A${row}*SIN($A${row})".format(row = i + 3)])

chart1 = ScatterChart()
chart1.title = "Default Orientation"
chart1.x_axis.title = 'x'
chart1.y_axis.title = 'y'
chart1.legend = None

chart2 = ScatterChart()
chart2.title = "Flip X"
chart2.x_axis.title = 'x'
chart2.y_axis.title = 'y'
chart2.legend = None
chart2.x_axis.scaling.orientation = "maxMin"
chart2.y_axis.scaling.orientation = "minMax"

chart3 = ScatterChart()
chart3.title = "Flip Y"
chart3.x_axis.title = 'x'
chart3.y_axis.title = 'y'
chart3.legend = None
chart3.x_axis.scaling.orientation = "minMax"
chart3.y_axis.scaling.orientation = "maxMin"

chart4 = ScatterChart()
chart4.title = "Flip Both"
chart4.x_axis.title = 'x'
chart4.y_axis.title = 'y'
chart4.legend = None
chart4.x_axis.scaling.orientation = "maxMin"
chart4.y_axis.scaling.orientation = "maxMin"

x = Reference(ws, min_col=2, min_row=2, max_row=102)
y = Reference(ws, min_col=3, min_row=2, max_row=102)
s = Series(y, xvalues=x)
chart1.append(s)
```

(下页继续)

(续上页)

```
chart2.append(s)
chart3.append(s)
chart4.append(s)

ws.add_chart(chart1, "D1")
ws.add_chart(chart2, "J1")
ws.add_chart(chart3, "D15")
ws.add_chart(chart4, "J15")

wb.save("orientation.xlsx")
```

This produces four charts with the axes in each possible combination of orientations that look something like this:



Adding a second axis

Adding a second axis actually involves creating a second chart that shares a common x-axis with the first chart but has a separate y-axis.

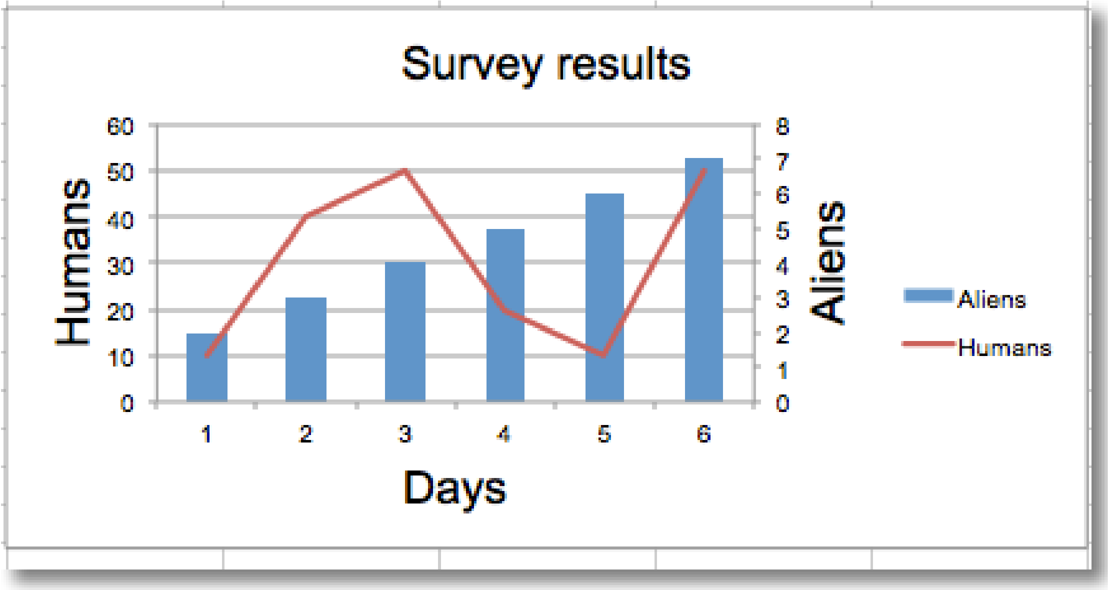
```
from openpyxl import Workbook
from openpyxl.chart import (
    LineChart,
    BarChart,
    Reference,
    Series,
)
```

(下页继续)

(续上页)

```
wb = Workbook()
ws = wb.active

rows = [
    ['Aliens', 2, 3, 4, 5, 6, 7],
    ['Humans', 10, 40, 50, 20, 10, 50],
]

for row in rows:
    ws.append(row)

c1 = BarChart()
v1 = Reference(ws, min_col=1, min_row=1, max_col=7)
c1.add_data(v1, titles_from_data=True, from_rows=True)

c1.x_axis.title = 'Days'
c1.y_axis.title = 'Aliens'
c1.y_axis.majorGridlines = None
c1.title = 'Survey results'


# Create a second chart
c2 = LineChart()
v2 = Reference(ws, min_col=1, min_row=2, max_col=7)
c2.add_data(v2, titles_from_data=True, from_rows=True)
c2.y_axis.axId = 200
c2.y_axis.title = "Humans"

# Display y-axis of the second chart on the right by setting it to cross the x-
↪axis at its maximum
c1.y_axis.crosses = "max"
c1 += c2

ws.add_chart(c1, "D4")

wb.save("secondary.xlsx")
```

This produces a combined line and bar chart looking something like this:



6.4.4 更改图表布局

更改绘图区和图例的布局

可以通过使用 layout 类实例的 layout 属性来设置图表的布局。

表格布局

位置和大小

图表可以放置在容器中。可以通过 `x` 和 `y` 调整位置。`w` 和 `h` 调整大小。单位是容器的比例。图表不能放置在容器的外部，并且宽度和高度是主要限制：如果 x+w>1，则 x=1-w。

x 是从左侧开始的水平位置

y 是从顶部开始的垂直位置

h 是图表相对于其容器的高度

w 是盒子（box）的宽度

模式

除了大小和位置外，相关属性的模式也可以设置为 *factor* 或 *edge*。默认值为 *factor*:

```
layout.xMode = edge
```

目标（Target）

~layoutTarget‘ 属性可以设置成 outer 或者 inner. 默认值为 outer:

```
layout.layoutTarget = inner
```

图例布局

图例的位置可以通过设置位置参数来进行改变：r、l、t`、，``b 和 tr，分别代表右、左、上、下以及右上。默认值为 r.

```
legend.position = 'tr'
```

或者应用手动布局：

```
legend.layout = ManualLayout()
```

```
from openpyxl import Workbook, load_workbook
from openpyxl.chart import ScatterChart, Series, Reference
from openpyxl.chart.layout import Layout, ManualLayout

wb = Workbook()
ws = wb.active

rows = [
    ['Size', 'Batch 1', 'Batch 2'],
    [2, 40, 30],
    [3, 40, 25],
    [4, 50, 30],
    [5, 30, 25],
    [6, 25, 35],
    [7, 20, 40],
]

for row in rows:
    ws.append(row)

ch1 = ScatterChart()
xvalues = Reference(ws, min_col=1, min_row=2, max_row=7)
for i in range(2, 4):
    values = Reference(ws, min_col=i, min_row=1, max_row=7)
```

(下页继续)

(续上页)

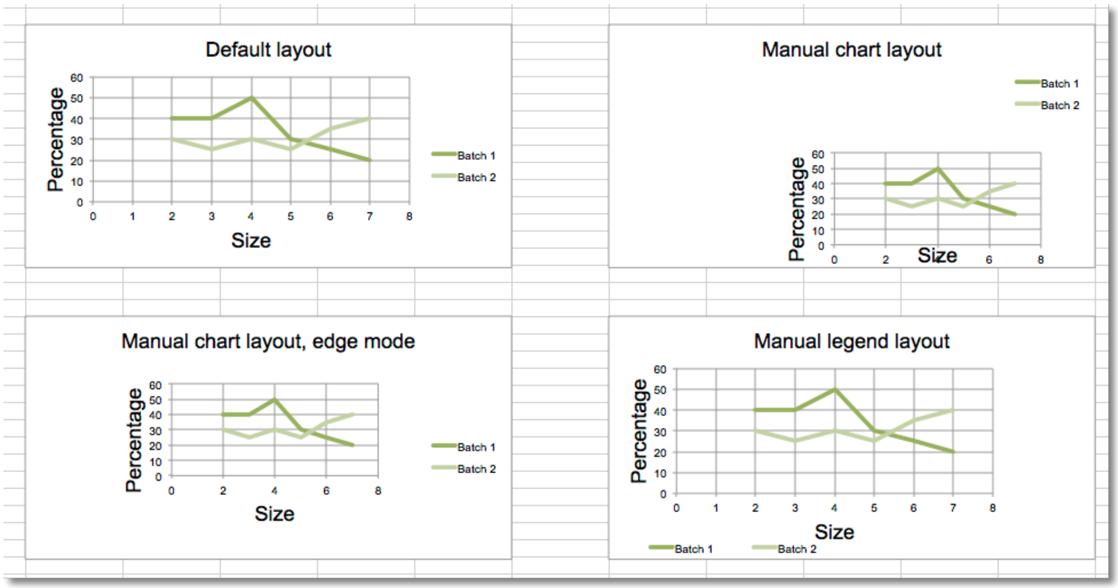
```
    series = Series(values, xvalues, title_from_data=True)
    ch1.series.append(series)


ch1.title = "Default layout"
ch1.style = 13
ch1.x_axis.title = 'Size'
ch1.y_axis.title = 'Percentage'
ch1.legend.position = 'r'

ws.add_chart(ch1, "B10")

from copy import deepcopy

# Half-size chart, bottom right
ch2 = deepcopy(ch1)
ch2.title = "Manual chart layout"
ch2.legend.position = "tr"
ch2.layout=Layout(
    manualLayout=ManualLayout(
        x=0.25, y=0.25,
        h=0.5, w=0.5,
    )
)
ws.add_chart(ch2, "H10")

# Half-size chart, centred
ch3 = deepcopy(ch1)
ch3.layout = Layout(
    ManualLayout(
    x=0.25, y=0.25,
    h=0.5, w=0.5,
    xMode="edge",
    yMode="edge",
    )
)
ch3.title = "Manual chart layout, edge mode"
ws.add_chart(ch3, "B27")

# Manually position the legend bottom left
```

(下页继续)

(续上页)

```
ch4 = deepcopy(ch1)
ch4.title = "Manual legend layout"
ch4.legend.layout = Layout(
    manualLayout=ManualLayout(
        yMode='edge',
        xMode='edge',
        x=0, y=0.9,
        h=0.1, w=0.5
    )
)

ws.add_chart(ch4, "H27")

wb.save("chart_layout.xlsx")
```

以上会创建四个图表并展示了各种可能性：



6.4.5 图表样式

Adding Patterns

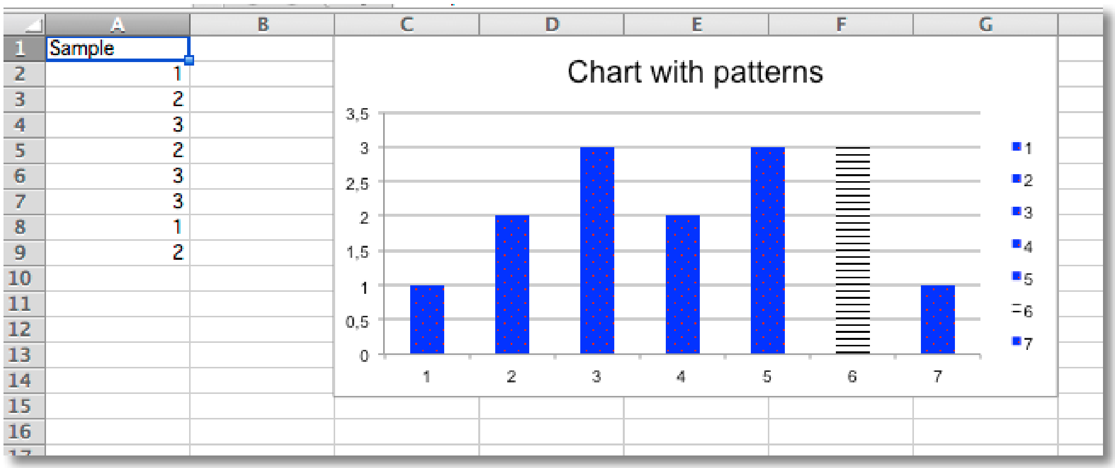
Whole data series and individual data points can be extensively styled through the *graphicalProperties*. Getting things just right may take some time.

```
from openpyxl import Workbook
from openpyxl.chart import BarChart, Reference
from openpyxl.chart.marker import DataPoint

from openpyxl.drawing.fill import PatternFillProperties, ColorChoice

wb = Workbook()
ws = wb.active

rows = [
    ("Sample",),
    (1,),
    (2,),
    (3,),
    (2,),
    (3,),
    (3,),
    (1,),
    (2,),
]

for r in rows:
    ws.append(r)


c = BarChart()
data = Reference(ws, min_col=1, min_row=1, max_row=8)
c.add_data(data, titles_from_data=True)
c.title = "Chart with patterns"

# set a pattern for the whole series
series = c.series[0]
fill =  PatternFillProperties(prst="pct5")
fill.foreground = ColorChoice(prstClr="red")
fill.background = ColorChoice(prstClr="blue")
series.graphicalProperties.pattFill = fill

# set a pattern for a 6th data point (0-indexed)
pt = DataPoint(idx=5)
pt.graphicalProperties.pattFill = PatternFillProperties(prst="ltHorz")
```

(下页继续)

(续上页)

```
series.dPt.append(pt)

ws.add_chart(c, "C1")

wb.save("pattern.xlsx")
```



	A
1	Sample
2	1
3	2
4	3
5	2
6	3
7	3
8	1
9	2
10	
11	
12	
13	
14	
15	
16	

6.4.6 高级图表

图表能合并生成新的图表:

Gauge Charts

Gauge charts combine a pie chart and a doughnut chart to create a “gauge” . The first chart is a doughnut chart with four slices. The first three slices correspond to the colours of the gauge; the fourth slice, which is half of the doughnut, is made invisible.

A pie chart containing three slices is added. The first and third slice are invisible so that the second slice can act as the needle on the gauge.

The effects are done using the graphical properties of individual data points in a data series.

```
from openpyxl import Workbook

from openpyxl.chart import PieChart, DoughnutChart, Series, Reference
from openpyxl.chart.series import DataPoint


data = [
    ["Donut", "Pie"],
```

(下页继续)

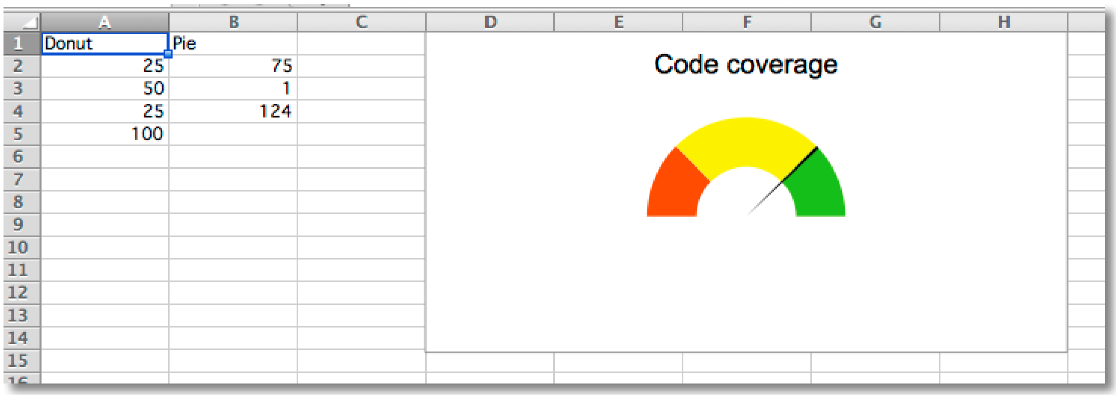
(续上页)

```
    [25, 75],
    [50, 1],
    [25, 124],
    [100],
]

# based on http://www.excel-easy.com/examples/gauge-chart.html

wb = Workbook()
ws = wb.active
for row in data:
    ws.append(row)

# First chart is a doughnut chart
c1 = DoughnutChart(firstSliceAng=270, holeSize=50)
c1.title = "Code coverage"
c1.legend = None

ref = Reference(ws, min_col=1, min_row=2, max_row=5)
s1 = Series(ref, title_from_data=False)

slices = [DataPoint(idx=i) for i in range(4)]
slices[0].graphicalProperties.solidFill = "FF3300" # red
slices[1].graphicalProperties.solidFill = "FCF305" # yellow
slices[2].graphicalProperties.solidFill = "1FB714" # green
slices[3].graphicalProperties.noFill = True # invisible

s1.data_points = slices
c1.series = [s1]

# Second chart is a pie chart
c2 = PieChart(firstSliceAng=270)
c2.legend = None

ref = Reference(ws, min_col=2, min_row=2, max_col=2, max_row=4)
s2 = Series(ref, title_from_data=False)

slices = [DataPoint(idx=i) for i in range(3)]
slices[0].graphicalProperties.noFill = True # invisible
slices[1].graphicalProperties.solidFill = "000000" # black needle
```

(下页继续)

(续上页)

```
slices[2].graphicalProperties.noFill = True # invisible
s2.data_points = slices
c2.series = [s2]

c1 += c2 # combine charts

ws.add_chart(c1, "D1")

wb.save("gauge.xlsx")
```



	A	B	C
1	Donut	Pie	
2	25	75	
3	50	1	
4	25	124	
5	100		

6.4.7 使用 chartsheets

图表能被加入到一个称为 chartsheets 特殊工作簿中:

Chartsheets

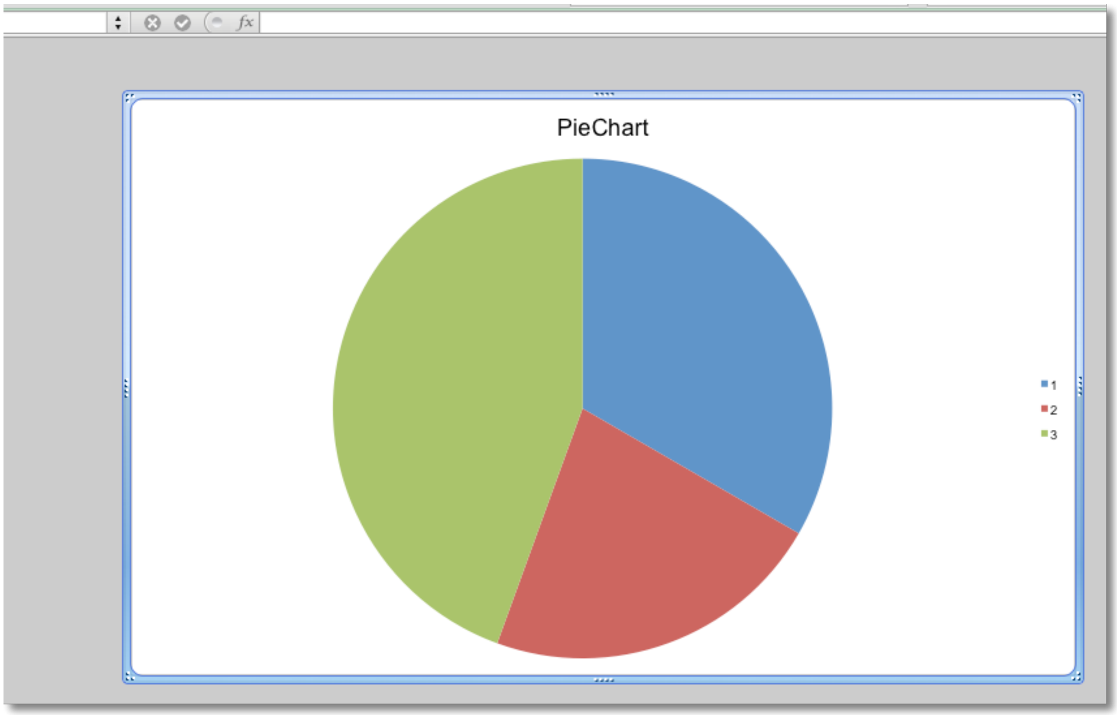
Chartsheets are special worksheets which only contain charts. All the data for the chart must be on a different worksheet.

```
from openpyxl import Workbook

from openpyxl.chart import PieChart, Reference, Series

wb = Workbook()
ws = wb.active
cs = wb.create_chartsheet()

rows = [
    ["Bob", 3],
```

(下页继续)

(续上页)

```
    ["Harry", 2],
    ["James", 4],
]

for row in rows:
    ws.append(row)


chart = PieChart()
labels = Reference(ws, min_col=1, min_row=1, max_row=3)
data = Reference(ws, min_col=2, min_row=1, max_row=3)
chart.series = (Series(data),)
chart.title = "PieChart"

cs.add_chart(chart)

wb.save("demo.xlsx")
```



6.5 注释

警告： openpyxl 目前只支持读写文字注释。格式信息会丢失。在读取时，注释尺寸也会丢失，但是可以重新写入。注释目前不支持 *read_only=True* 模式下使用。

6.5.1 为单元格添加注释

注释的 text 和 author 是必填属性

```
>>> from openpyxl import Workbook
>>> from openpyxl.comments import Comment
>>> wb = Workbook()
>>> ws = wb.active
>>> comment = ws["A1"].comment
>>> comment = Comment('This is the comment text', 'Comment Author')
>>> comment.text
'This is the comment text'
>>> comment.author
'Comment Author'
```

如果你为不同的单元格设置了相同的注释，那么 openpyxl 会自动进行复制

```
>>> from openpyxl import Workbook
>>> from openpyxl.comments import Comment
>>> wb=Workbook()
>>> ws=wb.active
>>> comment = Comment("Text", "Author")
>>> ws["A1"].comment = comment
>>> ws["B2"].comment = comment
>>> ws["A1"].comment is comment
True
>>> ws["B2"].comment is comment
False
```

6.5.2 加载和保存注释

加载时工作簿中存在的注释会自动存储在其相应单元格的 comment 属性中。格式信息（如字体大小，粗体和斜体）以及注释的容器框的原始尺寸和位置都将丢失。

保存工作簿时保留在工作簿中的注释会自动保存到工作簿文件中。

注释尺寸可以设定成只写。注释尺寸以像素为单位。

```
>>> from openpyxl import Workbook
>>> from openpyxl.comments import Comment
>>> from openpyxl.utils import units
>>>
>>> wb=Workbook()
>>> ws=wb.active
>>>
>>> comment = Comment("Text", "Author")
>>> comment.width = 300
>>> comment.height = 50
>>>
>>> ws["A1"].comment = comment
>>>
>>> wb.save('commented_book.xlsx')
```

如果有需要的话, openpyxl.utils.units 有将其他度量单位（mm，points）转换为像素的辅助函数:

```
>>> from openpyxl import Workbook
>>> from openpyxl.comments import Comment
>>> from openpyxl.utils import units
>>>
>>> wb=Workbook()
>>> ws=wb.active
>>>
>>> comment = Comment("Text", "Author")
>>> comment.width = units.points_to_pixels(300)
>>> comment.height = units.points_to_pixels(50)
>>>
>>> ws["A1"].comment = comment
```

6.6 样式

6.6.1 介绍

样式用于在屏幕上显示时更改数据的外观。它们还用于确定数字的格式。

样式可以应用于以下方面:

- Font：设置字体大小、颜色、下划线等等

- PatternFill：设置图案或者颜色渐变
- Border：设置单元格的边框
- Alignment：单元格对齐
- Protection：保护工作表

以下是默认值

```
>>> from openpyxl.styles import PatternFill, Border, Side, Alignment,␣
↪Protection, Font
>>> font = Font(name='Calibri',
...                 size=11,
...                 bold=False,
...                 italic=False,
...                 vertAlign=None,
...                 underline='none',
...                 strike=False,
...                 color='FF000000')
>>> fill = PatternFill(fill_type=None,
...                 start_color='FFFFFFFF',
...                 end_color='FF000000')
>>> border = Border(left=Side(border_style=None,
...                           color='FF000000'),
...                 right=Side(border_style=None,
...                            color='FF000000'),
...                 top=Side(border_style=None,
...                          color='FF000000'),
...                 bottom=Side(border_style=None,
...                             color='FF000000'),
...                 diagonal=Side(border_style=None,
...                               color='FF000000'),
...                 diagonal_direction=0,
...                 outline=Side(border_style=None,
...                              color='FF000000'),
...                 vertical=Side(border_style=None,
...                               color='FF000000'),
...                 horizontal=Side(border_style=None,
...                                color='FF000000')
...                )
>>> alignment=Alignment(horizontal='general',
...                     vertical='bottom',
...                     text_rotation=0,
```

（下页继续）

(续上页)

```
...                     wrap_text=False,
...                     shrink_to_fit=False,
...                     indent=0)
>>> number_format = 'General'
>>> protection = Protection(locked=True,
...                         hidden=False)
>>>
```

6.6.2 单元格样式和命名样式

有两种不同的样式: 单元格样式和命名样式, 也被成为样式模板。

单元格样式

单元格样式在对象之间共享，一旦被分配之后就无法更改。这样可以避免不必要的副作用，例如仅更改一个单元格时就更改许多单元格的样式。

```
>>> from openpyxl.styles import colors
>>> from openpyxl.styles import Font, Color
>>> from openpyxl import Workbook
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> a1 = ws['A1']
>>> d4 = ws['D4']
>>> ft = Font(color="FF0000")
>>> a1.font = ft
>>> d4.font = ft
>>>
>>> a1.font.italic = True # is not allowed # doctest: +SKIP
>>>
>>> # If you want to change the color of a Font, you need to reassign it
>>>
>>> a1.font = Font(color="FF0000", italic=True) # the change only affects A1
```

6.6.3 复制样式

样式也可以被复制

```
>>> from openpyxl.styles import Font
>>> from copy import copy
>>>
>>> ft1 = Font(name='Arial', size=14)
>>> ft2 = copy(ft1)
>>> ft2.name = "Tahoma"
>>> ft1.name
'Arial'
>>> ft2.name
'Tahoma'
>>> ft2.size # copied from the
14.0
```

6.6.4 颜色

可以通过三种方式：indexed, aRGB 或者 theme 来设置字体、背景、边框等的颜色。索引颜色（indexed colours）是旧版实现，颜色本身取决于工作簿或应用程序默认提供的索引。主题颜色可用于互补色，但也取决于工作簿中存在的主题。因此，建议使用 aRGB 颜色。

aRGB 颜色

使用红色，绿色和蓝色的十六进制值设置 RGB 颜色。

```
>>> from openpyxl.styles import Font
>>> font = Font(color="FF0000")
```

理论上，alpha 值是指颜色的透明度，但这与单元格样式无关。默认值 00 将前置任何简单的 RGB 值：

```
>>> from openpyxl.styles import Font
>>> font = Font(color="00FF00")
>>> font.color.rgb
'0000FF00'
```

还支持传统索引颜色以及主题和色彩（themes and tints）。

```
>>> from openpyxl.styles.colors import Color
>>> c = Color(indexed=32)
>>> c = Color(theme=6, tint=0.5)
```

Indexed Colours

索引 64 和 65 不能设置，并且分别保留给系统前景色和背景色。

6.6.5 应用样式

样式被直接应用到单元格

```
>>> from openpyxl.workbook import Workbook
>>> from openpyxl.styles import Font, Fill
>>> wb = Workbook()
>>> ws = wb.active
>>> c = ws['A1']
>>> c.font = Font(size=12)
```

样式也可以应用于列和行，但是请注意，这仅适用于关闭文件后创建的单元格（在 Excel）。如果要对整个行和列应用样式，则必须自己将样式应用于每个单元格。这是文件格式的限制:: Styles can also applied to columns and rows but note that this applies only to cells created (in Excel) after the file is closed. If you want to apply styles to entire rows and columns then you must apply the style to each cell yourself. This is a restriction of the file format:

```
>>> col = ws.column_dimensions['A']
>>> col.font = Font(bold=True)
>>> row = ws.row_dimensions[1]
>>> row.font = Font(underline="single")
```

6.6.6 合并单元格的样式

合并单元格和其他单元格对象的行为相似，通过左上单元格来定义值和样式。可以改变左上单元格的边框来改变整个合并单元格的边框。这种格式是出于编辑目的才被生成（The formatting is generated for the purpose of writing.）

```
>>> from openpyxl.styles import Border, Side, PatternFill, Font, GradientFill,␣
↪Alignment
>>> from openpyxl import Workbook
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>> ws.merge_cells('B2:F4')
>>>
>>> top_left_cell = ws['B2']
```

（下页继续）

(续上页)

```
>>> top_left_cell.value = "My Cell"
>>>
>>> thin = Side(border_style="thin", color="000000")
>>> double = Side(border_style="double", color="ff0000")
>>>
>>> top_left_cell.border = Border(top=double, left=thin, right=thin,␣
↪bottom=double)
>>> top_left_cell.fill = PatternFill("solid", fgColor="DDDDDD")
>>> top_left_cell.fill = fill = GradientFill(stop=("000000", "FFFFFF"))
>>> top_left_cell.font  = Font(b=True, color="FF0000")
>>> top_left_cell.alignment = Alignment(horizontal="center", vertical="center")
>>>
>>> wb.save("styled.xlsx")
```

6.6.7 编辑页面设置

```
>>> from openpyxl.workbook import Workbook
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> ws.page_setup.orientation = ws.ORIENTATION_LANDSCAPE
>>> ws.page_setup.paperSize = ws.PAPERSIZE_TABLOID
>>> ws.page_setup.fitToHeight = 0
>>> ws.page_setup.fitToWidth = 1
```

命名样式

与单元格样式相反，命名样式是可变的。当您想一次将格式应用于许多不同的单元格时，它们很有意义。注意一旦将命名样式分配给单元格后，对该样式的更改将 ** 不会 ** 影响单元格。

一旦命名样式被注册到工作簿，就可以简单的通过名字来进行引用

6.6.8 创建命名样式

```
>>> from openpyxl.styles import NamedStyle, Font, Border, Side
>>> highlight = NamedStyle(name="highlight")
>>> highlight.font = Font(bold=True, size=20)
```

(下页继续)

(续上页)

```
>>> bd = Side(style='thick', color="000000")
>>> highlight.border = Border(left=bd, top=bd, right=bd, bottom=bd)
```

创建命名样式后，即可将其注册到工作簿中：

```
>>> wb.add_named_style(highlight)
```

命名样式在首次分配给单元格时也会自动注册：

```
>>> ws['A1'].style = highlight
```

注册后，仅使用名称分配样式：

```
>>> ws['D5'].style = 'highlight'
```

6.6.9 使用内置样式（Ps：以下注释由译者根据 office365 中文版进行添加）

该规范（specification）包括一些可以使用的内置样式。不幸的是，这些样式的名称以其本地化形式存储。openpyxl 仅会识别英文名称，并且只能与此处的文字完全一样。

- ‘Normal’ # 无样式

数字格式

- ‘Comma’ # 千位分隔，保留两位小数 ‘Warning Text’
- ‘Comma [0]’ # 千位分隔，不保留小数
- ‘Currency’ # 货币，保留两位小数
- ‘Currency [0]’ # 货币，不保留小数
- ‘Percent’ # 百分比

Informative

- ‘Calculation’ # 计算
- ‘Total’ # 汇总
- ‘Note’ # 注释
- ‘Warning Text’ # 警告文本
- ‘Explanatory Text’ # 解释性文本

文字样式

- ‘Title’ # 标题
- ‘Headline 1’ # 标题 1
- ‘Headline 2’ # 标题 2
- ‘Headline 3’ # 标题 3
- ‘Headline 4’ # 标题 4
- ‘Hyperlink’ # 超链接
- ‘Followed Hyperlink’ # 已访问的超链接
- ‘Linked Cell’ # 链接单元格

Comparisons

- ‘Input’ # 输入
- ‘Output’ # 输出
- ‘Check Cell’ # 检查单元格
- ‘Good’ # 好
- ‘Bad’ # 坏
- ‘Neutral’ # 始终

高亮

- ‘Accent1’ # 着色 1
- ‘20 % - Accent1’
- ‘40 % - Accent1’
- ‘60 % - Accent1’
- ‘Accent2’ # 着色 2
- ‘20 % - Accent2’
- ‘40 % - Accent2’
- ‘60 % - Accent2’
- ‘Accent3’ # 着色 3
- ‘20 % - Accent3’
- ‘40 % - Accent3’

- ‘60 % - Accent3’
- ‘Accent4’ # 着色 4
- ‘20 % - Accent4’
- ‘40 % - Accent4’
- ‘60 % - Accent4’
- ‘Accent5’ # 着色 5
- ‘20 % - Accent5’
- ‘40 % - Accent5’
- ‘60 % - Accent5’
- ‘Accent6’ # 着色 6
- ‘20 % - Accent6’
- ‘40 % - Accent6’
- ‘60 % - Accent6’
- ‘Pandas’ # 好像是自定义的

有关内置样式的更多信息，请参阅*openpyxl.styles.builtins*

6.7 其他工作表属性

有一些特定行为的高级属性，最常用的是页面设置参数（page setup property）*fitTopage* 和定义工作表选项卡颜色的 ‘tabColor‘。

6.7.1 工作表可用属性

- “enableFormatConditionsCalculation”
- “filterMode”
- “published”
- “syncHorizontal”
- “syncRef”
- “syncVertical”
- “transitionEvaluation”
- “transitionEntry”
- “tabColor”

6.7.2 页面设置属性的可用字段

“autoPageBreaks” “fitToPage”

6.7.3 outlines 的可用字段

- “applyStyles”
- “summaryBelow”
- “summaryRight”
- “showOutlineSymbols”

更多信息请查询 http://msdn.microsoft.com/en-us/library/documentformat.openxml.spreadsheet.sheetproperties%28v=office.14%29.aspx_

注解: 默认情况下，会对 *outline* 属性进行初始化，因此您可以直接修改它们的 4 个属性，而页面设置属性不一样。如果要修改后者，首先要必要的参数初始化对 *openpyxl.worksheet.properties.PageSetupProperties* 对象进行初始化。一旦完成，可以在以后需要时通过例程直接对其进行修改。

```
>>> from openpyxl.workbook import Workbook
>>> from openpyxl.worksheet.properties import WorksheetProperties,␣
↪PageSetupProperties
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> wsprops = ws.sheet_properties
>>> wsprops.tabColor = "1072BA"
>>> wsprops.filterMode = False
>>> wsprops.pageSetUpPr = PageSetupProperties(fitToPage=True,␣
↪autoPageBreaks=False)
>>> wsprops.outlinePr.summaryBelow = False
>>> wsprops.outlinePr.applyStyles = True
>>> wsprops.pageSetUpPr.autoPageBreaks = True
```

6.8 条件格式

Excel 支持三种类型的条件格式：内置、标准和自定义内建条件格式将特定规则与预定义样式结合在一起。标准条件格式将特定规则与自定义格式结合在一起。In additional it is possible to define custom formulae for applying custom formats using differential styles.

注解： 不同规则的语法差异很大，以至于 openpyxl 不知道规则是否有意义。

创建条件格式规则的基本语法为：

```
>>> from openpyxl.formatting import Rule
>>> from openpyxl.styles import Font, PatternFill, Border
>>> from openpyxl.styles.differential import DifferentialStyle
>>> dxf = DifferentialStyle(font=Font(bold=True), fill=PatternFill(start_color=
↪'EE1111', end_color='EE1111'))
>>> rule = Rule(type='cellIs', dxf=dxf, formula=["10"])
```

由于某些规则的签名可能非常冗长，因此也有一些方便的工厂（factories）来创建它们。

6.8.1 内置格式

内置格式有：

- 色阶（ColorScale）
- 图表集（IconSet）
- 数据条（DataBar）

Builtin formats contain a sequence of formatting settings which combine a type with an integer for comparison. 可能的类型有：’ *num*’ ,‘*percent*’ ,‘*max*’ ,‘*min*’ ,‘*formula*’ ,‘*percentile*’。

色阶

你可以使用 2 或 3 种颜色的色阶。2 种色阶产生一种颜色到另一种颜色的渐变；3 种颜色色阶会将 1 种颜色用于 2 个颜色的渐变。

创建色阶的完整规则为:

```
>>> from openpyxl.formatting.rule import ColorScale, FormatObject
>>> from openpyxl.styles import Color
>>> first = FormatObject(type='min')
>>> last = FormatObject(type='max')
```

(下页继续)

(续上页)

```
>>> # colors match the format objects:
>>> colors = [Color('AA0000'), Color('00AA00')]
>>> cs2 = ColorScale(cfvo=[first, last], color=colors)
>>> # a three color scale would extend the sequences
>>> mid = FormatObject(type='num', val=40)
>>> colors.insert(1, Color('00AA00'))
>>> cs3 = ColorScale(cfvo=[first, mid, last], color=colors)
>>> # create a rule with the color scale
>>> from openpyxl.formatting.rule import Rule
>>> rule = Rule(type='colorScale', colorScale=cs3)
```

有一个方便创建色阶规则的函数:

```
>>> from openpyxl.formatting.rule import ColorScaleRule
>>> rule = ColorScaleRule(start_type='percentile', start_value=10, start_color=
↪'FFAA0000',
...                       mid_type='percentile', mid_value=50, mid_color=
↪'FF0000AA',
...                       end_type='percentile', end_value=90, end_color=
↪'FF00AA00')
```

图标集

从以下图标中进行选择: ‘*3Arrows*’ , ‘*3ArrowsGray*’ , ‘*3Flags*’ , ‘*3TrafficLights1*’ , ‘*3TrafficLights2*’ , ‘*3Signs*’ , ‘*3Symbols*’ , ‘*3Symbols2*’ , ‘*4Arrows*’ , ‘*4ArrowsGray*’ , ‘*4RedToBlack*’ , ‘*4Rating*’ , ‘*4TrafficLights*’ , ‘*5Arrows*’ , ‘*5ArrowsGray*’ , ‘*5Rating*’ , ‘*5Quarters*’

创建图表集完整规则为:

```
>>> from openpyxl.formatting.rule import IconSet, FormatObject
>>> first = FormatObject(type='percent', val=0)
>>> second = FormatObject(type='percent', val=33)
>>> third = FormatObject(type='percent', val=67)
>>> iconset = IconSet(iconSet='3TrafficLights1', cfvo=[first, second, third],␣
↪showValue=None, percent=None, reverse=None)
>>> # assign the icon set to a rule
>>> from openpyxl.formatting.rule import Rule
>>> rule = Rule(type='iconSet', iconSet=iconset)
```

有一个方便创建色阶图表集规则的函数:

```
>>> from openpyxl.formatting.rule import IconSetRule
>>> rule = IconSetRule('5Arrows', 'percent', [10, 20, 30, 40, 50],␣
↪showValue=None, percent=None, reverse=None)
```

数据条

目前，openpyxl 支持原始规范中定义的数据条。之后的扩展中添加了边框和方向。

完整创建数据条的规则为：

```
>>> from openpyxl.formatting.rule import DataBar, FormatObject
>>> first = FormatObject(type='min')
>>> second = FormatObject(type='max')
>>> data_bar = DataBar(cfvo=[first, second], color="638EC6", showValue=None,␣
↪minLength=None, maxLength=None)
>>> # assign the data bar to a rule
>>> from openpyxl.formatting.rule import Rule
>>> rule = Rule(type='dataBar', dataBar=data_bar)
```

有一个方便创建数据条规则的函数：

```
>>> from openpyxl.formatting.rule import DataBarRule
>>> rule = DataBarRule(start_type='percentile', start_value=10, end_type=
↪'percentile', end_value='90',
...                     color="FF638EC6", showValue="None", minLength=None,␣
↪maxLength=None)
```

6.8.2 标准条件格式

标准条件格式为：

- 平均值（Average）
- 百分比（Percent）
- 唯一值或重复值（Unique or duplicate）
- 值（Value）
- 排名（Rank）

```
>>> from openpyxl import Workbook
>>> from openpyxl.styles import Color, PatternFill, Font, Border
>>> from openpyxl.styles.differential import DifferentialStyle
```

（下页继续）

(续上页)

```
>>> from openpyxl.formatting.rule import ColorScaleRule, CellIsRule,␣
↪FormulaRule
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> # Create fill
>>> redFill = PatternFill(start_color='EE1111',
...                 end_color='EE1111',
...                 fill_type='solid')
>>>
>>> # Add a two-color scale
>>> # Takes colors in excel 'RRGGBB' style.
>>> ws.conditional_formatting.add('A1:A10',
...              ColorScaleRule(start_type='min', start_color='AA0000',
...                            end_type='max', end_color='00AA00')
...                            )
>>>
>>> # Add a three-color scale
>>> ws.conditional_formatting.add('B1:B10',
...                 ColorScaleRule(start_type='percentile', start_value=10,␣
↪start_color='AA0000',
...                             mid_type='percentile', mid_value=50, mid_color=
↪'0000AA',
...                             end_type='percentile', end_value=90, end_color=
↪'00AA00')
...                               )
>>>
>>> # Add a conditional formatting based on a cell comparison
>>> # addCellIs(range_string, operator, formula, stopIfTrue, wb, font, border,␣
↪fill)
>>> # Format if cell is less than 'formula'
>>> ws.conditional_formatting.add('C2:C10',
...              CellIsRule(operator='lessThan', formula=['C$1'],␣
↪stopIfTrue=True, fill=redFill))
>>>
>>> # Format if cell is between 'formula'
>>> ws.conditional_formatting.add('D2:D10',
...              CellIsRule(operator='between', formula=['1','5'],␣
↪stopIfTrue=True, fill=redFill))
```

(下页继续)

(续上页)

```
>>>
>>> # Format using a formula
>>> ws.conditional_formatting.add('E1:E10',
...             FormulaRule(formula=['ISBLANK(E1)'], stopIfTrue=True,␣
↪fill=redFill))
>>>
>>> # Aside from the 2-color and 3-color scales, format rules take fonts,␣
↪borders and fills for styling:
>>> myFont = Font()
>>> myBorder = Border()
>>> ws.conditional_formatting.add('E1:E10',
...             FormulaRule(formula=['E1=0'], font=myFont, border=myBorder,␣
↪fill=redFill))
>>>
>>> # Highlight cells that contain particular text by using a special formula
>>> red_text = Font(color="9C0006")
>>> red_fill = PatternFill(bgColor="FFC7CE")
>>> dxf = DifferentialStyle(font=red_text, fill=red_fill)
>>> rule = Rule(type="containsText", operator="containsText", text="highlight",
↪ dxf=dxf)
>>> rule.formula = ['NOT(ISERROR(SEARCH("highlight",A1)))']
>>> ws.conditional_formatting.add('A1:F40', rule)
>>> wb.save("test.xlsx")
```

6.8.3 条件格式应用在全部行

有时你想将条件格式应用于多个单元格，例如一行包含特定值的一些单元格。

```
>>> ws.append(['Software', 'Developer', 'Version'])
>>> ws.append(['Excel', 'Microsoft', '2016'])
>>> ws.append(['openpyxl', 'Open source', '2.6'])
>>> ws.append(['OpenOffice', 'Apache', '4.1.4'])
>>> ws.append(['Word', 'Microsoft', '2010'])
```

我们要突出开发人员是 Microsoft 的行。我们通过创建表达式规则并使用公式来识别哪些行包含了 Microsoft 开发的 Software。

```
>>> red_fill = PatternFill(bgColor="FFC7CE")
>>> dxf = DifferentialStyle(fill=red_fill)
>>> r = Rule(type="expression", dxf=dxf, stopIfTrue=True)
```

(下页继续)

(续上页)

```
>>> r.formula = ['$A2="Microsoft"']
>>> ws.conditional_formatting.add("A1:C10", r)
```

注解： 在这种情况下，该公式使用 ** 绝对引用 ** B 列，以及 ** 相对引用 ** 行号，在这种情况下，1 是行号相对于应用格式的范围。做到这一点可能很棘手，但是即使已将规则添加到工作表的条件格式集合中，也可以对其进行调整。

6.9 数据透视表

openpyxl 为数据透视表提供读取支持以便于可以保留在现有的文件中。数据透视表的规范虽然很广泛，但不是很清楚，也不意味着客户端代码应该能够创建数据透视表。(The specification for pivot tables, while extensive, is not very clear and it is not intended that client code should be able to create pivot tables.) 但是，应该可以编辑和操作现有的数据透视表，例如。更改其范围或是能自动更新设置。

和图表、图片、表格一样，数据透视表没有专门管理的 API，因此客户端代码不得不遍历工作表 `_pivots` 列表

6.9.1 例子

```
from openpyxl import load_workbook
wb = load_workbook("campaign.xlsx")
ws = wb["Results"]
pivot = ws._pivots[0] # any will do as they share the same cache
pivot.cache.refreshOnLoad = True
```

更多信息请查询*openpyxl.pivot.cache.CacheDefinition*

6.10 打印设置

openpyxl 为打印设置提供合理的全面支持

6.10.1 编辑打印设置

```
>>> from openpyxl.workbook import Workbook
>>>
```

(下页继续)

(续上页)

```
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> ws.print_options.horizontalCentered = True
>>> ws.print_options.verticalCentered = True
```

6.10.2 页眉页脚

页眉和页脚使用自己的格式语言。在编辑的时候完全可以支持但是由于于复杂和嵌套的可能性，在读取它们时仅部分支持。支持字体，大小和颜色，居左，居中或居右元素。粒度控制（突出显示单个单词）需要手动应用控制代码（Granular control (highlighting individuals words) will require applying control codes manually）

```
>>> from openpyxl.workbook import Workbook
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> ws.oddHeader.left.text = "Page &[Page] of &N"
>>> ws.oddHeader.left.size = 14
>>> ws.oddHeader.left.font = "Tahoma,Bold"
>>> ws.oddHeader.left.color = "CC3366"
```

也支持 *evenHeader* 和 *evenFooter* 以及 *firstHeader* 和 *firstFooter*.

6.10.3 增加打印标题

您可以在每页上打印标题，以确保正确标记数据。

```
>>> from openpyxl.workbook import Workbook
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> ws.print_title_cols = 'A:B' # the first two cols
>>> ws.print_title_rows = '1:1' # the first row
```

6.10.4 增加打印区域

你可以只选择工作簿的一部分来作为打印区域

```
>>> from openpyxl.workbook import Workbook
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> ws.print_area = 'A1:F10'
```

6.11 筛选和排序

在工作簿中添加筛选是可能的

注解: 筛选和排序只能通过 openpyxl 进行设置，但是只有在 Excel 这样的程序中才会被应用。这是由于他们会在范围内重新排列或者格式化单元格或行。(This is because they actually rearranges or format cells or rows in the range)

定义一个范围后，你可以对一列添加筛选或者添加排序条件: (To add a filter you define a range and then add columns and sort conditions:)

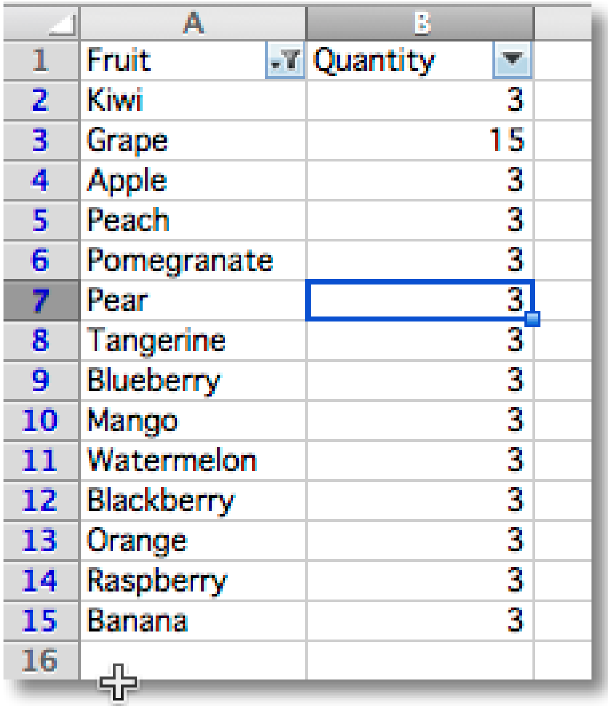
```
from openpyxl import Workbook

wb = Workbook()
ws = wb.active

data = [
    ["Fruit", "Quantity"],
    ["Kiwi", 3],
    ["Grape", 15],
    ["Apple", 3],
    ["Peach", 3],
    ["Pomegranate", 3],
    ["Pear", 3],
    ["Tangerine", 3],
    ["Blueberry", 3],
    ["Mango", 3],
    ["Watermelon", 3],
    ["Blackberry", 3],
    ["Orange", 3],
    ["Raspberry", 3],
```

(下页继续)

（续上页）

```
    ["Banana", 3]
]

for r in data:
    ws.append(r)

ws.auto_filter.ref = "A1:B15"
ws.auto_filter.add_filter_column(0, ["Kiwi", "Apple", "Mango"])
ws.auto_filter.add_sort_condition("B2:B15")

wb.save("filtered.xlsx")
```

这会将相关指令添加到文件中，但实际上不会 **过滤或排序**。（PS：译者使用上诉代码在 Excel 中试了一下这个功能，其中已经出现了筛选栏控件，但是未生效，点击“确认”即可生效，排序功能点了“确认也没办法使用”）



	A	B
1	Fruit	Quantity
2	Kiwi	3
3	Grape	15
4	Apple	3
5	Peach	3
6	Pomegranate	3
7	Pear	3
8	Tangerine	3
9	Blueberry	3
10	Mango	3
11	Watermelon	3
12	Blackberry	3
13	Orange	3
14	Raspberry	3
15	Banana	3
16		

6.12 数据验证

数据验证器可以应用于范围单元格，但也不是强制和 evaluated。范围不必是连续的：例如“A1 B2：B5”包含 A1 和单元格 B2 至 B5，但不包含 A2 或 B2。

6.12.1 例子

```
>>> from openpyxl import Workbook
>>> from openpyxl.worksheet.datavalidation import DataValidation
>>>
>>> # Create the workbook and worksheet we'll be working with
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> # Create a data-validation object with list validation
>>> dv = DataValidation(type="list", formula1='"Dog,Cat,Bat"', allow_
↪blank=True)
>>>
>>> # Optionally set a custom error message
>>> dv.error ='Your entry is not in the list'
>>> dv.errorTitle = 'Invalid Entry'
>>>
>>> # Optionally set a custom prompt message
>>> dv.prompt = 'Please select from the list'
>>> dv.promptTitle = 'List Selection'
>>>
>>> # Add the data-validation object to the worksheet
>>> ws.add_data_validation(dv)
```

```
>>> # Create some cells, and add them to the data-validation object
>>> c1 = ws["A1"]
>>> c1.value = "Dog"
>>> dv.add(c1)
>>> c2 = ws["A2"]
>>> c2.value = "An invalid value"
>>> dv.add(c2)
>>>
>>> # Or, apply the validation to a range of cells
>>> dv.add('B1:B1048576') # This is the same as for the whole of column B
>>>
>>> # Check with a cell is in the validator
>>> "B4" in dv
True
```

注解： 没有在任何单元格应用的验证将会在保存的时候被忽略。

6.12.2 其他验证的例子

任何证书:

```
dv = DataValidation(type="whole")
```

任何大于 100 的整数:

```
dv = DataValidation(type="whole",
                    operator="greaterThan",
                    formula1=100)
```

任何小数:

```
dv = DataValidation(type="decimal")
```

任何在 0 至 1 之间的小数:

```
dv = DataValidation(type="decimal",
                    operator="between",
                    formula1=0,
                    formula2=1)
```

任何日期:

```
dv = DataValidation(type="date")
```

时间:

```
dv = DataValidation(type="time")
```

15 长度以下的文本:

```
dv = DataValidation(type="textLength",
                    operator="lessThanOrEqual"),
                    formula1=15)
```

序列:

```
from openpyxl.utils import quote_sheetname
dv = DataValidation(type="list",
                    formula1="{0}!$B$1:$B$10".format(quote_
↪sheetname(sheetname))
                    )
```

自定义规则:

```
dv = DataValidation(type="custom",
                    formula1"=SOMEFORMULA")
```

注解: See http://www.contextures.com/xlDataVal07.html for custom rules

6.13 定义名称

该规范对定义的名称有以下说法:

> “定义名称是用于表示单元格，区域，公式或常量值的描述性文本。”

这意味着它们的定义是非常宽松的。它们可能包含一个常数，一个公式，一个单元格引用，一个区域或跨不同工作表的多个区域。它们在工作簿全局定义并可以通过 *defined_names* 属性进行访问。

6.13.1 区域的使用示例

访问名为“my_range”的区域:

```
my_range = wb.defined_names['my_range']
# if this contains a range of cells then the destinations attribute is not None
dests = my_range.destinations # returns a generator of (worksheet title, cell␣
↪range) tuples

cells = []
for title, coord in dests:
    ws = wb[title]
    cells.append(ws[coord])
```

6.13.2 创建新的命名区域

```
import openpyxl
wb = openpyxl.Workbook()
new_range = openpyxl.workbook.defined_name.DefinedName('newrange', attr_text=
↪'Sheet!$A$1:$A$5')
wb.defined_names.append(new_range)

# create a local named range (only valid for a specific sheet)
sheetid = wb.sheetnames.index('Sheet')
private_range = openpyxl.workbook.defined_name.DefinedName('privaterange',␣
↪attr_text='Sheet!$A$6', localSheetId=sheetid)
wb.defined_names.append(private_range)
# this local range can't be retrieved from the global defined names
assert('privaterange' not in wb.defined_names)

# the scope has to be supplied to retrieve local ranges:
print(wb.defined_names.localnames(sheetid))
print(wb.defined_names.get('privaterange', sheetid).attr_text)
```

```
['privaterange']
Sheet!$A$6
```

6.14 工作簿表格

工作簿表格是对单元格组的引用。这使得某些操作（例如，对表格中的单元格进行样式设置）更加容易。

6.14.1 创建表格

```
from openpyxl import Workbook
from openpyxl.worksheet.table import Table, TableStyleInfo

wb = Workbook()
ws = wb.active

data = [
    ['Apples', 10000, 5000, 8000, 6000],
```

（下页继续）

(续上页)

```
    ['Pears',   2000, 3000, 4000, 5000],
    ['Bananas', 6000, 6000, 6500, 6000],
    ['Oranges',  500,  300,  200,  700],
]

# add column headings. NB. these must be strings
ws.append(["Fruit", "2011", "2012", "2013", "2014"])
for row in data:
    ws.append(row)

tab = Table(displayName="Table1", ref="A1:E5")

# Add a default style with striped rows and banded columns
style = TableStyleInfo(name="TableStyleMedium9", showFirstColumn=False,
                       showLastColumn=False, showRowStripes=True,␣
↪showColumnStripes=True)
tab.tableStyleInfo = style

'''
Table must be added using ws.add_table() method to avoid duplicate names.
Using this method ensures table name is unque through out defined names and␣
↪all other table name.
'''
ws.add_table(tab)
wb.save("table.xlsx")
```

在一个工作簿中表格名称必须是唯一的。默认情况下，表是从第一行的标题开始创建的，并且所有列的筛选以及表标题和列标题必须始终包含字符串。

警告: 在只写模式下，您必须手动将列标题添加到表格中，并且值必须始终与相应单元格的值相同（有关如何执行此操作的示例，请参见下面的例子），否则 Excel 可能会认为该文件无效并删除表格。

通过 *TableStyleInfo* 来管理样式。这允许你对行和列设置条纹以及应用不同的颜色主题。

6.14.2 使用表格

`ws.tables` 是特定工作簿下所有表格的 dictionary-like 对象:

```
>>> ws.tables
{"Table1",  <openpyxl.worksheet.table.Table object>}
```

通过范围或者名称获取表格

```
>>> ws.tables["Table1"]
or
>>> ws.tables["A1:D10"]
```

遍历工作簿下所有的表格

```
>>> for table in ws.tables.values():
>>>     print(table)
```

获取表名以及工作簿内所有表格的范围

返回表格名和范围的列表

```
>>> ws.tables.items()
>>> [("Table1", "A1:D10")]
```

删除表格

```
>>> del ws.tables["Table1"]
```

工作簿中的表格数量

```
>>> len(ws.tables)
>>> 1
```

6.14.3 手动添加表格表头

在只写模式下你可以添加没有表头的表格:

```
>>> table.headerRowCount = False
```

或者手动初始化表头:

```
>>> headings = ["Fruit", "2011", "2012", "2013", "2014"] # all values must be␣
↪strings
>>> table._initialise_columns()
>>> for column, value in zip(table.tableColumns, headings):
        column.name = value
```

6.15 Parsing Formulas

openpyxl 支持对单元格中嵌入的公式进行有限的解析。*openpyxl.formula* 类包含 *Tokenizer* 类用于将公式分解为其组成的 tokens（*openpyxl.formula* package contains a *Tokenizer* class to break formulas into their consitutuent tokens.）用法如下:

```
>>> from openpyxl.formula import Tokenizer
>>> tok = Tokenizer("""=IF($A$1,"then True",MAX(DEFAULT_VAL,'Sheet 2'!B1))""")
>>> print("\n".join("%12s%11s%9s" % (t.value, t.type, t.subtype) for t in tok.
↪items))
         IF(       FUNC     OPEN
        $A$1    OPERAND    RANGE
           ,        SEP      ARG
 "then True"    OPERAND     TEXT
           ,        SEP      ARG
        MAX(       FUNC     OPEN
 DEFAULT_VAL    OPERAND    RANGE
           ,        SEP      ARG
'Sheet 2'!B1    OPERAND    RANGE
           )       FUNC    CLOSE
           )       FUNC    CLOSE
```

如上所示, tokens 有三个令人感兴趣的属性:

- `.value`: The substring of the formula that produced this token
- `.type`: The type of token this represents. Can be one of
  - `Token.LITERAL`: If the cell does not contain a formula, its value is represented by a single `LITERAL` token.
  - `Token.OPERAND`: A generic term for any value in the Excel formula. (See `.subtype` below for more details).
  - `Token.FUNC`: Function calls are broken up into tokens for the opener (e.g., `SUM()`, followed by the arguments, followed by the closer (i.e., `)`). The function name and opening

parenthesis together form one FUNC token, and the matching parenthesis forms another FUNC token.

  - Token.ARRAY: Array literals (enclosed between curly braces) get two ARRAY tokens each, one for the opening { and one for the closing }.
  - Token.PAREN: When used for grouping subexpressions (and not to denote function calls), parentheses are tokenized as PAREN tokens (one per character).
  - Token.SEP: These tokens are created from either commas (,) or semicolons (;). Commas create SEP tokens when they are used to separate function arguments (e.g., SUM(a, b)) or when they are used to separate array elements (e.g., {a,b}). (They have another use as an infix operator for joining ranges). Semicolons are always used to separate rows in an array literal, so always create SEP tokens.
  - Token.OP_PRE: Designates a prefix unary operator. Its value is always + or -
  - Token.OP_IN: Designates an infix binary operator. Possible values are >=, <=, <>, =, >, <, *, /, +, -, ^, or &.
  - Token.OP_POST: Designates a postfix unary operator. Its value is always %.
  - Token.WSPACE: Created for any whitespace encountered. Its value is always a single space, regardless of how much whitespace is found.

- .subtype: Some of the token types above use the subtype to provide additional information about the token. Possible subtypes are:
  - Token.TEXT, Token.NUMBER, Token.LOGICAL, Token.ERROR, Token.RANGE: these subtypes describe the various forms of OPERAND found in formulae. LOGICAL is either TRUE or FALSE, RANGE is either a named range or a direct reference to another range. TEXT, NUMBER, and ERROR all refer to literal values in the formula
  - Token.OPEN and Token.CLOSE: these two subtypes are used by PAREN, FUNC, and ARRAY, to describe whether the token is opening a new subexpression or closing it.
  - Token.ARG and Token.ROW: are used by the SEP tokens, to distinguish between the comma and semicolon. Commas produce tokens of subtype ARG whereas semicolons produce tokens of subtype ROW

6.15.1 Translating formulae from one location to another

It is possible to translate (in the mathematical sense) formulae from one location to another using the openpyxl.formulas.translate.Translator class. For example, there a range of cells B2:E7 with a sum of each row in column F:

```
>>> from openpyxl.formula.translate import Translator
>>> ws['F2'] = "=SUM(B2:E2)"
```

(下页继续)

(续上页)

```
>>> # move the formula one colum to the right
>>> ws['G2'] = Translator("=SUM(B2:E2)", origin="F2").translate_formula("G2")
>>> ws['G2'].value
'=SUM(C2:F2)'
```

注解： This is limited to the same general restrictions of formulae: *A1* cell-references only and no support for defined names.

6.16 保护

警告： 工作簿或工作表的密码保护仅提供了十分基础的安全。数据未进行加密，所以可以使用各种免费工具进行修改。实际上，规范指出：工作表或工作簿的保护不应该与文件安全性混淆。这是为了保护你的工作簿免受意外修改的影响，并不能保护免受恶意修改的影响。

Openpyxl 支持保护工作簿和工作表不被修改。除非指定明确算法，否则将使用 Open XML "Legacy Password Hash Algorithm" 来生成哈希密码值。

6.16.1 工作簿保护

为防止其他用户查看隐藏的工作表、添加、移动、删除或隐藏工作表以及重命名工作表，可以使用密码保护工作簿的结构。可以使用 `openpyxl.workbook.protection.WorkbookProtection.workbookPassword` 属性设置密码:

```
>>> wb.security.workbookPassword = '...'
>>> wb.security.lockStructure = True
```

同样，可以通过设置另一个密码来防止从共享工作簿中删除更改跟踪和更改历史记录。可以使用 `openpyxl.workbook.protection.WorkbookProtection.revisionsPassword` 属性设置密码:

```
   >>> wb.security.revisionsPassword = '...'

:class:`openpyxl.workbook.protection.WorkbookProtection` 对象上的其他属性可以精确
控制所设置的限制 (restrictions are in place)，但是只有设置密码后，这些属性才能生效。
```

如果需要设置原始密码值而非使用默认哈希算法，我们也提供特定的设置函数-例如:

```
hashed_password = ...
wb.security.set_workbook_password(hashed_password, already_hashed=True)
```

6.16.2 工作表保护

也可以通过在*openpyxl.worksheet.protection.SheetProtection* 对象上设置属性来锁定工作表。与工作簿保护不同，可以使用或不使用密码来启用工作表保护。使用*openpyxl.worksheet.protection.SheetProtection.sheet* 属性或调用 enable() 或 disable() 俩启用工作表保护:

```
>>> ws = wb.active
>>> ws.protection.sheet = True
>>> ws.protection.enable()
>>> ws.protection.disable()
```

如果未设置密码，那么用户不需要密码即可禁用工作表保护。否则，他们必要提供密码才能修改保护配置。使用 `openpxyl.worksheet.protection.SheetProtection.password()` 设置密码:

```
>>> ws = wb.active
>>> ws.protection.password = '...'
```

CHAPTER 7

开发者信息

7.1 Development

If you find the openpyxl project intriguing and want to contribute a new awesome feature, fix a nasty bug or improve the documentation this section will guide you in setting up your development environment.

We will look into the coding standards and version control system workflows used, as well as cloning the openpyxl code to your local machine, setting up a virtual Python environment, running tests and building the documentation.

7.1.1 Getting the source

The source code of openpyxl is hosted on BitBucket as a Mercurial project which you can download using e.g. the GUI client SourceTree by Atlassian. If you prefer working with the command line you can use the following:

```
$ hg clone https://bitbucket.org/openpyxl/openpyxl
$ hg up 3.0
```

Please note that the default branch should never be used for development work. For bug fixes and minor patches you should base your work on the branch of the current release, e.g 3.0. New features should generally be based on the development branch of the **next** minor version. If in doubt get in touch with the openpyxl development team.

It is worthwhile to add an upstream remote reference to the original repository to update your fork with the latest changes, by adding to the `./hg/hgrc` file the following:

```
[paths]
default = ...
openpyxl-master = https://bitbucket.org/openpyxl/openpyxl
```

You can then grab any new changes using:

```
$ hg pull openpyxl-master
```

After that you should create a virtual environment using `virtualenv` and install the project requirements and the project itself:

```
$ cd openpyxl
$ virtualenv openpyxl-env
```

Activate the environment using:

```
$ source bin/activate  # or ./openpyxl-env/Scripts/activate on Windows
```

Install the dev and prod dependencies and the package itself using:

```
(openpyxl-env) $ pip install -U -r requirements.txt
(openpyxl-env) $ pip install -e .
```

7.1.2 Running tests

Note that contributions to the project without tests will **not** be accepted.

We use `pytest` as the test runner with `pytest-cov` for coverage information and `pytest-flakes` for static code analysis.

To run all the tests you need to either execute:

```
(openpxyl-env) $ pytest -xrf openpyxl  # the flags will stop testing at the␣
↪first error
```

Or use `tox` to run the tests on different Python versions and configurations:

```
$ tox openpyxl
```

Coverage

The goal is 100 % coverage for unit tests - data types and utility functions. Coverage information can be obtained using:

```
py.test --cov openpyxl
```

Organisation

Tests should be preferably at package / module level e.g `openpyxl/cell`. This makes testing and getting statistics for code under development easier:

```
py.test --cov openpyxl/cell openpyxl/cell
```

Checking XML

Use the `openpyxl.tests.helper.compare_xml` function to compare generated and expected fragments of XML.

Schema validation

When working on code to generate XML it is possible to validate that the generated XML conforms to the published specification. Note, this won’ t necessarily guarantee that everything is fine but is preferable to reverse engineering!

Microsoft Tools

Along with the SDK, Microsoft also has a “Productivity Tool” for working with Office OpenXML.

This allows you to quickly inspect or compare whole Excel files. Unfortunately, validation errors contain many false positives. The tool also contain links to the specification and implementers’ notes.

7.1.3 File Support and Specifications

The primary aim of openpyxl is to support reading and writing Microsoft Excel 2010 files. These are zipped OOXML files that are specified by ECMA 376 and ISO 29500.

Where possible we try to support files generated by other libraries or programs, but can’ t guarantee it, because often these do not strictly adhere to the above format.

7.1.4 Support of Python Versions

Python 3.6 and upwards are supported

7.1.5 Coding style

We orient ourselves at PEP-8 for the coding style, except when implementing attributes for roundtripping. Despite that you are encouraged to use Python data conventions (boolean, None, etc.). Note exceptions from this convestion in docstrings.

7.1.6 Contributing

Contributions in the form of pull requests are always welcome. Don' t forget to add yourself to the list of authors!

7.1.7 Branch naming convention

We use a "major.minor.patch" numbering system, ie. 3.0.7. Development branches are named after "major.minor" releases. In general, API change will only happen major releases but there will be exceptions. Always communicate API changes to the mailing list before making them. If you are changing an API try and an implement a fallback (with deprecation warning) for the old behaviour.

The "default branch" is used for releases and always has changes from a development branch merged in. It should never be the target for a pull request.

7.1.8 Pull Requests

Pull requests should be submitted to the current, unreleased development branch. Eg. if the current release is 3.0.7, pull requests should be made to the 3.0 branch. Exceptions are bug fixes to released versions which should be made to the relevant release branch and merged upstream into development.

Please use `tox` to test code for different submissions **before** making a pull request. This is especially important for picking up problems across Python versions.

Documentation

Remember to update the documentation when adding or changing features. Check that documentation is syntactically correct.:

```
tox -e doc
```

7.1.9 Benchmarking

Benchmarking and profiling are ongoing tasks. Contributions to these are very welcome as we know there is a lot to do.

Memory Use

There is a tox profile for long-running memory benchmarks using the *memory_utils* package.:

```
tox -e memory
```

Pympler

As openpyxl does not include any internal memory benchmarking tools, the python *pympler* package was used during the testing of styles to profile the memory usage in `openpyxl.reader.excel.read_style_table()`:

```
# in openpyxl/reader/style.py
from pympler import muppy, summary

def read_style_table(xml_source):
  ...
  if cell_xfs is not None:  # ~ line 47
      initialState = summary.summarize(muppy.get_objects())  # Capture the␣
↪initial state
      for index, cell_xfs_node in enumerate(cell_xfs_nodes):
         ...
         table[index] = new_style
      finalState = summary.summarize(muppy.get_objects())  # Capture the final␣
↪state
      diff = summary.get_diff(initialState, finalState)  # Compare
      summary.print_(diff)
```

`pympler.summary.print_()` prints to the console a report of object memory usage, allowing the comparison of different methods and examination of memory usage. A useful future development would be to construct a benchmarking package to measure the performance of different components.

CHAPTER 8

API 文档

8.1 关键类

- *openpyxl.workbook.workbook.Workbook*
- *openpyxl.worksheet.worksheet.Worksheet*
- *openpyxl.cell.cell.Cell*

8.2 完整 API

8.2.1 openpyxl package

Subpackages

openpyxl.cell package

Submodules

openpyxl.cell.cell module

Manage individual cells in a spreadsheet.

The Cell class is required to know its value and type, display options, and any other features of an Excel cell. Utilities for referencing cells using Excel’s ‘A1’ column/row nomenclature are also provided.

class openpyxl.cell.cell.Cell(*worksheet*, *row=None*, *column=None*, *value=None*, *style_array=None*)

基类：*openpyxl.styles.styleable.StyleableObject*

Describes cell associated properties.

Properties of interest include style, type, value, and address.

base_date

check_error(*value*)

Tries to convert Error” else N/A

check_string(*value*)

Check string coding, length, and line break character

col_idx

The numerical index of the column

column

Column number of this cell (1-based)

column_letter

comment

Returns the comment associated with this cell

Type openpyxl.comments.Comment

coordinate

This cell’s coordinate (ex. ‘A5’)

data_type

encoding

hyperlink

Return the hyperlink target or an empty string

internal_value

Always returns the value for excel.

is_date

True if the value is formatted as a date

Type bool

offset(*row=0*, *column=0*)

Returns a cell location relative to this cell.

参数

- **row** (*int*) – number of rows to offset
- **column** (*int*) – number of columns to offset

返回类型 openpyxl.cell.Cell

parent

row

Row number of this cell (1-based)

value

Get or set the value held in the cell.

Type depends on the value (string, float, int or datetime.datetime)

class openpyxl.cell.cell.**MergedCell**(*worksheet*, *row=None*, *column=None*)

基类：*openpyxl.styles.styleable.StyleableObject*

Describes the properties of a cell in a merged cell and helps to display the borders of the merged cell.

The value of a MergedCell is always None.

column

comment = None

coordinate

This cell’s coordinate (ex. ‘A5’)

data_type = 'n'

hyperlink = None

row

value = None

openpyxl.cell.cell.**WriteOnlyCell**(*ws=None*, *value=None*)

openpyxl.cell.cell.**get_time_format**(*t*)

openpyxl.cell.cell.**get_type**(*t*, *value*)

openpyxl.cell.read_only module

class openpyxl.cell.read_only.**EmptyCell**

基类：object

alignment = None

border = None

data_type = 'n'

fill = None

font = None

is_date = False

number_format = None

value = None

class openpyxl.cell.read_only.**ReadOnlyCell**(*sheet*, *row*, *column*, *value*, *data_type='n'*, *style_id=0*)

基类：object

alignment

border

column

column_letter

coordinate

data_type

fill

font

has_style

internal_value

is_date

number_format

parent

protection

row

style_array

value

openpyxl.cell.text module

Richtext definition

class openpyxl.cell.text.InlineFont(*rFont=None*, *charset=None*, *family=None*, *b=None*, *i=None*, *strike=None*, *outline=None*, *shadow=None*, *condense=None*, *extend=None*, *color=None*, *sz=None*, *u=None*, *vertAlign=None*, *scheme=None*)

基类：*openpyxl.styles.fonts.Font*

Font for inline text because, yes what you need are different objects with the same elements but different constraints.

b

Values must be of type <class 'bool'>

charset

Values must be of type <class 'int'>

color

Values must be of type <class 'openpyxl.styles.colors.Color'>

condense

Values must be of type <class 'bool'>

extend

Values must be of type <class 'bool'>

family

Values must be of type <class 'float'>

i

Values must be of type <class 'bool'>

outline

Values must be of type <class 'bool'>

rFont

Values must be of type <class 'str'>

scheme

Value must be one of {'major', 'minor'}

shadow

Values must be of type <class 'bool'>

strike

Values must be of type <class 'bool'>

sz

Values must be of type <class 'float'>

tagname = 'RPrElt'

u

Value must be one of {'doubleAccounting', 'double', 'singleAccounting', 'single'}

vertAlign

Value must be one of { 'subscript' , 'baseline' , 'superscript' }

`class openpyxl.cell.text.`**`PhoneticProperties`**`(`*fontId=None, type=None, alignment=None*`)`

基类: *openpyxl.descriptors.serialisable.Serialisable*

alignment

Value must be one of { 'left' , 'noControl' , 'distributed' , 'center' }

fontId

Values must be of type <class 'int' >

tagname = 'phoneticPr'

type

Value must be one of { 'fullwidthKatakana' , 'Hiragana' , 'halfwidthKatakana' , 'noConversion' }

`class openpyxl.cell.text.`**`PhoneticText`**`(`*sb=None, eb=None, t=None*`)`

基类: *openpyxl.descriptors.serialisable.Serialisable*

eb

Values must be of type <class 'int' >

sb

Values must be of type <class 'int' >

t

Values must be of type <class 'str' >

tagname = 'rPh'

text

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`class openpyxl.cell.text.`**`RichText`**`(`*rPr=None, t=None*`)`

基类: *openpyxl.descriptors.serialisable.Serialisable*

font

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

rPr

Values must be of type <class 'openpyxl.cell.text.InlineFont' >

t

Values must be of type <class 'str' >

tagname = 'RElt'

`text`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`class openpyxl.cell.text.Text(`*t=None*, *r=()*, *rPh=()*, *phoneticPr=None*`)`

基类： *openpyxl.descriptors.serialisable.Serialisable*

`PhoneticProperties`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`content`

Text stripped of all formatting

`formatted`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`phonetic`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`phoneticPr`

Values must be of type <class 'openpyxl.cell.text.PhoneticProperties' >

`plain`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`r`

A sequence (list or tuple) that may only contain objects of the declared type

`rPh`

A sequence (list or tuple) that may only contain objects of the declared type

`t`

Values must be of type <class 'str' >

`tagname = 'text'`

openpyxl.chart package

Submodules

openpyxl.chart.area_chart module

`class openpyxl.chart.area_chart.AreaChart(`*axId=None*, *extLst=None*, ***kw*`)`

基类： `openpyxl.chart.area_chart._AreaChartBase`

dLbls

Values must be of type <class ‘openpyxl.chart.label.DataLabelList’ >

dropLines

Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

grouping

Value must be one of { ‘standard’ , ‘stacked’ , ‘percentStacked’ }

ser

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'areaChart'

varyColors

Values must be of type <class ‘bool’ >

x_axis

Values must be of type <class ‘openpyxl.chart.axis.TextAxis’ >

y_axis

Values must be of type <class ‘openpyxl.chart.axis.NumericAxis’ >

class openpyxl.chart.area_chart.**AreaChart3D**(*gapDepth=None*, ***kw*)

基类：*openpyxl.chart.area_chart.AreaChart*

dLbls

Values must be of type <class ‘openpyxl.chart.label.DataLabelList’ >

dropLines

Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

gapDepth

Values must be of type <class ‘float’ >

grouping

Value must be one of { ‘standard’ , ‘stacked’ , ‘percentStacked’ }

ser

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'area3DChart'

varyColors

Values must be of type <class ‘bool’ >

x_axis

Values must be of type <class ‘openpyxl.chart.axis.TextAxis’ >

y_axis
Values must be of type <class ‘openpyxl.chart.axis.NumericAxis’ >

z_axis
Values must be of type <class ‘openpyxl.chart.axis.SeriesAxis’ >

openpyxl.chart.axis module

class openpyxl.chart.axis.**ChartLines**(*spPr=None*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

graphicalProperties
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

spPr
Values must be of type <class ‘openpyxl.chart.shapes.GraphicalProperties’ >

tagname = 'chartLines'

class openpyxl.chart.axis.**DateAxis**(*auto=None, lblOffset=None, baseTimeUnit=None, majorUnit=None, majorTimeUnit=None, minorUnit=None, minorTimeUnit=None, extLst=None, **kw*)
基类：*openpyxl.chart.axis.TextAxis*

auto
Values must be of type <class ‘bool’ >

axId
Values must be of type <class ‘int’ >

axPos
Value must be one of { ‘l’ , ‘t’ , ‘b’ , ‘r’ }

baseTimeUnit
Value must be one of { ‘months’ , ‘years’ , ‘days’ }

crossAx
Values must be of type <class ‘int’ >

crosses
Value must be one of { ‘max’ , ‘autoZero’ , ‘min’ }

crossesAt
Values must be of type <class ‘float’ >

delete
Values must be of type <class ‘bool’ >

extLst
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

lblOffset
Values must be of type <class 'int'>

majorGridlines
Values must be of type <class 'openpyxl.chart.axis.ChartLines'>

majorTickMark
Value must be one of {'cross', 'out', 'in'}

majorTimeUnit
Value must be one of {'months', 'years', 'days'}

majorUnit
Values must be of type <class 'float'>

minorGridlines
Values must be of type <class 'openpyxl.chart.axis.ChartLines'>

minorTickMark
Value must be one of {'cross', 'out', 'in'}

minorTimeUnit
Value must be one of {'months', 'years', 'days'}

minorUnit
Values must be of type <class 'float'>

numFmt
Values must be of type <class 'openpyxl.chart.data_source.NumFmt'>

scaling
Values must be of type <class 'openpyxl.chart.axis.Scaling'>

spPr
Values must be of type <class 'openpyxl.chart.shapes.GraphicalProperties'>

tagname = 'dateAx'

tickLblPos
Value must be one of {'low', 'nextTo', 'high'}

title
Values must be of type <class 'openpyxl.chart.title.Title'>

txPr
Values must be of type <class 'openpyxl.chart.text.RichText'>

class openpyxl.chart.axis.DisplayUnitsLabel(*layout=None*, *tx=None*, *spPr=None*, *txPr=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

graphicalProperties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

layout

Values must be of type <class ‘openpyxl.chart.layout.Layout’ >

spPr

Values must be of type <class ‘openpyxl.chart.shapes.GraphicalProperties’ >

tagname = 'dispUnitsLbl'

text

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

textPropertes

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

tx

Values must be of type <class ‘openpyxl.chart.text.Text’ >

txPr

Values must be of type <class ‘openpyxl.chart.text.RichText’ >

class openpyxl.chart.axis.DisplayUnitsLabelList(*custUnit=None*, *builtInUnit=None*, *dispUnitsLbl=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

builtInUnit

Value must be one of { ‘billions’ , ‘hundreds’ , ‘tenThousands’ , ‘hundredMillions’ , ‘trillions’ , ‘tenMillions’ , ‘millions’ , ‘thousands’ , ‘hundredThousands’ }

custUnit

Values must be of type <class ‘float’ >

dispUnitsLbl

Values must be of type <class ‘openpyxl.chart.axis.DisplayUnitsLabel’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

tagname = 'dispUnits'

class openpyxl.chart.axis.NumericAxis(*crossBetween=None*, *majorUnit=None*, *minorUnit=None*, *dispUnits=None*, *extLst=None*, ***kw*)

基类：openpyxl.chart.axis._BaseAxis

`axId`

Values must be of type <class ‘int’ >

`axPos`

Value must be one of { ‘l’ , ‘t’ , ‘b’ , ‘r’ }

`crossAx`

Values must be of type <class ‘int’ >

`crossBetween`

Value must be one of { ‘midCat’ , ‘between’ }

`crosses`

Value must be one of { ‘max’ , ‘autoZero’ , ‘min’ }

`crossesAt`

Values must be of type <class ‘float’ >

`delete`

Values must be of type <class ‘bool’ >

`dispUnits`

Values must be of type <class ‘openpyxl.chart.axis.DisplayUnitsLabelList’ >

`extLst`

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

`classmethod from_tree`(*node*)

Special case value axes with no gridlines

`majorGridlines`

Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

`majorTickMark`

Value must be one of { ‘cross’ , ‘out’ , ‘in’ }

`majorUnit`

Values must be of type <class ‘float’ >

`minorGridlines`

Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

`minorTickMark`

Value must be one of { ‘cross’ , ‘out’ , ‘in’ }

`minorUnit`

Values must be of type <class ‘float’ >

`numFmt`

Values must be of type <class ‘openpyxl.chart.data_source.NumFmt’ >

scaling

Values must be of type <class ‘openpyxl.chart.axis.Scaling’ >

spPr

Values must be of type <class ‘openpyxl.chart.shapes.GraphicalProperties’ >

tagname = 'valAx'

tickLblPos

Value must be one of { ‘low’ , ‘nextTo’ , ‘high’ }

title

Values must be of type <class ‘openpyxl.chart.title.Title’ >

txPr

Values must be of type <class ‘openpyxl.chart.text.RichText’ >

class openpyxl.chart.axis.**Scaling**(*logBase=None*, *orientation='minMax'*, *max=None*, *min=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

logBase

Values must be of type <class ‘float’ >

max

Values must be of type <class ‘float’ >

min

Values must be of type <class ‘float’ >

orientation

Value must be one of { ‘minMax’ , ‘maxMin’ }

tagname = 'scaling'

class openpyxl.chart.axis.**SeriesAxis**(*tickLblSkip=None*, *tickMarkSkip=None*, *extLst=None*, ***kw*)

基类：openpyxl.chart.axis._BaseAxis

axId

Values must be of type <class ‘int’ >

axPos

Value must be one of { ‘l’ , ‘t’ , ‘b’ , ‘r’ }

crossAx

Values must be of type <class ‘int’ >

crosses

Value must be one of { ‘max’ , ‘autoZero’ , ‘min’ }

crossesAt
Values must be of type <class ‘float’ >

delete
Values must be of type <class ‘bool’ >

extLst
Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

majorGridlines
Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

majorTickMark
Value must be one of { ‘cross’ , ‘out’ , ‘in’ }

minorGridlines
Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

minorTickMark
Value must be one of { ‘cross’ , ‘out’ , ‘in’ }

numFmt
Values must be of type <class ‘openpyxl.chart.data_source.NumFmt’ >

scaling
Values must be of type <class ‘openpyxl.chart.axis.Scaling’ >

spPr
Values must be of type <class ‘openpyxl.chart.shapes.GraphicalProperties’ >

tagname = 'serAx'

tickLblPos
Value must be one of { ‘low’ , ‘nextTo’ , ‘high’ }

tickLblSkip
Values must be of type <class ‘int’ >

tickMarkSkip
Values must be of type <class ‘int’ >

title
Values must be of type <class ‘openpyxl.chart.title.Title’ >

txPr
Values must be of type <class ‘openpyxl.chart.text.RichText’ >

class openpyxl.chart.axis.**TextAxis**(*auto=None*, *lblAlgn=None*, *lblOffset=100*, *tickLblSkip=None*, *tickMarkSkip=None*, *noMultiLvlLbl=None*, *extLst=None*, ***kw*)

基类：openpyxl.chart.axis._BaseAxis

`auto`
: Values must be of type <class 'bool'>

`axId`
: Values must be of type <class 'int'>

`axPos`
: Value must be one of {'l', 't', 'b', 'r'}

`crossAx`
: Values must be of type <class 'int'>

`crosses`
: Value must be one of {'max', 'autoZero', 'min'}

`crossesAt`
: Values must be of type <class 'float'>

`delete`
: Values must be of type <class 'bool'>

`extLst`
: Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

`lblAlgn`
: Value must be one of {'l', 'r', 'ctr'}

`lblOffset`
: Values must be of type <class 'float'>

`majorGridlines`
: Values must be of type <class 'openpyxl.chart.axis.ChartLines'>

`majorTickMark`
: Value must be one of {'cross', 'out', 'in'}

`minorGridlines`
: Values must be of type <class 'openpyxl.chart.axis.ChartLines'>

`minorTickMark`
: Value must be one of {'cross', 'out', 'in'}

`noMultiLvlLbl`
: Values must be of type <class 'bool'>

`numFmt`
: Values must be of type <class 'openpyxl.chart.data_source.NumFmt'>

`scaling`
: Values must be of type <class 'openpyxl.chart.axis.Scaling'>

spPr
Values must be of type <class 'openpyxl.chart.shapes.GraphicalProperties'>

tagname = 'catAx'

tickLblPos
Value must be one of {'low', 'nextTo', 'high'}

tickLblSkip
Values must be of type <class 'int'>

tickMarkSkip
Values must be of type <class 'int'>

title
Values must be of type <class 'openpyxl.chart.title.Title'>

txPr
Values must be of type <class 'openpyxl.chart.text.RichText'>

openpyxl.chart.bar_chart module

class openpyxl.chart.bar_chart.**BarChart**(*gapWidth=150*, *overlap=None*, *serLines=None*, *extLst=None*, ***kw*)

基类: openpyxl.chart.bar_chart._BarChartBase

barDir
Value must be one of {'bar', 'col'}

dLbls
Values must be of type <class 'openpyxl.chart.label.DataLabelList'>

extLst
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

gapWidth
Values must be of type <class 'float'>

grouping
Value must be one of {'standard', 'clustered', 'stacked', 'percentStacked'}

overlap
Values must be of type <class 'float'>

ser
A sequence (list or tuple) that may only contain objects of the declared type

serLines
Values must be of type <class 'openpyxl.chart.axis.ChartLines'>

tagname = 'barChart'

varyColors

Values must be of type <class 'bool'>

x_axis

Values must be of type <class 'openpyxl.chart.axis.TextAxis'>

y_axis

Values must be of type <class 'openpyxl.chart.axis.NumericAxis'>

class openpyxl.chart.bar_chart.**BarChart3D**(*gapWidth=150*, *gapDepth=150*, *shape=None*, *serLines=None*, *extLst=None*, ***kw*)

基类：openpyxl.chart.bar_chart._BarChartBase, openpyxl.chart._3d._3DBase

backWall

Values must be of type <class 'openpyxl.chart._3d.Surface'>

barDir

Value must be one of {'bar', 'col'}

dLbls

Values must be of type <class 'openpyxl.chart.label.DataLabelList'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

floor

Values must be of type <class 'openpyxl.chart._3d.Surface'>

gapDepth

Values must be of type <class 'float'>

gapWidth

Values must be of type <class 'float'>

grouping

Value must be one of {'standard', 'clustered', 'stacked', 'percentStacked'}

ser

A sequence (list or tuple) that may only contain objects of the declared type

serLines

Values must be of type <class 'openpyxl.chart.axis.ChartLines'>

shape

Value must be one of {'cone', 'pyramid', 'cylinder', 'box', 'pyramidToMax', 'coneToMax'}

sideWall

Values must be of type <class 'openpyxl.chart._3d.Surface'>

tagname = 'bar3DChart'

`varyColors`
Values must be of type <class 'bool' >

`view3D`
Values must be of type <class 'openpyxl.chart._3d.View3D' >

`x_axis`
Values must be of type <class 'openpyxl.chart.axis.TextAxis' >

`y_axis`
Values must be of type <class 'openpyxl.chart.axis.NumericAxis' >

`z_axis`
Values must be of type <class 'openpyxl.chart.axis.SeriesAxis' >

openpyxl.chart.bubble_chart module

`class openpyxl.chart.bubble_chart.BubbleChart(`*varyColors=None, ser=(), dLbls=None, bubble3D=None, bubbleScale=None, showNegBubbles=None, sizeRepresents=None, extLst=None, **kw*)

基类：`openpyxl.chart._chart.ChartBase`

`bubble3D`
Values must be of type <class 'bool' >

`bubbleScale`
Values must be of type <class 'float' >

`dLbls`
Values must be of type <class 'openpyxl.chart.label.DataLabelList' >

`dataLabels`
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`extLst`
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

`ser`
A sequence (list or tuple) that may only contain objects of the declared type

`showNegBubbles`
Values must be of type <class 'bool' >

`sizeRepresents`
Value must be one of { 'w' , 'area' }

`tagname = 'bubbleChart'`

varyColors

Values must be of type <class 'bool'>

x_axis

Values must be of type <class 'openpyxl.chart.axis.NumericAxis'>

y_axis

Values must be of type <class 'openpyxl.chart.axis.NumericAxis'>

openpyxl.chart.chartspace module

`class openpyxl.chart.chartspace.ChartContainer`(*title=None, autoTitleDeleted=None, pivotFmts=(), view3D=None, floor=None, sideWall=None, backWall=None, plotArea=None, legend=None, plotVisOnly=True, dispBlanksAs='gap', showDLblsOverMax=None, extLst=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

autoTitleDeleted

Values must be of type <class 'bool'>

backWall

Values must be of type <class 'openpyxl.chart._3d.Surface'>

dispBlanksAs

Value must be one of {'zero', 'gap', 'span'}

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

floor

Values must be of type <class 'openpyxl.chart._3d.Surface'>

legend

Values must be of type <class 'openpyxl.chart.legend.Legend'>

pivotFmts

Wrap a sequence in an containing object

plotArea

Values must be of type <class 'openpyxl.chart.plotarea.PlotArea'>

plotVisOnly

Values must be of type <class 'bool'>

showDLblsOverMax

Values must be of type <class 'bool'>

sideWall

Values must be of type <class 'openpyxl.chart._3d.Surface'>

tagname = 'chart'

title

Values must be of type <class 'openpyxl.chart.title.Title'>

view3D

Values must be of type <class 'openpyxl.chart._3d.View3D'>

class openpyxl.chart.chartspace.**ChartSpace**(*date1904=None, lang=None, roundedCorners=None, style=None, clrMapOvr=None, pivotSource=None, protection=None, chart=None, spPr=None, txPr=None, externalData=None, printSettings=None, userShapes=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

chart

Values must be of type <class 'openpyxl.chart.chartspace.ChartContainer'>

clrMapOvr

Values must be of type <class 'openpyxl.drawing.colors.ColorMapping'>

date1904

Values must be of type <class 'bool'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

externalData

Values must be of type <class 'openpyxl.chart.chartspace.ExternalData'>

graphicalProperties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

lang

Values must be of type <class 'str'>

pivotSource

Values must be of type <class 'openpyxl.chart.pivot.PivotSource'>

printSettings

Values must be of type <class 'openpyxl.chart.print_settings.PrintSettings'>

protection

Values must be of type <class 'openpyxl.chart.chartspace.Protection'>

roundedCorners
Values must be of type <class 'bool' >

spPr
Values must be of type <class 'openpyxl.chart.shapes.GraphicalProperties' >

style
Values must be of type <class 'float' >

tagname = 'chartSpace'

textProperties
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

to_tree(*tagname=None*, *idx=None*, *namespace=None*)

txPr
Values must be of type <class 'openpyxl.chart.text.RichText' >

userShapes
Values must be of type <class 'str' >

class openpyxl.chart.chartspace.**ExternalData**(*autoUpdate=None*, *id=None*)
基类: *openpyxl.descriptors.serialisable.Serialisable*

autoUpdate
Values must be of type <class 'bool' >

id
Values must be of type <class 'str' >

tagname = 'externalData'

class openpyxl.chart.chartspace.**Protection**(*chartObject=None*, *data=None*, *formatting=None*, *selection=None*, *userInterface=None*)
基类: *openpyxl.descriptors.serialisable.Serialisable*

chartObject
Values must be of type <class 'bool' >

data
Values must be of type <class 'bool' >

formatting
Values must be of type <class 'bool' >

selection
Values must be of type <class 'bool' >

tagname = 'protection'

userInterface
Values must be of type <class 'bool'>

openpyxl.chart.data_source module

Collection of utility primitives for charts.

class openpyxl.chart.data_source.**AxDataSource**(*numRef=None*, *numLit=None*, *strRef=None*, *strLit=None*, *multiLvlStrRef=None*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

multiLvlStrRef
Values must be of type <class 'openpyxl.chart.data_source.MultiLevelStrRef'>

numLit
Values must be of type <class 'openpyxl.chart.data_source.NumData'>

numRef
Values must be of type <class 'openpyxl.chart.data_source.NumRef'>

strLit
Values must be of type <class 'openpyxl.chart.data_source.StrData'>

strRef
Values must be of type <class 'openpyxl.chart.data_source.StrRef'>

tagname = 'cat'

class openpyxl.chart.data_source.**Level**(*pt=()*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

pt
A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'lvl'

class openpyxl.chart.data_source.**MultiLevelStrData**(*ptCount=None*, *lvl=()*, *extLst=None*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

lvl
A sequence (list or tuple) that may only contain objects of the declared type

ptCount
Values must be of type <class 'int'>

tagname = 'multiLvlStrData'

class openpyxl.chart.data_source.**MultiLevelStrRef**(*f=None*, *multiLvlStrCache=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

f

Values must be of type <class 'str'>

multiLvlStrCache

Values must be of type <class 'openpyxl.chart.data_source.MultiLevelStrData'>

tagname = 'multiLvlStrRef'

class openpyxl.chart.data_source.**NumData**(*formatCode=None*, *ptCount=None*, *pt=()*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

formatCode

Values must be of type <class 'str'>

pt

A sequence (list or tuple) that may only contain objects of the declared type

ptCount

Values must be of type <class 'int'>

class openpyxl.chart.data_source.**NumDataSource**(*numRef=None*, *numLit=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

numLit

Values must be of type <class 'openpyxl.chart.data_source.NumData'>

numRef

Values must be of type <class 'openpyxl.chart.data_source.NumRef'>

class openpyxl.chart.data_source.**NumFmt**(*formatCode=None*, *sourceLinked=False*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

formatCode

Values must be of type <class 'str'>

sourceLinked

Values must be of type <class 'bool'>

class openpyxl.chart.data_source.**NumRef**(*f=None*, *numCache=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

f

Values must be of type <class 'str'>

numCache

Values must be of type <class 'openpyxl.chart.data_source.NumData'>

ref

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

class openpyxl.chart.data_source.**NumVal**(*idx=None*, *formatCode=None*, *v=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

formatCode

Values must be of type <class 'str'>

idx

Values must be of type <class 'int'>

v

Values must be of type <class 'NoneType'>

class openpyxl.chart.data_source.**NumberValueDescriptor**(**args*, ***kw*)

基类: *openpyxl.descriptors.nested.NestedText*

Data should be numerical but isn't always :-/

allow_none = True

class openpyxl.chart.data_source.**StrData**(*ptCount=None*, *pt=()*, *extLst=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

pt

A sequence (list or tuple) that may only contain objects of the declared type

ptCount

Values must be of type <class 'int'>

tagname = 'strData'

class openpyxl.chart.data_source.**StrRef**(*f=None*, *strCache=None*, *extLst=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

f

Values must be of type <class 'str'>

strCache

Values must be of type <class 'openpyxl.chart.data_source.StrData'>

tagname = 'strRef'

class openpyxl.chart.data_source.**StrVal**(*idx=0*, *v=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

idx

Values must be of type <class 'int'>

tagname = 'strVal'

v

Values must be of type <class 'str'>

openpyxl.chart.descriptors module

class openpyxl.chart.descriptors.**NestedGapAmount**(***kw*)

基类: *openpyxl.descriptors.nested.NestedMinMax*

allow_none = True

max = 500

min = 0

class openpyxl.chart.descriptors.**NestedOverlap**(***kw*)

基类: *openpyxl.descriptors.nested.NestedMinMax*

allow_none = True

max = 100

min = -100

class openpyxl.chart.descriptors.**NumberFormatDescriptor**(**args*, ***kw*)

基类: *openpyxl.descriptors.base.Typed*

Allow direct assignment of format code

allow_none = True

expected_type

openpyxl.chart.data_source.NumFmt 的别名

openpyxl.chart.error_bar module

class `openpyxl.chart.error_bar.ErrorBars`(*errDir=None*, *errBarType='both'*, *errValType='fixedVal'*, *noEndCap=None*, *plus=None*, *minus=None*, *val=None*, *spPr=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

direction

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

errBarType

Value must be one of { 'both' , 'minus' , 'plus' }

errDir

Value must be one of { 'y' , 'x' }

errValType

Value must be one of { 'percentage' , 'stdDev' , 'stdErr' , 'cust' , 'fixedVal' }

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

graphicalProperties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

minus

Values must be of type <class 'openpyxl.chart.data_source.NumDataSource' >

noEndCap

Values must be of type <class 'bool' >

plus

Values must be of type <class 'openpyxl.chart.data_source.NumDataSource' >

size

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

spPr

Values must be of type <class 'openpyxl.chart.shapes.GraphicalProperties' >

style

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

tagname = 'errBars'

val

Values must be of type <class ‘float’ >

openpyxl.chart.label module

class openpyxl.chart.label.**DataLabel**(*idx=0*, ***kw*)

基类: openpyxl.chart.label._DataLabelBase

dLblPos

Value must be one of { ‘b’ , ‘bestFit’ , ‘r’ , ‘outEnd’ , ‘inEnd’ , ‘inBase’ , ‘ctr’ , ‘l’ , ‘t’ }

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

idx

Values must be of type <class ‘int’ >

numFmt

Values must be of type <class ‘str’ >

separator

Values must be of type <class ‘str’ >

showBubbleSize

Values must be of type <class ‘bool’ >

showCatName

Values must be of type <class ‘bool’ >

showLeaderLines

Values must be of type <class ‘bool’ >

showLegendKey

Values must be of type <class ‘bool’ >

showPercent

Values must be of type <class ‘bool’ >

showSerName

Values must be of type <class ‘bool’ >

showVal

Values must be of type <class ‘bool’ >

spPr

Values must be of type <class ‘openpyxl.chart.shapes.GraphicalProperties’ >

tagname = 'dLbl'

txPr

Values must be of type <class ‘openpyxl.chart.text.RichText’ >

class openpyxl.chart.label.**DataLabelList**(*dLbl=()*, *delete=None*, ***kw*)

基类：openpyxl.chart.label._DataLabelBase

dLbl

A sequence (list or tuple) that may only contain objects of the declared type

dLblPos

Value must be one of { ‘b’ , ‘bestFit’ , ‘r’ , ‘outEnd’ , ‘inEnd’ , ‘inBase’ , ‘ctr’ , ‘l’ , ‘t’ }

delete

Values must be of type <class ‘bool’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

numFmt

Values must be of type <class ‘str’ >

separator

Values must be of type <class ‘str’ >

showBubbleSize

Values must be of type <class ‘bool’ >

showCatName

Values must be of type <class ‘bool’ >

showLeaderLines

Values must be of type <class ‘bool’ >

showLegendKey

Values must be of type <class ‘bool’ >

showPercent

Values must be of type <class ‘bool’ >

showSerName

Values must be of type <class ‘bool’ >

showVal

Values must be of type <class ‘bool’ >

spPr

Values must be of type <class ‘openpyxl.chart.shapes.GraphicalProperties’ >

tagname = 'dLbls'

txPr
Values must be of type <class ‘openpyxl.chart.text.RichText’ >

openpyxl.chart.layout module

`class openpyxl.chart.layout.Layout(manualLayout=None, extLst=None)`
基类： *openpyxl.descriptors.serialisable.Serialisable*

extLst
Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

manualLayout
Values must be of type <class ‘openpyxl.chart.layout.ManualLayout’ >

tagname = 'layout'

`class openpyxl.chart.layout.ManualLayout(layoutTarget=None, xMode=None, yMode=None, wMode='factor', hMode='factor', x=None, y=None, w=None, h=None, extLst=None)`
基类： *openpyxl.descriptors.serialisable.Serialisable*

extLst
Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

h
Values must be of type <class ‘float’ >

hMode
Value must be one of { ‘factor’ , ‘edge’ }

height
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

layoutTarget
Value must be one of { ‘inner’ , ‘outer’ }

tagname = 'manualLayout'

w
Values must be of type <class ‘float’ >

wMode
Value must be one of { ‘factor’ , ‘edge’ }

width
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

x

Values must be of type <class 'float' >

xMode

Value must be one of { 'factor' , 'edge' }

y

Values must be of type <class 'float' >

yMode

Value must be one of { 'factor' , 'edge' }

openpyxl.chart.legend module

class openpyxl.chart.legend.Legend(*legendPos='r'*, *legendEntry=()*, *layout=None*, *overlay=None*, *spPr=None*, *txPr=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

graphicalProperties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

layout

Values must be of type <class 'openpyxl.chart.layout.Layout' >

legendEntry

A sequence (list or tuple) that may only contain objects of the declared type

legendPos

Value must be one of { 'b' , 'r' , 'tr' , 'l' , 't' }

overlay

Values must be of type <class 'bool' >

position

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

spPr

Values must be of type <class 'openpyxl.chart.shapes.GraphicalProperties' >

tagname = 'legend'

textProperties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

txPr

Values must be of type <class ‘openpyxl.chart.text.RichText’ >

class openpyxl.chart.legend.**LegendEntry**(*idx=0*, *delete=False*, *txPr=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

delete

Values must be of type <class ‘bool’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

idx

Values must be of type <class ‘int’ >

tagname = 'legendEntry'

txPr

Values must be of type <class ‘openpyxl.chart.text.RichText’ >

openpyxl.chart.line_chart module

class openpyxl.chart.line_chart.**LineChart**(*hiLowLines=None*, *upDownBars=None*, *marker=None*, *smooth=None*, *extLst=None*, ***kw*)

基类：openpyxl.chart.line_chart._LineChartBase

dLbls

Values must be of type <class ‘openpyxl.chart.label.DataLabelList’ >

dropLines

Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

grouping

Value must be one of { ‘standard’ , ‘stacked’ , ‘percentStacked’ }

hiLowLines

Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

marker

Values must be of type <class ‘bool’ >

ser

A sequence (list or tuple) that may only contain objects of the declared type

smooth

Values must be of type <class ‘bool’ >

tagname = 'lineChart'

upDownBars

Values must be of type <class ‘openpyxl.chart.updown_bars.UpDownBars’ >

varyColors

Values must be of type <class ‘bool’ >

x_axis

Values must be of type <class ‘openpyxl.chart.axis._BaseAxis’ >

y_axis

Values must be of type <class ‘openpyxl.chart.axis.NumericAxis’ >

class openpyxl.chart.line_chart.**LineChart3D**(*gapDepth=None*, *hiLowLines=None*, *upDownBars=None*, *marker=None*, *smooth=None*, ***kw*)

基类：openpyxl.chart.line_chart._LineChartBase

dLbls

Values must be of type <class ‘openpyxl.chart.label.DataLabelList’ >

dropLines

Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

gapDepth

Values must be of type <class ‘float’ >

grouping

Value must be one of { ‘standard’ , ‘stacked’ , ‘percentStacked’ }

hiLowLines

Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

marker

Values must be of type <class ‘bool’ >

ser

A sequence (list or tuple) that may only contain objects of the declared type

smooth

Values must be of type <class ‘bool’ >

tagname = 'line3DChart'

upDownBars

Values must be of type <class ‘openpyxl.chart.updown_bars.UpDownBars’ >

`varyColors`
Values must be of type <class 'bool'>

`x_axis`
Values must be of type <class 'openpyxl.chart.axis.TextAxis'>

`y_axis`
Values must be of type <class 'openpyxl.chart.axis.NumericAxis'>

`z_axis`
Values must be of type <class 'openpyxl.chart.axis.SeriesAxis'>

openpyxl.chart.marker module

`class openpyxl.chart.marker.DataPoint`(*idx=None*, *invertIfNegative=None*, *marker=None*, *bubble3D=None*, *explosion=None*, *spPr=None*, *pictureOptions=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`bubble3D`
Values must be of type <class 'bool'>

`explosion`
Values must be of type <class 'int'>

`extLst`
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

`graphicalProperties`
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`idx`
Values must be of type <class 'int'>

`invertIfNegative`
Values must be of type <class 'bool'>

`marker`
Values must be of type <class 'openpyxl.chart.marker.Marker'>

`pictureOptions`
Values must be of type <class 'openpyxl.chart.picture.PictureOptions'>

`spPr`
Values must be of type <class 'openpyxl.chart.shapes.GraphicalProperties'>

`tagname = 'dPt'`

`class openpyxl.chart.marker.Marker`(*symbol=None*, *size=None*, *spPr=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

graphicalProperties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

size

Values must be of type <class 'float'>

spPr

Values must be of type <class 'openpyxl.chart.shapes.GraphicalProperties'>

symbol

Value must be one of {'circle', 'auto', 'star', 'dot', 'picture', 'plus', 'x', 'diamond', 'dash', 'square', 'triangle'}

tagname = 'marker'

openpyxl.chart.picture module

`class openpyxl.chart.picture.PictureOptions`(*applyToFront=None*, *applyToSides=None*, *applyToEnd=None*, *pictureFormat=None*, *pictureStackUnit=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

applyToEnd

Values must be of type <class 'bool'>

applyToFront

Values must be of type <class 'bool'>

applyToSides

Values must be of type <class 'bool'>

pictureFormat

Value must be one of {'stack', 'stretch', 'stackScale'}

pictureStackUnit

Values must be of type <class 'float'>

tagname = 'pictureOptions'

openpyxl.chart.pie_chart module

class openpyxl.chart.pie_chart.**CustomSplit**(*secondPiePt=()*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

secondPiePt

A sequence of primitive types that are stored as a single attribute. “val” is the default attribute

tagname = 'custSplit'

class openpyxl.chart.pie_chart.**DoughnutChart**(*firstSliceAng=0*, *holeSize=10*, *extLst=None*, ***kw*)

基类: openpyxl.chart.pie_chart._PieChartBase

dLbls

Values must be of type <class ‘openpyxl.chart.label.DataLabelList’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

firstSliceAng

Values must be of type <class ‘float’ >

holeSize

Values must be of type <class ‘float’ >

ser

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'doughnutChart'

varyColors

Values must be of type <class ‘bool’ >

class openpyxl.chart.pie_chart.**PieChart**(*firstSliceAng=0*, *extLst=None*, ***kw*)

基类: openpyxl.chart.pie_chart._PieChartBase

dLbls

Values must be of type <class ‘openpyxl.chart.label.DataLabelList’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

firstSliceAng

Values must be of type <class ‘float’ >

ser

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'pieChart'

varyColors

Values must be of type <class 'bool'>

class openpyxl.chart.pie_chart.**PieChart3D**(*varyColors=True*, *ser=()*, *dLbls=None*)

基类：`openpyxl.chart.pie_chart._PieChartBase`

dLbls

Values must be of type <class 'openpyxl.chart.label.DataLabelList'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

ser

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'pie3DChart'

varyColors

Values must be of type <class 'bool'>

class openpyxl.chart.pie_chart.**ProjectedPieChart**(*ofPieType='pie'*, *gapWidth=None*, *splitType='auto'*, *splitPos=None*, *custSplit=None*, *secondPieSize=75*, *serLines=None*, *extLst=None*, ***kw*)

基类：`openpyxl.chart.pie_chart._PieChartBase`

From the spec 21.2.2.126

This element contains the pie of pie or bar of pie series on this chart. Only the first series shall be displayed. The splitType element shall determine whether the splitPos and custSplit elements apply.

custSplit

Values must be of type <class 'openpyxl.chart.pie_chart.CustomSplit'>

dLbls

Values must be of type <class 'openpyxl.chart.label.DataLabelList'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

gapWidth

Values must be of type <class 'float'>

join_lines

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

ofPieType

Value must be one of {'pie', 'bar'}

secondPieSize

Values must be of type <class 'float'>

`ser`

A sequence (list or tuple) that may only contain objects of the declared type

`serLines`

Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

`splitPos`

Values must be of type <class ‘float’ >

`splitType`

Value must be one of { ‘auto’ , ‘percent’ , ‘val’ , ‘cust’ , ‘pos’ }

`tagname = 'ofPieChart'`

`type`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

`varyColors`

Values must be of type <class ‘bool’ >

openpyxl.chart.pivot module

`class openpyxl.chart.pivot.PivotFormat`(*idx=0, spPr=None, txPr=None, marker=None, dLbl=None, extLst=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

`DataLabel`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

`TextBody`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

`dLbl`

Values must be of type <class ‘openpyxl.chart.label.DataLabel’ >

`extLst`

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

`graphicalProperties`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

`idx`

Values must be of type <class ‘int’ >

`marker`

Values must be of type <class ‘openpyxl.chart.marker.Marker’ >

spPr
Values must be of type <class 'openpyxl.chart.shapes.GraphicalProperties' >

tagname = 'pivotFmt'

txPr
Values must be of type <class 'openpyxl.chart.text.RichText' >

class openpyxl.chart.pivot.**PivotSource**(*name=None*, *fmtId=None*, *extLst=None*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

fmtId
Values must be of type <class 'int' >

name
Values must be of type <class 'str' >

tagname = 'pivotSource'

openpyxl.chart.plotarea module

class openpyxl.chart.plotarea.**DataTable**(*showHorzBorder=None*, *showVertBorder=None*, *showOutline=None*, *showKeys=None*, *spPr=None*, *txPr=None*, *extLst=None*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

graphicalProperties
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

showHorzBorder
Values must be of type <class 'bool' >

showKeys
Values must be of type <class 'bool' >

showOutline
Values must be of type <class 'bool' >

showVertBorder
Values must be of type <class 'bool' >

spPr
Values must be of type <class 'openpyxl.chart.shapes.GraphicalProperties' >

tagname = 'dTable'

txPr

Values must be of type <class ‘openpyxl.chart.text.RichText’ >

class openpyxl.chart.plotarea.**PlotArea**(*layout=None*, *dTable=None*, *spPr=None*, *_charts=()*, *_axes=()*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

area3DChart

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

areaChart

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

bar3DChart

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

barChart

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

bubbleChart

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

catAx

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

dTable

Values must be of type <class ‘openpyxl.chart.plotarea.DataTable’ >

dateAx

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

doughnutChart

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

classmethod from_tree(*node*)

Create object from XML

graphicalProperties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

layout

Values must be of type <class ‘openpyxl.chart.layout.Layout’ >

line3DChart

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

lineChart

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

ofPieChart

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

pie3DChart

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

pieChart

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

radarChart

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

scatterChart

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

serAx

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

spPr

Values must be of type <class ‘openpyxl.chart.shapes.GraphicalProperties’ >

`stockChart`

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

`surface3DChart`

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

`surfaceChart`

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

`tagname = 'plotArea'`

`to_tree`(*tagname=None*, *idx=None*, *namespace=None*)

`valAx`

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

openpyxl.chart.print_settings module

`class openpyxl.chart.print_settings.PageMargins`(*l=0.75*, *r=0.75*, *t=1*, *b=1*, *header=0.5*, *footer=0.5*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Identical to openpyxl.worksheet.page.Pagemargins but element names are different :-/

`b`

Values must be of type <class ‘float’ >

`bottom`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

`footer`

Values must be of type <class ‘float’ >

`header`

Values must be of type <class ‘float’ >

`l`

Values must be of type <class ‘float’ >

`left`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

r

Values must be of type <class ‘float’ >

right

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

t

Values must be of type <class ‘float’ >

tagname = 'pageMargins'

top

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

class openpyxl.chart.print_settings.**PrintSettings**(*headerFooter=None*, *pageMargins=None*, *pageSetup=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

headerFooter

Values must be of type <class ‘openpyxl.worksheet.header_footer.HeaderFooter’ >

pageMargins

Values must be of type <class ‘openpyxl.chart.print_settings.PageMargins’ >

pageSetup

Values must be of type <class ‘openpyxl.worksheet.page.PrintPageSetup’ >

tagname = 'printSettings'

openpyxl.chart.radar_chart module

class openpyxl.chart.radar_chart.**RadarChart**(*radarStyle='standard'*, *varyColors=None*, *ser=()*, *dLbls=None*, *extLst=None*, ***kw*)

基类：openpyxl.chart._chart.ChartBase

dLbls

Values must be of type <class ‘openpyxl.chart.label.DataLabelList’ >

dataLabels

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

radarStyle

Value must be one of { ‘marker’ , ‘standard’ , ‘filled’ }

ser

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'radarChart'

type

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

varyColors

Values must be of type <class 'bool' >

x_axis

Values must be of type <class 'openpyxl.chart.axis.TextAxis' >

y_axis

Values must be of type <class 'openpyxl.chart.axis.NumericAxis' >

openpyxl.chart.reader module

Read a chart

openpyxl.chart.reader.**read_chart**(*chartspace*)

openpyxl.chart.reference module

class openpyxl.chart.reference.**DummyWorksheet**(*title*)

基类：object

class openpyxl.chart.reference.**Reference**(*worksheet=None*, *min_col=None*, *min_row=None*, *max_col=None*, *max_row=None*, *range_string=None*)

基类：*openpyxl.descriptors.Strict*

Normalise cell range references

cols

Return all columns in the range

max_col

Values must be of type <class 'int' >

max_row

Values must be of type <class 'int' >

min_col

Values must be of type <class 'int' >

min_row

Values must be of type <class 'int'>

pop()

Return and remove the first cell

range_string

Values must be of type <class 'str'>

rows

Return all rows in the range

sheetname

openpyxl.chart.scatter_chart module

class openpyxl.chart.scatter_chart.**ScatterChart**(*scatterStyle=None*, *varyColors=None*, *ser=()*, *dLbls=None*, *extLst=None*, ***kw*)

基类：openpyxl.chart._chart.ChartBase

dLbls

Values must be of type <class 'openpyxl.chart.label.DataLabelList'>

dataLabels

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

scatterStyle

Value must be one of {'marker', 'lineMarker', 'smooth', 'line', 'smoothMarker'}

ser

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'scatterChart'

varyColors

Values must be of type <class 'bool'>

x_axis

Values must be of type <class 'openpyxl.chart.axis.NumericAxis'>

y_axis

Values must be of type <class 'openpyxl.chart.axis.NumericAxis'>

openpyxl.chart.series module

class openpyxl.chart.series.Series(*idx=0, order=0, tx=None, spPr=None, pictureOptions=None, dPt=(), dLbls=None, trendline=None, errBars=None, cat=None, val=None, invertIfNegative=None, shape=None, xVal=None, yVal=None, bubbleSize=None, bubble3D=None, marker=None, smooth=None, explosion=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Generic series object. Should not be instantiated directly. User the chart.Series factory instead.

bubble3D

Values must be of type <class 'bool'>

bubbleSize

Values must be of type <class 'openpyxl.chart.data_source.NumDataSource'>

cat

Values must be of type <class 'openpyxl.chart.data_source.AxDataSource'>

dLbls

Values must be of type <class 'openpyxl.chart.label.DataLabelList'>

dPt

A sequence (list or tuple) that may only contain objects of the declared type

data_points

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

errBars

Values must be of type <class 'openpyxl.chart.error_bar.ErrorBars'>

explosion

Values must be of type <class 'int'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

graphicalProperties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

identifiers

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

idx

Values must be of type <class 'int'>

invertIfNegative

Values must be of type <class 'bool'>

labels

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

marker

Values must be of type <class 'openpyxl.chart.marker.Marker'>

order

Values must be of type <class 'int'>

pictureOptions

Values must be of type <class 'openpyxl.chart.picture.PictureOptions'>

shape

Value must be one of {'cone', 'pyramid', 'cylinder', 'box', 'pyramidToMax', 'coneToMax'}

smooth

Values must be of type <class 'bool'>

spPr

Values must be of type <class 'openpyxl.chart.shapes.GraphicalProperties'>

tagname = 'ser'

title

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

to_tree(*tagname=None*, *idx=None*)

trendline

Values must be of type <class 'openpyxl.chart.trendline.Trendline'>

tx

Values must be of type <class 'openpyxl.chart.series.SeriesLabel'>

val

Values must be of type <class 'openpyxl.chart.data_source.NumDataSource'>

xVal

Values must be of type <class 'openpyxl.chart.data_source.AxDataSource'>

yVal

Values must be of type <class 'openpyxl.chart.data_source.NumDataSource'>

zVal

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

class openpyxl.chart.series.**SeriesLabel**(*strRef=None*, *v=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

strRef

Values must be of type <class ‘openpyxl.chart.data_source.StrRef’ >

tagname = 'tx'

v

Values must be of type <class ‘str’ >

value

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

class openpyxl.chart.series.**XYSeries**(*idx=0*, *order=0*, *tx=None*, *spPr=None*, *pictureOptions=None*, *dPt=()*, *dLbls=None*, *trendline=None*, *errBars=None*, *cat=None*, *val=None*, *invertIfNegative=None*, *shape=None*, *xVal=None*, *yVal=None*, *bubbleSize=None*, *bubble3D=None*, *marker=None*, *smooth=None*, *explosion=None*, *extLst=None*)

基类：*openpyxl.chart.series.Series*

Dedicated series for charts that have x and y series

bubble3D

Values must be of type <class ‘bool’ >

bubbleSize

Values must be of type <class ‘openpyxl.chart.data_source.NumDataSource’ >

dLbls

Values must be of type <class ‘openpyxl.chart.label.DataLabelList’ >

dPt

A sequence (list or tuple) that may only contain objects of the declared type

errBars

Values must be of type <class ‘openpyxl.chart.error_bar.ErrorBars’ >

idx

Values must be of type <class ‘int’ >

invertIfNegative

Values must be of type <class ‘bool’ >

marker

Values must be of type <class ‘openpyxl.chart.marker.Marker’ >

`order`
Values must be of type <class ‘int’ >

`smooth`
Values must be of type <class ‘bool’ >

`spPr`
Values must be of type <class ‘openpyxl.chart.shapes.GraphicalProperties’ >

`trendline`
Values must be of type <class ‘openpyxl.chart.trendline.Trendline’ >

`tx`
Values must be of type <class ‘openpyxl.chart.series.SeriesLabel’ >

`xVal`
Values must be of type <class ‘openpyxl.chart.data_source.AxDataSource’ >

`yVal`
Values must be of type <class ‘openpyxl.chart.data_source.NumDataSource’ >

openpyxl.chart.series_factory module

`openpyxl.chart.series_factory.SeriesFactory`(*values*, *xvalues=None*, *zvalues=None*, *title=None*, *title_from_data=False*)
Convenience Factory for creating chart data series.

openpyxl.chart.shapes module

`class openpyxl.chart.shapes.GraphicalProperties`(*bwMode=None*, *xfrm=None*, *noFill=None*, *solidFill=None*, *gradFill=None*, *pattFill=None*, *ln=None*, *scene3d=None*, *custGeom=None*, *prstGeom=None*, *sp3d=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Somewhat vaguely 21.2.2.197 says this:

This element specifies the formatting for the parent chart element. The custGeom, prstGeom, scene3d, and xfrm elements are not supported. The bwMode attribute is not supported.

This doesn’t leave much. And the element is used in different places.

`bwMode`
Value must be one of { ‘black’ , ‘blackGray’ , ‘clr’ , ‘blackWhite’ , ‘auto’ , ‘gray’ , ‘ltGray’ , ‘hidden’ , ‘invGray’ , ‘white’ , ‘grayWhite’ }

`custGeom`
Values must be of type <class ‘openpyxl.drawing.geometry.CustomGeometry2D’ >

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

gradFill

Values must be of type <class 'openpyxl.drawing.fill.GradientFillProperties'>

line

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

ln

Values must be of type <class 'openpyxl.drawing.line.LineProperties'>

noFill

Values must be of type <class 'bool'>

pattFill

Values must be of type <class 'openpyxl.drawing.fill.PatternFillProperties'>

prstGeom

Values must be of type <class 'openpyxl.drawing.geometry.PresetGeometry2D'>

scene3d

Values must be of type <class 'openpyxl.drawing.geometry.Scene3D'>

shape3D

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

solidFill

Values must be of type <class 'openpyxl.drawing.colors.ColorChoice'>

sp3d

Values must be of type <class 'openpyxl.drawing.geometry.Shape3D'>

tagname = 'spPr'

transform

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

xfrm

Values must be of type <class 'openpyxl.drawing.geometry.Transform2D'>

openpyxl.chart.stock_chart module

class openpyxl.chart.stock_chart.**StockChart**(*ser=()*, *dLbls=None*, *dropLines=None*, *hiLowLines=None*, *upDownBars=None*, *extLst=None*, ***kw*)

基类：openpyxl.chart._chart.ChartBase

dLbls

Values must be of type <class ‘openpyxl.chart.label.DataLabelList’ >

dataLabels

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

dropLines

Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

hiLowLines

Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

ser

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'stockChart'

upDownBars

Values must be of type <class ‘openpyxl.chart.updown_bars.UpDownBars’ >

x_axis

Values must be of type <class ‘openpyxl.chart.axis.TextAxis’ >

y_axis

Values must be of type <class ‘openpyxl.chart.axis.NumericAxis’ >

openpyxl.chart.surface_chart module

class openpyxl.chart.surface_chart.**BandFormat**(*idx=0*, *spPr=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

graphicalProperties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

idx

Values must be of type <class ‘int’ >

spPr

Values must be of type <class ‘openpyxl.chart.shapes.GraphicalProperties’ >

tagname = 'bandFmt'

class openpyxl.chart.surface_chart.**BandFormatList**(*bandFmt=()*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

bandFmt

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'bandFmts'

class openpyxl.chart.surface_chart.**SurfaceChart**(***kw*)

基类：*openpyxl.chart.surface_chart.SurfaceChart3D*

bandFmts

Values must be of type <class ‘openpyxl.chart.surface_chart.BandFormatList’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

ser

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'surfaceChart'

wireframe

Values must be of type <class ‘bool’ >

class openpyxl.chart.surface_chart.**SurfaceChart3D**(***kw*)

基类：openpyxl.chart.surface_chart._SurfaceChartBase, openpyxl.chart._3d._3DBase

bandFmts

Values must be of type <class ‘openpyxl.chart.surface_chart.BandFormatList’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

ser

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'surface3DChart'

wireframe

Values must be of type <class ‘bool’ >

x_axis

Values must be of type <class ‘openpyxl.chart.axis.TextAxis’ >

y_axis

Values must be of type <class ‘openpyxl.chart.axis.NumericAxis’ >

z_axis

Values must be of type <class ‘openpyxl.chart.axis.SeriesAxis’ >

openpyxl.chart.text module

class openpyxl.chart.text.**RichText**(*bodyPr=None*, *lstStyle=None*, *p=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

From the specification: 21.2.2.216

This element specifies text formatting. The lstStyle element is not supported.

bodyPr

Values must be of type <class ‘openpyxl.drawing.text.RichTextProperties’ >

lstStyle

Values must be of type <class ‘openpyxl.drawing.text.ListStyle’ >

p

A sequence (list or tuple) that may only contain objects of the declared type

paragraphs

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

properties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

tagname = 'rich'

class openpyxl.chart.text.**Text**(*strRef=None*, *rich=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

The value can be either a cell reference or a text element If both are present then the reference will be used.

rich

Values must be of type <class ‘openpyxl.chart.text.RichText’ >

strRef

Values must be of type <class ‘openpyxl.chart.data_source.StrRef’ >

tagname = 'tx'

to_tree(*tagname=None*, *idx=None*, *namespace=None*)

openpyxl.chart.title module

class openpyxl.chart.title.**Title**(*tx=None*, *layout=None*, *overlay=None*, *spPr=None*, *txPr=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

body

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

graphicalProperties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

layout

Values must be of type <class ‘openpyxl.chart.layout.Layout’ >

overlay

Values must be of type <class ‘bool’ >

spPr

Values must be of type <class ‘openpyxl.chart.shapes.GraphicalProperties’ >

tagname = 'title'

text

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

tx

Values must be of type <class ‘openpyxl.chart.text.Text’ >

txPr

Values must be of type <class ‘openpyxl.chart.text.RichText’ >

class openpyxl.chart.title.**TitleDescriptor**(**args*, ***kw*)

基类：*openpyxl.descriptors.base.Typed*

allow_none = True

expected_type

Title 的别名

openpyxl.chart.title.**title_maker**(*text*)

openpyxl.chart.trendline module

class openpyxl.chart.trendline.**Trendline**(*name=None*, *spPr=None*, *trendlineType='linear'*, *order=None*, *period=None*, *forward=None*, *backward=None*, *intercept=None*, *dispRSqr=None*, *dispEq=None*, *trendlineLbl=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`backward`

Values must be of type <class ‘float’ >

`dispEq`

Values must be of type <class ‘bool’ >

`dispRSqr`

Values must be of type <class ‘bool’ >

`extLst`

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

`forward`

Values must be of type <class ‘float’ >

`graphicalProperties`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

`intercept`

Values must be of type <class ‘float’ >

`name`

Values must be of type <class ‘str’ >

`order`

Values must be of type <class ‘int’ >

`period`

Values must be of type <class ‘int’ >

`spPr`

Values must be of type <class ‘openpyxl.chart.shapes.GraphicalProperties’ >

`tagname = 'trendline'`

`trendlineLbl`

Values must be of type <class ‘openpyxl.chart.trendline.TrendlineLabel’ >

`trendlineType`

Value must be one of { ‘log’ , ‘poly’ , ‘exp’ , ‘linear’ , ‘power’ , ‘movingAvg’ }

`class openpyxl.chart.trendline.`**`TrendlineLabel`**`(`*layout=None, tx=None, numFmt=None, spPr=None, txPr=None, extLst=None*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`extLst`

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

`graphicalProperties`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

`layout`

Values must be of type <class ‘openpyxl.chart.layout.Layout’ >

`numFmt`

Values must be of type <class ‘openpyxl.chart.data_source.NumFmt’ >

`spPr`

Values must be of type <class ‘openpyxl.chart.shapes.GraphicalProperties’ >

`tagname = 'trendlineLbl'`

`textProperties`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

`tx`

Values must be of type <class ‘openpyxl.chart.text.Text’ >

`txPr`

Values must be of type <class ‘openpyxl.chart.text.RichText’ >

openpyxl.chart.updown_bars module

`class openpyxl.chart.updown_bars.UpDownBars(`*gapWidth=150*, *upBars=None*, *downBars=None*, *extLst=None*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`downBars`

Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

`extLst`

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

`gapWidth`

Values must be of type <class ‘float’ >

`tagname = 'upbars'`

`upBars`

Values must be of type <class ‘openpyxl.chart.axis.ChartLines’ >

openpyxl.chartsheet package

Submodules

openpyxl.chartsheet.chartsheet module

class openpyxl.chartsheet.chartsheet.**Chartsheet**(*sheetPr=None, sheetViews=None, sheetProtection=None, customSheetViews=None, pageMargins=None, pageSetup=None, headerFooter=None, drawing=None, drawingHF=None, picture=None, webPublishItems=None, extLst=None, parent=None, title='', sheet_state='visible'*)

基类：openpyxl.workbook.child._WorkbookChild, *openpyxl.descriptors.serialisable.Serialisable*

HeaderFooter

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

add_chart(*chart*)

customSheetViews

Values must be of type <class 'openpyxl.chartsheet.custom.CustomChartsheetViews' >

drawing

Values must be of type <class 'openpyxl.worksheet.drawing.Drawing' >

drawingHF

Values must be of type <class 'openpyxl.chartsheet.relation.DrawingHF' >

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

headerFooter

Values must be of type <class 'openpyxl.worksheet.header_footer.HeaderFooter' >

mime_type = 'application/vnd.openxmlformats-officedocument.spreadsheetml.chartsheet+xml'

pageMargins

Values must be of type <class 'openpyxl.worksheet.page.PageMargins' >

pageSetup

Values must be of type <class 'openpyxl.worksheet.page.PrintPageSetup' >

picture

Values must be of type <class 'openpyxl.chartsheet.relation.SheetBackgroundPicture' >

sheetPr

Values must be of type <class 'openpyxl.chartsheet.properties.ChartsheetProperties' >

sheetProtection

Values must be of type <class 'openpyxl.chartsheet.protection.ChartsheetProtection' >

sheetViews

Values must be of type <class 'openpyxl.chartsheet.views.ChartsheetViewList'>

sheet_state

Value must be one of {'veryHidden', 'visible', 'hidden'}

tagname = 'chartsheet'

to_tree()

webPublishItems

Values must be of type <class 'openpyxl.chartsheet.publish.WebPublishItems'>

openpyxl.chartsheet.custom module

class openpyxl.chartsheet.custom.**CustomChartsheetView**(*guid=None*, *scale=None*, *state='visible'*, *zoomToFit=None*, *pageMargins=None*, *pageSetup=None*, *headerFooter=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

guid

headerFooter

Values must be of type <class 'openpyxl.worksheet.header_footer.HeaderFooter'>

pageMargins

Values must be of type <class 'openpyxl.worksheet.page.PageMargins'>

pageSetup

Values must be of type <class 'openpyxl.worksheet.page.PrintPageSetup'>

scale

Values must be of type <class 'int'>

state

Value must be one of {'veryHidden', 'visible', 'hidden'}

tagname = 'customSheetView'

zoomToFit

Values must be of type <class 'bool'>

class openpyxl.chartsheet.custom.**CustomChartsheetViews**(*customSheetView=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

customSheetView

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'customSheetViews'

openpyxl.chartsheet.properties module

class openpyxl.chartsheet.properties.**ChartsheetProperties**(*published=None, codeName=None, tabColor=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

codeName

Values must be of type <class 'str'>

published

Values must be of type <class 'bool'>

tabColor

Values must be of type <class 'openpyxl.styles.colors.Color'>

tagname = 'sheetPr'

openpyxl.chartsheet.protection module

class openpyxl.chartsheet.protection.**ChartsheetProtection**(*content=None, objects=None, hashValue=None, spinCount=None, saltValue=None, algorithmName=None, password=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*, openpyxl.worksheet.protection._Protected

algorithmName

Values must be of type <class 'str'>

content

Values must be of type <class 'bool'>

hashValue

objects

Values must be of type <class 'bool'>

saltValue

spinCount

Values must be of type <class 'int'>

tagname = 'sheetProtection'

openpyxl.chartsheet.publish module

`class openpyxl.chartsheet.publish.WebPublishItem`(*id=None*, *divId=None*, *sourceType=None*, *sourceRef=None*, *sourceObject=None*, *destinationFile=None*, *title=None*, *autoRepublish=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

autoRepublish

Values must be of type <class 'bool' >

destinationFile

Values must be of type <class 'str' >

divId

Values must be of type <class 'str' >

id

Values must be of type <class 'int' >

sourceObject

Values must be of type <class 'str' >

sourceRef

Values must be of type <class 'str' >

sourceType

Value must be one of { 'sheet' , 'chart' , 'printArea' , 'range' , 'query' , 'label' , 'autoFilter' , 'pivotTable' }

tagname = 'webPublishItem'

title

Values must be of type <class 'str' >

`class openpyxl.chartsheet.publish.WebPublishItems`(*count=None*, *webPublishItem=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

count

Values must be of type <class 'int' >

tagname = 'WebPublishItems'

webPublishItem

A sequence (list or tuple) that may only contain objects of the declared type

openpyxl.chartsheet.relation module

class openpyxl.chartsheet.relation.**DrawingHF**(*id=None*, *lho=None*, *lhe=None*, *lhf=None*, *cho=None*, *che=None*, *chf=None*, *rho=None*, *rhe=None*, *rhf=None*, *lfo=None*, *lfe=None*, *lff=None*, *cfo=None*, *cfe=None*, *cff=None*, *rfo=None*, *rfe=None*, *rff=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

centerFooterEvenPages

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

centerFooterFirstPage

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

centerFooterOddPages

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

centerHeaderEvenPages

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

centerHeaderFirstPage

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

centerHeaderOddPages

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

cfe

Values must be of type <class ‘int’ >

cff

Values must be of type <class ‘int’ >

cfo

Values must be of type <class ‘int’ >

che

Values must be of type <class ‘int’ >

chf

Values must be of type <class ‘int’ >

cho
Values must be of type <class 'int'>

id
Values must be of type <class 'str'>

leftFooterEvenPages
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

leftFooterFirstPage
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

leftFooterOddPages
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

leftHeaderEvenPages
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

leftHeaderFirstPage
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

leftHeaderOddPages
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

lfe
Values must be of type <class 'int'>

lff
Values must be of type <class 'int'>

lfo
Values must be of type <class 'int'>

lhe
Values must be of type <class 'int'>

lhf
Values must be of type <class 'int'>

lho
Values must be of type <class 'int'>

rfe
Values must be of type <class 'int'>

rff
Values must be of type <class ‘int’ >

rfo
Values must be of type <class ‘int’ >

rhe
Values must be of type <class ‘int’ >

rhf
Values must be of type <class ‘int’ >

rho
Values must be of type <class ‘int’ >

rightFooterEvenPages
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

rightFooterFirstPage
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

rightFooterOddPages
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

rightHeaderEvenPages
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

rightHeaderFirstPage
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

rightHeaderOddPages
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

class openpyxl.chartsheet.relation.**SheetBackgroundPicture**(*id*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

id
Values must be of type <class ‘str’ >

tagname = 'picture'

openpyxl.chartsheet.views module

class openpyxl.chartsheet.views.**ChartsheetView**(*tabSelected=None*, *zoomScale=None*, *workbookViewId=0*, *zoomToFit=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

tabSelected

Values must be of type <class 'bool' >

tagname = 'sheetView'

workbookViewId

Values must be of type <class 'int' >

zoomScale

Values must be of type <class 'int' >

zoomToFit

Values must be of type <class 'bool' >

class openpyxl.chartsheet.views.**ChartsheetViewList**(*sheetView=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

sheetView

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'sheetViews'

openpyxl.comments package

Submodules

openpyxl.comments.author module

class openpyxl.comments.author.**AuthorList**(*author=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

author

A sequence (list or tuple) that may only contain objects of the declared type

authors

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

tagname = 'authors'

openpyxl.comments.comment_sheet module

class openpyxl.comments.comment_sheet.**CommentRecord**(*ref=", authorId=0, guid=None, shapeId=0, text=None, commentPr=None, author=None, height=79, width=144*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

author

Values must be of type <class 'str' >

authorId

Values must be of type <class 'int' >

commentPr

Values must be of type <class 'openpyxl.comments.comment_sheet.Properties' >

content

Remove all inline formatting and stuff

classmethod **from_cell**(*cell*)

Class method to convert cell comment

guid

ref

Values must be of type <class 'str' >

shapeId

Values must be of type <class 'int' >

tagname = 'comment'

text

Values must be of type <class 'openpyxl.cell.text.Text' >

class openpyxl.comments.comment_sheet.**CommentSheet**(*authors=None, commentList=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

authors

Values must be of type <class 'openpyxl.comments.author.AuthorList' >

commentList

Wrap a sequence in an containing object

comments

Return a dictionary of comments keyed by coord

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

classmethod **from_comments**(*comments*)

Create a comment sheet from a list of comments for a particular worksheet

mime_type = 'application/vnd.openxmlformats-officedocument.spreadsheetml.comments+xml'

path

Return path within the archive

tagname = 'comments'

to_tree()

write_shapes(*vml=None*)

Create the VML for comments

class openpyxl.comments.comment_sheet.**Properties**(*locked=None*, *defaultSize=None*, *_print=None*, *disabled=None*, *uiObject=None*, *autoFill=None*, *autoLine=None*, *altText=None*, *textHAlign=None*, *textVAlign=None*, *lockText=None*, *justLastX=None*, *autoScale=None*, *rowHidden=None*, *colHidden=None*, *anchor=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

altText

Values must be of type <class 'str'>

autoFill

Values must be of type <class 'bool'>

autoLine

Values must be of type <class 'bool'>

autoScale

Values must be of type <class 'bool'>

colHidden

Values must be of type <class 'bool'>

defaultSize

Values must be of type <class 'bool'>

disabled

Values must be of type <class 'bool'>

justLastX
Values must be of type <class 'bool'>

lockText
Values must be of type <class 'bool'>

locked
Values must be of type <class 'bool'>

rowHidden
Values must be of type <class 'bool'>

textHAlign
Value must be one of {'left', 'justify', 'right', 'distributed', 'center'}

textVAlign
Value must be one of {'top', 'bottom', 'justify', 'distributed', 'center'}

uiObject
Values must be of type <class 'bool'>

openpyxl.comments.comment_sheet.**tostring**(*element*, ***, *encoding='utf-8'*, *method=None*, *short_empty_elements=True*)

Generate string representation of XML element.

All subelements are included. If encoding is "unicode", a string is returned. Otherwise a bytestring is returned.

element is an Element instance, *encoding* is an optional output encoding defaulting to US-ASCII, *method* is an optional output which can be one of "xml" (default), "html", "text" or "c14n".

Returns an (optionally) encoded string containing the XML data.

openpyxl.comments.comments module

class openpyxl.comments.comments.**Comment**(*text*, *author*, *height=79*, *width=144*)
基类：object

bind(*cell*)
Bind comment to a particular cell

parent

text
Any comment text stripped of all formatting.

unbind()
Unbind a comment from a cell

openpyxl.comments.shape_writer module

`class openpyxl.comments.shape_writer.`**`ShapeWriter`**`(`*comments*`)`

基类：`object`

Create VML for comments

`add_comment_shape``(`*root, idx, coord, height, width*`)`

`add_comment_shapetype``(`*root*`)`

`vml = None`

`vml_path = None`

`write``(`*root*`)`

`openpyxl.comments.shape_writer.`**`tostring`**`(`*element, *, encoding='utf-8', method=None, short_empty_elements=True*`)`

Generate string representation of XML element.

All subelements are included. If encoding is "unicode", a string is returned. Otherwise a bytestring is returned.

element is an Element instance, *encoding* is an optional output encoding defaulting to US-ASCII, *method* is an optional output which can be one of "xml" (default), "html", "text" or "c14n".

Returns an (optionally) encoded string containing the XML data.

openpyxl.compat package

`class openpyxl.compat.`**`DummyCode`**

基类：`object`

`openpyxl.compat.`**`deprecated`**`(`*reason*`)`

Submodules

openpyxl.compat.abc module

openpyxl.compat.numbers module

openpyxl.compat.product module

math.prod equivalent for < Python 3.8

`openpyxl.compat.product.`**`prod`**`(`*sequence*`)`

`openpyxl.compat.product.`**`product`**`(`*sequence*`)`

openpyxl.compat.singleton module

class openpyxl.compat.singleton.**Cached**(**args*, ***kw*)

基类： type

Caching metaclass Child classes will only create new instances of themselves if one doesn’t already exist. Does not work with __slots__

class openpyxl.compat.singleton.**Singleton**(**args*, ***kw*)

基类： type

Singleton metaclass Based on Python Cookbook 3rd Edition Recipe 9.13 Only one instance of a class can exist. Does not work with __slots__

openpyxl.compat.strings module

openpyxl.compat.strings.**safe_string**(*value*)

Safely and consistently format numeric values

openpyxl.descriptors package

class openpyxl.descriptors.**MetaSerialisable**

基类： type

class openpyxl.descriptors.**MetaStrict**

基类： type

class openpyxl.descriptors.**Strict**

基类： object

Submodules

openpyxl.descriptors.base module

Based on Python Cookbook 3rd Edition, 8.13 http://chimera.labs.oreilly.com/books/1230000000393/ch08.html#_discussiuncion_130

class openpyxl.descriptors.base.**ASCII**(**args*, ***kw*)

基类： *openpyxl.descriptors.base.Typed*

expected_type

builtins.bytes 的别名

class openpyxl.descriptors.base.**Alias**(*alias*)

基类： *openpyxl.descriptors.base.Descriptor*

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

class openpyxl.descriptors.base.**Bool**(**args*, ***kw*)

基类：*openpyxl.descriptors.base.Convertible*

expected_type

builtins.bool 的别名

class openpyxl.descriptors.base.**Convertible**(**args*, ***kw*)

基类：*openpyxl.descriptors.base.Typed*

Values must be convertible to a particular type

class openpyxl.descriptors.base.**DateTime**(**args*, ***kw*)

基类：*openpyxl.descriptors.base.Typed*

expected_type

datetime.datetime 的别名

class openpyxl.descriptors.base.**Default**(*name=None*, ***kw*)

基类：*openpyxl.descriptors.base.Typed*

When called returns an instance of the expected type. Additional default values can be passed in to the descriptor

class openpyxl.descriptors.base.**Descriptor**(*name=None*, ***kw*)

基类：object

class openpyxl.descriptors.base.**Float**(**args*, ***kw*)

基类：*openpyxl.descriptors.base.Convertible*

expected_type

builtins.float 的别名

class openpyxl.descriptors.base.**Integer**(**args*, ***kw*)

基类：*openpyxl.descriptors.base.Convertible*

expected_type

builtins.int 的别名

class openpyxl.descriptors.base.**Length**(*name=None*, ***kw*)

基类：*openpyxl.descriptors.base.Descriptor*

class openpyxl.descriptors.base.**MatchPattern**(*name=None*, ***kw*)

基类：*openpyxl.descriptors.base.Descriptor*

Values must match a regex pattern

allow_none = False

class openpyxl.descriptors.base.**Max**(***kw*)

基类：*openpyxl.descriptors.base.Convertible*

Values must be less than a *max* value

allow_none = False

expected_type

builtins.float 的别名

class openpyxl.descriptors.base.Min(***kw*)

基类：*openpyxl.descriptors.base.Convertible*

Values must be greater than a *min* value

allow_none = False

expected_type

builtins.float 的别名

class openpyxl.descriptors.base.MinMax(***kw*)

基类：*openpyxl.descriptors.base.Min*, *openpyxl.descriptors.base.Max*

Values must be greater than *min* value and less than a *max* one

class openpyxl.descriptors.base.NoneSet(*name=None*, ***kw*)

基类：*openpyxl.descriptors.base.Set*

‘none’ will be treated as None

class openpyxl.descriptors.base.Set(*name=None*, ***kw*)

基类：*openpyxl.descriptors.base.Descriptor*

Value can only be from a set of know values

class openpyxl.descriptors.base.String(**args*, ***kw*)

基类：*openpyxl.descriptors.base.Typed*

expected_type

builtins.str 的别名

class openpyxl.descriptors.base.Text(**args*, ***kw*)

基类：*openpyxl.descriptors.base.String*, *openpyxl.descriptors.base.Convertible*

class openpyxl.descriptors.base.Tuple(**args*, ***kw*)

基类：*openpyxl.descriptors.base.Typed*

expected_type

builtins.tuple 的别名

class openpyxl.descriptors.base.Typed(**args*, ***kw*)

基类：*openpyxl.descriptors.base.Descriptor*

Values must of a particular type

allow_none = False

`expected_type`

  `builtins.NoneType` 的别名

`nested = False`

openpyxl.descriptors.excel module

Excel specific descriptors

`class openpyxl.descriptors.excel.`**`Base64Binary`**`(`*name=None*, ***kw*`)`

  基类：*openpyxl.descriptors.base.MatchPattern*

  `pattern = '^(?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=|[A-Za-z0-9+/]{4})$'`

`class openpyxl.descriptors.excel.`**`CellRange`**`(`*name=None*, ***kw*`)`

  基类：*openpyxl.descriptors.base.MatchPattern*

  `allow_none = True`

  `pattern = '^[$]?([A-Za-z]{1,3})[$]?(\\d+)(:[$]?([A-Za-z]{1,3})[$]?(\\d+)?)?$|^[A-Za-z]{1,3}:[A-Za-z`

`class openpyxl.descriptors.excel.`**`Extension`**`(`*uri=None*`)`

  基类：*openpyxl.descriptors.serialisable.Serialisable*

  `uri`

    Values must be of type <class 'str'>

`class openpyxl.descriptors.excel.`**`ExtensionList`**`(`*ext=()*`)`

  基类：*openpyxl.descriptors.serialisable.Serialisable*

  `ext`

    A sequence (list or tuple) that may only contain objects of the declared type

`class openpyxl.descriptors.excel.`**`Guid`**`(`*name=None*, ***kw*`)`

  基类：*openpyxl.descriptors.base.MatchPattern*

  `pattern = '{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\\}'`

`class openpyxl.descriptors.excel.`**`HexBinary`**`(`*name=None*, ***kw*`)`

  基类：*openpyxl.descriptors.base.MatchPattern*

  `pattern = '[0-9a-fA-F]+$'`

`class openpyxl.descriptors.excel.`**`Percentage`**`(`***kw*`)`

  基类：*openpyxl.descriptors.base.MinMax*

  `max = 1000000`

  `min = -1000000`

  `pattern = '((100)|([0-9][0-9]?))(\\.[0-9][0-9]?)?%'`

class openpyxl.descriptors.excel.**Relation**(**args*, ***kw*)

基类：*openpyxl.descriptors.base.String*

allow_none = True

namespace = 'http://schemas.openxmlformats.org/officeDocument/2006/relationships'

class openpyxl.descriptors.excel.**TextPoint**(***kw*)

基类：*openpyxl.descriptors.base.MinMax*

Size in hundredths of points. In theory other units of measurement can be used but these are unbounded

expected_type

builtins.int 的别名

max = 400000

min = -400000

class openpyxl.descriptors.excel.**UniversalMeasure**(*name=None*, ***kw*)

基类：*openpyxl.descriptors.base.MatchPattern*

pattern = '[0-9]+(\\.[0-9]+)?(mm|cm|in|pt|pc|pi)'

openpyxl.descriptors.namespace module

openpyxl.descriptors.namespace.**namespaced**(*obj*, *tagname*, *namespace=None*)

Utility to create a namespaced tag for an object

openpyxl.descriptors.nested module

Generic serialisable classes

class openpyxl.descriptors.nested.**EmptyTag**(**args*, ***kw*)

基类：*openpyxl.descriptors.nested.Nested*, *openpyxl.descriptors.base.Bool*

Boolean if a tag exists or not.

from_tree(*node*)

to_tree(*tagname=None*, *value=None*, *namespace=None*)

class openpyxl.descriptors.nested.**Nested**(*name=None*, ***kw*)

基类：*openpyxl.descriptors.base.Descriptor*

attribute = 'val'

from_tree(*node*)

nested = True

to_tree(*tagname=None*, *value=None*, *namespace=None*)

class openpyxl.descriptors.nested.**NestedBool**(**args*, ***kw*)

基类：*openpyxl.descriptors.nested.NestedValue*, *openpyxl.descriptors.base.Bool*

from_tree(*node*)

class openpyxl.descriptors.nested.**NestedFloat**(**args*, ***kw*)

基类：*openpyxl.descriptors.nested.NestedValue*, *openpyxl.descriptors.base.Float*

class openpyxl.descriptors.nested.**NestedInteger**(**args*, ***kw*)

基类：*openpyxl.descriptors.nested.NestedValue*, *openpyxl.descriptors.base.Integer*

class openpyxl.descriptors.nested.**NestedMinMax**(***kw*)

基类：*openpyxl.descriptors.nested.Nested*, *openpyxl.descriptors.base.MinMax*

class openpyxl.descriptors.nested.**NestedNoneSet**(*name=None*, ***kw*)

基类：*openpyxl.descriptors.nested.Nested*, *openpyxl.descriptors.base.NoneSet*

class openpyxl.descriptors.nested.**NestedSet**(*name=None*, ***kw*)

基类：*openpyxl.descriptors.nested.Nested*, *openpyxl.descriptors.base.Set*

class openpyxl.descriptors.nested.**NestedString**(**args*, ***kw*)

基类：*openpyxl.descriptors.nested.NestedValue*, *openpyxl.descriptors.base.String*

class openpyxl.descriptors.nested.**NestedText**(**args*, ***kw*)

基类：*openpyxl.descriptors.nested.NestedValue*

Represents any nested tag with the value as the contents of the tag

from_tree(*node*)

to_tree(*tagname=None*, *value=None*, *namespace=None*)

class openpyxl.descriptors.nested.**NestedValue**(**args*, ***kw*)

基类：*openpyxl.descriptors.nested.Nested*, *openpyxl.descriptors.base.Convertible*

Nested tag storing the value on the ‘val’ attribute

openpyxl.descriptors.sequence module

class openpyxl.descriptors.sequence.**MultiSequence**(*name=None*, ***kw*)

基类：*openpyxl.descriptors.sequence.Sequence*

Sequences can contain objects with different tags

to_tree(*tagname*, *obj*, *namespace=None*)

Convert the sequence represented by the descriptor to an XML element

class openpyxl.descriptors.sequence.**MultiSequencePart**(*expected_type*, *store*)

基类：*openpyxl.descriptors.base.Alias*

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

class openpyxl.descriptors.sequence.**NestedSequence**(*name=None*, ***kw*)

基类：*openpyxl.descriptors.sequence.Sequence*

Wrap a sequence in an containing object

count = False

from_tree(*node*)

to_tree(*tagname*, *obj*, *namespace=None*)

Convert the sequence represented by the descriptor to an XML element

class openpyxl.descriptors.sequence.**Sequence**(*name=None*, ***kw*)

基类：*openpyxl.descriptors.base.Descriptor*

A sequence (list or tuple) that may only contain objects of the declared type

expected_type

builtins.NoneType 的别名

idx_base = 0

seq_types = (<class 'list'>, <class 'tuple'>)

to_tree(*tagname*, *obj*, *namespace=None*)

Convert the sequence represented by the descriptor to an XML element

unique = False

class openpyxl.descriptors.sequence.**ValueSequence**(*name=None*, ***kw*)

基类：*openpyxl.descriptors.sequence.Sequence*

A sequence of primitive types that are stored as a single attribute. “val” is the default attribute

attribute = 'val'

from_tree(*node*)

to_tree(*tagname*, *obj*, *namespace=None*)

Convert the sequence represented by the descriptor to an XML element

openpyxl.descriptors.serialisable module

class openpyxl.descriptors.serialisable.**Serialisable**

基类：openpyxl.descriptors._Serialisable

Objects can serialise to XML their attributes and child objects. The following class attributes are created by the metaclass at runtime: __attrs__ = attributes __nested__ = single-valued child treated as an attribute __elements__ = child elements

classmethod from_tree(*node*)

Create object from XML

idx_base = 0

namespace = None

tagname

to_tree(*tagname=None, idx=None, namespace=None*)

openpyxl.descriptors.slots module

class openpyxl.descriptors.slots.AutoSlotProperties

基类：type

openpyxl.drawing package

Submodules

openpyxl.drawing.colors module

class openpyxl.drawing.colors.ColorChoice(*scrgbClr=None, srgbClr=None, hslClr=None, sysClr=None, schemeClr=None, prstClr=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

RGB

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

RGBPercent

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

hslClr

Values must be of type <class 'openpyxl.drawing.colors.HSLColor' >

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

prstClr

Value must be one of { 'darkSlateBlue' , 'mediumBlue' , 'sandyBrown' , 'medAquamarine' , 'ltGray' , 'deepPink' , 'cornsilk' , 'cyan' , 'dkSalmon' , 'darkSalmon' , 'ltSlateGrey' , 'linen' , 'steelBlue' , 'lightSeaGreen' , 'dkSeaGreen' , 'darkSlateGray' , 'medSeaGreen' , 'hotPink' , 'dkMagenta' , 'darkGrey' , 'rosyBrown' , 'white' , 'dkViolet' , 'ltSkyBlue' , 'fuchsia' , 'mediumVioletRed' , 'snow' , 'darkGoldenrod' , 'tan' , 'lightSteelBlue' , 'cornflowerBlue' , 'cadetBlue' , 'lightYellow' , 'orangeRed' , 'dkGreen' , 'paleGreen' , 'sienna' , 'lemonChiffon' , 'darkSeaGreen' , 'blueViolet' , 'crimson' , 'khaki' , 'ltYellow' , ‘dkOrchid’, ‘gainsboro’, ‘ivory’, ‘mediumAquamarine’, ‘dkKhaki’, ‘ltCoral’, ‘azure’, ‘indianRed’, ‘darkOliveGreen’, ‘aliceBlue’, ‘dkBlue’, ‘mediumPurple’, ‘dkSlateGray’, ‘dodgerBlue’, ‘teal’, ‘medPurple’, ‘lightSalmon’, ‘springGreen’, ‘darkKhaki’, ‘dkRed’, ‘chocolate’, ‘darkGreen’, ‘ltCyan’, ‘moccasin’, ‘paleGoldenrod’, ‘mediumTurquoise’, ‘deepSkyBlue’, ‘grey’, ‘green’, ‘peachPuff’, ‘plum’, ‘blue’, ‘firebrick’, ‘ltSalmon’, ‘lightGreen’, ‘darkRed’, ‘royalBlue’, ‘saddleBrown’, ‘medVioletRed’, ‘darkOrchid’, ‘floralWhite’, ‘lightPink’, ‘lightSkyBlue’, ‘ltGrey’, ‘aquamarine’, ‘medSpringGreen’, ‘seaShell’, ‘thistle’, ‘lightBlue’, ‘darkMagenta’, ‘medOrchid’, ‘ltSeaGreen’, ‘magenta’, ‘turquoise’, ‘dkTurquoise’, ‘wheat’, ‘dkOrange’, ‘lawnGreen’, ‘oliveDrab’, ‘skyBlue’, ‘lightGoldenrodYellow’, ‘whiteSmoke’, ‘ltGoldenrodYellow’, ‘chartreuse’, ‘lightCoral’, ‘violet’, ‘silver’, ‘black’, ‘dimGray’, ‘lavender’, ‘medSlateBlue’, ‘salmon’, ‘navajoWhite’, ‘gray’, ‘dkOliveGreen’, ‘dkSlateGrey’, ‘lightSlateGrey’, ‘yellowGreen’, ‘darkSlateGrey’, ‘gold’, ‘mintCream’, ‘paleVioletRed’, ‘peru’, ‘ltBlue’, ‘ltSteelBlue’, ‘dkGray’, ‘seaGreen’, ‘orchid’, ‘lightSlateGray’, ‘ltPink’, ‘greenYellow’, ‘midnightBlue’, ‘red’, ‘dkCyan’, ‘lime’, ‘orange’, ‘darkTurquoise’, ‘forestGreen’, ‘honeydew’, ‘pink’, ‘mediumSlateBlue’, ‘powderBlue’, ‘darkBlue’, ‘goldenrod’, ‘yellow’, ‘mediumSpringGreen’, ‘medTurquoise’, ‘paleTurquoise’, ‘ltGreen’, ‘slateBlue’, ‘darkOrange’, ‘lightGrey’, ‘dkGrey’, ‘darkCyan’, ‘burlyWood’, ‘slateGray’, ‘mistyRose’, ‘dkGoldenrod’, ‘slateGrey’, ‘medBlue’, ‘lavenderBlush’, ‘dimGrey’, ‘coral’, ‘blanchedAlmond’, ‘darkViolet’, ‘aqua’, ‘limeGreen’, ‘indigo’, ‘lightGray’, ‘lightCyan’, ‘purple’, ‘antiqueWhite’, ‘tomato’, ‘bisque’, ‘oldLace’, ‘beige’, ‘papayaWhip’, ‘maroon’, ‘mediumOrchid’, ‘ghostWhite’, ‘dkSlateBlue’, ‘ltSlateGray’, ‘mediumSeaGreen’, ‘brown’, ‘navy’, ‘olive’, ‘darkGray’ }

`schemeClr`

Values must be of type <class ‘openpyxl.drawing.colors.SchemeColor’ >

`scrgbClr`

Values must be of type <class ‘openpyxl.drawing.colors.RGBPercent’ >

`srgbClr`

Values must be of type <class ‘str’ >

`sysClr`

Values must be of type <class ‘openpyxl.drawing.colors.SystemColor’ >

`tagname = 'colorChoice'`

`class openpyxl.drawing.colors.ColorChoiceDescriptor(*args, **kw)`

基类：*openpyxl.descriptors.base.Typed*

Objects can choose from 7 different kinds of color system. Assume RGBHex if a string is passed in.

`allow_none = True`

`expected_type`

ColorChoice 的别名

class openpyxl.drawing.colors.ColorMapping(*bg1='lt1', tx1='dk1', bg2='lt2', tx2='dk2', accent1='accent1', accent2='accent2', accent3='accent3', accent4='accent4', accent5='accent5', accent6='accent6', hlink='hlink', folHlink='folHlink', extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

accent1

Value must be one of { 'accent5' , 'folHlink' , 'hlink' , 'accent1' , 'dk2' , 'accent2' , 'lt2' , 'accent6' , 'accent3' , 'lt1' , 'accent4' , 'dk1' }

accent2

Value must be one of { 'accent5' , 'folHlink' , 'hlink' , 'accent1' , 'dk2' , 'accent2' , 'lt2' , 'accent6' , 'accent3' , 'lt1' , 'accent4' , 'dk1' }

accent3

Value must be one of { 'accent5' , 'folHlink' , 'hlink' , 'accent1' , 'dk2' , 'accent2' , 'lt2' , 'accent6' , 'accent3' , 'lt1' , 'accent4' , 'dk1' }

accent4

Value must be one of { 'accent5' , 'folHlink' , 'hlink' , 'accent1' , 'dk2' , 'accent2' , 'lt2' , 'accent6' , 'accent3' , 'lt1' , 'accent4' , 'dk1' }

accent5

Value must be one of { 'accent5' , 'folHlink' , 'hlink' , 'accent1' , 'dk2' , 'accent2' , 'lt2' , 'accent6' , 'accent3' , 'lt1' , 'accent4' , 'dk1' }

accent6

Value must be one of { 'accent5' , 'folHlink' , 'hlink' , 'accent1' , 'dk2' , 'accent2' , 'lt2' , 'accent6' , 'accent3' , 'lt1' , 'accent4' , 'dk1' }

bg1

Value must be one of { 'accent5' , 'folHlink' , 'hlink' , 'accent1' , 'dk2' , 'accent2' , 'lt2' , 'accent6' , 'accent3' , 'lt1' , 'accent4' , 'dk1' }

bg2

Value must be one of { 'accent5' , 'folHlink' , 'hlink' , 'accent1' , 'dk2' , 'accent2' , 'lt2' , 'accent6' , 'accent3' , 'lt1' , 'accent4' , 'dk1' }

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

folHlink

Value must be one of { 'accent5' , 'folHlink' , 'hlink' , 'accent1' , 'dk2' , 'accent2' , 'lt2' , 'accent6' , 'accent3' , 'lt1' , 'accent4' , 'dk1' }

hlink

Value must be one of { 'accent5' , 'folHlink' , 'hlink' , 'accent1' , 'dk2' , 'accent2' , 'lt2' , 'accent6' , 'accent3' , 'lt1' , 'accent4' , 'dk1' }

tagname = 'clrMapOvr'

tx1

Value must be one of { 'accent5' , 'folHlink' , 'hlink' , 'accent1' , 'dk2' , 'accent2' , 'lt2' , 'accent6' , 'accent3' , 'lt1' , 'accent4' , 'dk1' }

tx2

Value must be one of { 'accent5' , 'folHlink' , 'hlink' , 'accent1' , 'dk2' , 'accent2' , 'lt2' , 'accent6' , 'accent3' , 'lt1' , 'accent4' , 'dk1' }

class openpyxl.drawing.colors.**HSLColor**(*hue=None*, *sat=None*, *lum=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

hue

Values must be of type <class 'int' >

lum

Values must be of type <class 'float' >

sat

Values must be of type <class 'float' >

tagname = 'hslClr'

class openpyxl.drawing.colors.**RGBPercent**(*r=None*, *g=None*, *b=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

b

Values must be of type <class 'float' >

g

Values must be of type <class 'float' >

r

Values must be of type <class 'float' >

tagname = 'rgbClr'

class openpyxl.drawing.colors.**SchemeColor**(*tint=None*, *shade=None*, *comp=None*, *inv=None*, *gray=None*, *alpha=None*, *alphaOff=None*, *alphaMod=None*, *hue=None*, *hueOff=None*, *hueMod=None*, *sat=None*, *satOff=None*, *satMod=None*, *lum=None*, *lumOff=None*, *lumMod=None*, *red=None*, *redOff=None*, *redMod=None*, *green=None*, *greenOff=None*, *greenMod=None*, *blue=None*, *blueOff=None*, *blueMod=None*, *gamma=None*, *invGamma=None*, *val=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

alpha
Values must be of type <class 'int' >

alphaMod
Values must be of type <class 'int' >

alphaOff
Values must be of type <class 'int' >

blue
Values must be of type <class 'int' >

blueMod
Values must be of type <class 'int' >

blueOff
Values must be of type <class 'int' >

comp
Values must be of type <class 'bool' >

gamma
Values must be of type <class 'bool' >

gray
Values must be of type <class 'int' >

green
Values must be of type <class 'int' >

greenMod
Values must be of type <class 'int' >

greenOff
Values must be of type <class 'int' >

hue
Values must be of type <class 'int' >

hueMod
Values must be of type <class 'int' >

hueOff
Values must be of type <class 'int' >

inv
Values must be of type <class 'int' >

invGamma
Values must be of type <class 'bool' >

lum

Values must be of type <class ‘int’ >

lumMod

Values must be of type <class ‘int’ >

lumOff

Values must be of type <class ‘int’ >

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

red

Values must be of type <class ‘int’ >

redMod

Values must be of type <class ‘int’ >

redOff

Values must be of type <class ‘int’ >

sat

Values must be of type <class ‘int’ >

satMod

Values must be of type <class ‘int’ >

satOff

Values must be of type <class ‘int’ >

shade

Values must be of type <class ‘int’ >

tagname = 'schemeClr'

tint

Values must be of type <class ‘int’ >

val

Value must be one of { ‘phClr’ , ‘accent5’ , ‘folHlink’ , ‘bg1’ , ‘hlink’ , ‘accent1’ , ‘accent2’ , ‘dk2’ , ‘tx2’ , ‘accent3’ , ‘lt2’ , ‘bg2’ , ‘tx1’ , ‘lt1’ , ‘accent6’ , ‘accent4’ , ‘dk1’ } `class openpyxl.drawing.colors.SystemColor`(*val='windowText', lastClr=None, tint=None, shade=None, comp=None, inv=None, gray=None, alpha=None, alphaOff=None, alphaMod=None, hue=None, hueOff=None, hueMod=None, sat=None, satOff=None, satMod=None, lum=None, lumOff=None, lumMod=None, red=None, redOff=None, redMod=None, green=None, greenOff=None, greenMod=None, blue=None, blueOff=None, blueMod=None, gamma=None, invGamma=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

alpha

Values must be of type <class 'int' >

alphaMod

Values must be of type <class 'int' >

alphaOff

Values must be of type <class 'int' >

blue

Values must be of type <class 'int' >

blueMod

Values must be of type <class 'int' >

blueOff

Values must be of type <class 'int' >

comp

Values must be of type <class 'openpyxl.drawing.colors.Transform' >

gamma

Values must be of type <class 'openpyxl.drawing.colors.Transform' >

gray

Values must be of type <class 'openpyxl.drawing.colors.Transform' >

green

Values must be of type <class 'int' >

greenMod

Values must be of type <class 'int' >

greenOff

Values must be of type <class 'int' >

hue

Values must be of type <class 'int' >

`hueMod`
: Values must be of type <class 'int'>

`hueOff`
: Values must be of type <class 'int'>

`inv`
: Values must be of type <class 'openpyxl.drawing.colors.Transform'>

`invGamma`
: Values must be of type <class 'openpyxl.drawing.colors.Transform'>

`lastClr`
: Values must be of type <class 'str'>

`lum`
: Values must be of type <class 'int'>

`lumMod`
: Values must be of type <class 'int'>

`lumOff`
: Values must be of type <class 'int'>

`namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'`

`red`
: Values must be of type <class 'int'>

`redMod`
: Values must be of type <class 'int'>

`redOff`
: Values must be of type <class 'int'>

`sat`
: Values must be of type <class 'int'>

`satMod`
: Values must be of type <class 'int'>

`satOff`
: Values must be of type <class 'int'>

`shade`
: Values must be of type <class 'int'>

`tagname = 'sysClr'`

`tint`
: Values must be of type <class 'int'>

val

Value must be one of { ‘menu’ , ‘window’ , ‘inactiveCaption’ , ‘inactiveCaptionText’ , ‘menuBar’ , ‘menuText’ , ‘gradientActiveCaption’ , ‘inactiveBorder’ , ‘highlightText’ , ‘btnHighlight’ , ‘activeCaption’ , ‘btnText’ , ‘background’ , ‘hotLight’ , ‘activeBorder’ , ‘btnShadow’ , ‘3dLight’ , ‘windowFrame’ , ‘btnFace’ , ‘gradientInactiveCaption’ , ‘appWorkspace’ , ‘menuHighlight’ , ‘3dDkShadow’ , ‘infoText’ , ‘highlight’ , ‘grayText’ , ‘windowText’ , ‘captionText’ , ‘scrollBar’ , ‘infoBk’ }

class openpyxl.drawing.colors.**Transform**

基类：*openpyxl.descriptors.serialisable.Serialisable*

openpyxl.drawing.connector module

class openpyxl.drawing.connector.**Connection**(*id=None*, *idx=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

id

Values must be of type <class ‘int’ >

idx

Values must be of type <class ‘int’ >

class openpyxl.drawing.connector.**ConnectorLocking**(*extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

class openpyxl.drawing.connector.**ConnectorNonVisual**(*cNvPr=None*, *cNvCxnSpPr=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

cNvCxnSpPr

Values must be of type <class ‘openpyxl.drawing.connector.NonVisualConnectorProperties’ >

cNvPr

Values must be of type <class ‘openpyxl.drawing.properties.NonVisualDrawingProps’ >

class openpyxl.drawing.connector.**ConnectorShape**(*nvCxnSpPr=None*, *spPr=None*, *style=None*, *macro=None*, *fPublished=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

fPublished

Values must be of type <class ‘bool’ >

macro

Values must be of type <class ‘str’ >

nvCxnSpPr
Values must be of type <class 'openpyxl.drawing.connector.ConnectorNonVisual'>

spPr
Values must be of type <class 'openpyxl.chart.shapes.GraphicalProperties'>

style
Values must be of type <class 'openpyxl.drawing.geometry.ShapeStyle'>

tagname = 'cxnSp'

class openpyxl.drawing.connector.**NonVisualConnectorProperties**(*cxnSpLocks=None, stCxn=None, endCxn=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

cxnSpLocks
Values must be of type <class 'openpyxl.drawing.connector.ConnectorLocking'>

endCxn
Values must be of type <class 'openpyxl.drawing.connector.Connection'>

extLst
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

stCxn
Values must be of type <class 'openpyxl.drawing.connector.Connection'>

class openpyxl.drawing.connector.**Shape**(*macro=None, textlink=None, fPublished=None, fLocksText=None, nvSpPr=None, spPr=None, style=None, txBody=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

fLocksText
Values must be of type <class 'bool'>

fPublished
Values must be of type <class 'bool'>

graphicalProperties
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

macro
Values must be of type <class 'str'>

meta
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

nvSpPr

Values must be of type <class 'openpyxl.drawing.connector.ShapeMeta' >

spPr

Values must be of type <class 'openpyxl.chart.shapes.GraphicalProperties' >

style

Values must be of type <class 'openpyxl.drawing.geometry.ShapeStyle' >

textlink

Values must be of type <class 'str' >

txBody

Values must be of type <class 'openpyxl.chart.text.RichText' >

class openpyxl.drawing.connector.**ShapeMeta**(*cNvPr=None*, *cNvSpPr=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

cNvPr

Values must be of type <class 'openpyxl.drawing.properties.NonVisualDrawingProps' >

cNvSpPr

Values must be of type <class 'openpyxl.drawing.properties.NonVisualDrawingShapeProps' >

tagname = 'nvSpPr'

openpyxl.drawing.drawing module

class openpyxl.drawing.drawing.**Drawing**

基类: object

a drawing object - eg container for shapes or charts we assume user specifies dimensions in pixels; units are converted to EMU in the drawing part

anchor

count = 0

get_emu_dimensions()

return (x, y, w, h) in EMU

注解: Deprecated: Private method used when serialising

height

set_dimension(*w=0*, *h=0*)

width

openpyxl.drawing.effect module

class openpyxl.drawing.effect.**AlphaBiLevelEffect**(*thresh=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

thresh

Values must be of type <class 'int'>

class openpyxl.drawing.effect.**AlphaCeilingEffect**

基类：*openpyxl.descriptors.serialisable.Serialisable*

class openpyxl.drawing.effect.**AlphaFloorEffect**

基类：*openpyxl.descriptors.serialisable.Serialisable*

class openpyxl.drawing.effect.**AlphaInverseEffect**

基类：*openpyxl.descriptors.serialisable.Serialisable*

class openpyxl.drawing.effect.**AlphaModulateEffect**(*cont=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

cont

Values must be of type <class 'openpyxl.drawing.effect.EffectContainer'>

class openpyxl.drawing.effect.**AlphaModulateFixedEffect**(*amt=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

amt

Values must be of type <class 'int'>

class openpyxl.drawing.effect.**AlphaReplaceEffect**(*a=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

a

Values must be of type <class 'int'>

class openpyxl.drawing.effect.**BiLevelEffect**(*thresh=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

thresh

Values must be of type <class 'int'>

class openpyxl.drawing.effect.**BlurEffect**(*rad=None*, *grow=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

grow

Values must be of type <class 'bool'>

rad

Values must be of type <class 'float'>

class openpyxl.drawing.effect.**Color**

基类: *openpyxl.descriptors.serialisable.Serialisable*

class openpyxl.drawing.effect.**ColorChangeEffect**(*useA=None*, *clrFrom=None*, *clrTo=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

clrFrom

Values must be of type <class 'openpyxl.drawing.effect.Color'>

clrTo

Values must be of type <class 'openpyxl.drawing.effect.Color'>

useA

Values must be of type <class 'bool'>

class openpyxl.drawing.effect.**ColorReplaceEffect**

基类: *openpyxl.descriptors.serialisable.Serialisable*

class openpyxl.drawing.effect.**DuotoneEffect**

基类: *openpyxl.descriptors.serialisable.Serialisable*

class openpyxl.drawing.effect.**EffectContainer**(*type=None*, *name=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

name

Values must be of type <class 'str'>

type

Value must be one of {'tree', 'sib'}

class openpyxl.drawing.effect.**EffectList**(*blur=None*, *fillOverlay=None*, *glow=None*, *innerShdw=None*, *outerShdw=None*, *prstShdw=None*, *reflection=None*, *softEdge=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

blur

Values must be of type <class 'openpyxl.drawing.effect.BlurEffect'>

fillOverlay

Values must be of type <class 'openpyxl.drawing.effect.FillOverlayEffect'>

glow

Values must be of type <class 'openpyxl.drawing.effect.GlowEffect'>

innerShdw

Values must be of type <class 'openpyxl.drawing.effect.InnerShadowEffect'>

outerShdw

Values must be of type <class 'openpyxl.drawing.effect.OuterShadow'>

prstShdw

Values must be of type <class 'openpyxl.drawing.effect.PresetShadowEffect'>

reflection

Values must be of type <class ‘openpyxl.drawing.effect.ReflectionEffect’ >

softEdge

Values must be of type <class ‘openpyxl.drawing.effect.SoftEdgesEffect’ >

class openpyxl.drawing.effect.**FillOverlayEffect**(*blend=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

blend

Value must be one of { ‘screen’ , ‘lighten’ , ‘over’ , ‘mult’ , ‘darken’ }

class openpyxl.drawing.effect.**GlowEffect**(*rad=None*, ***kw*)

基类: *openpyxl.drawing.colors.ColorChoice*

hslClr

Values must be of type <class ‘openpyxl.drawing.colors.HSLColor’ >

prstClr

Value must be one of { ‘darkSlateBlue’ , ‘mediumBlue’ , ‘sandyBrown’ , ‘medAquamarine’ , ‘ltGray’ , ‘deepPink’ , ‘cornsilk’ , ‘cyan’ , ‘dkSalmon’ , ‘darkSalmon’ , ‘ltSlateGrey’ , ‘linen’ , ‘steelBlue’ , ‘lightSeaGreen’ , ‘dkSeaGreen’ , ‘darkSlateGray’ , ‘medSeaGreen’ , ‘hotPink’ , ‘dkMagenta’ , ‘darkGrey’ , ‘rosyBrown’ , ‘white’ , ‘dkViolet’ , ‘ltSkyBlue’ , ‘fuchsia’ , ‘mediumVioletRed’ , ‘snow’ , ‘darkGoldenrod’ , ‘tan’ , ‘lightSteelBlue’ , ‘cornflowerBlue’ , ‘cadetBlue’ , ‘lightYellow’ , ‘orangeRed’ , ‘dkGreen’ , ‘paleGreen’ , ‘sienna’ , ‘lemonChiffon’ , ‘darkSeaGreen’ , ‘blueViolet’ , ‘crimson’ , ‘khaki’ , ‘ltYellow’ , ‘dkOrchid’ , ‘gainsboro’ , ‘ivory’ , ‘mediumAquamarine’ , ‘dkKhaki’ , ‘ltCoral’ , ‘azure’ , ‘indianRed’ , ‘darkOliveGreen’ , ‘aliceBlue’ , ‘dkBlue’ , ‘mediumPurple’ , ‘dkSlateGray’ , ‘dodgerBlue’ , ‘teal’ , ‘medPurple’ , ‘lightSalmon’ , ‘springGreen’ , ‘darkKhaki’ , ‘dkRed’ , ‘chocolate’ , ‘darkGreen’ , ‘ltCyan’ , ‘moccasin’ , ‘paleGoldenrod’ , ‘mediumTurquoise’ , ‘deepSkyBlue’ , ‘grey’ , ‘green’ , ‘peachPuff’ , ‘plum’ , ‘blue’ , ‘firebrick’ , ‘ltSalmon’ , ‘lightGreen’ , ‘darkRed’ , ‘royalBlue’ , ‘saddleBrown’ , ‘medVioletRed’ , ‘darkOrchid’ , ‘floralWhite’ , ‘lightPink’ , ‘lightSkyBlue’ , ‘ltGrey’ , ‘aquamarine’ , ‘medSpringGreen’ , ‘seaShell’ , ‘thistle’ , ‘lightBlue’ , ‘darkMagenta’ , ‘medOrchid’ , ‘ltSeaGreen’ , ‘magenta’ , ‘turquoise’ , ‘dkTurquoise’ , ‘wheat’ , ‘dkOrange’ , ‘lawnGreen’ , ‘oliveDrab’ , ‘skyBlue’ , ‘lightGoldenrodYellow’ , ‘whiteSmoke’ , ‘ltGoldenrodYellow’ , ‘chartreuse’ , ‘lightCoral’ , ‘violet’ , ‘silver’ , ‘black’ , ‘dimGray’ , ‘lavender’ , ‘medSlateBlue’ , ‘salmon’ , ‘navajoWhite’ , ‘gray’ , ‘dkOliveGreen’ , ‘dkSlateGrey’ , ‘lightSlateGrey’ , ‘yellowGreen’ , ‘darkSlateGrey’ , ‘gold’ , ‘mintCream’ , ‘paleVioletRed’ , ‘peru’ , ‘ltBlue’ , ‘ltSteelBlue’ , ‘dkGray’ , ‘seaGreen’ , ‘orchid’ , ‘lightSlateGray’ , ‘ltPink’ , ‘greenYellow’ , ‘midnightBlue’ , ‘red’ , ‘dkCyan’ , ‘lime’ , ‘orange’ , ‘darkTurquoise’ , ‘forestGreen’ , ‘honeydew’ , ‘pink’ , ‘mediumSlateBlue’ , ‘powderBlue’ , ‘darkBlue’ , ‘goldenrod’ , ‘yellow’ , ‘mediumSpringGreen’ , ‘medTurquoise’ , ‘paleTurquoise’ , ‘ltGreen’ , ‘slateBlue’ , ‘darkOrange’ , ‘lightGrey’ , ‘dkGrey’ , ‘darkCyan’ , ‘burlyWood’ , ‘slateGray’ , ‘mistyRose’ , ‘dkGoldenrod’ , ‘slateGrey’ , ‘medBlue’ , ‘lavenderBlush’ , ‘dimGrey’ , 'coral', 'blanchedAlmond', 'darkViolet', 'aqua', 'limeGreen', 'indigo', 'lightGray', 'lightCyan', 'purple', 'antiqueWhite', 'tomato', 'bisque', 'oldLace', 'beige', 'papayaWhip', 'maroon', 'mediumOrchid', 'ghostWhite', 'dkSlateBlue', 'ltSlateGray', 'mediumSeaGreen', 'brown', 'navy', 'olive', 'darkGray'}

rad

Values must be of type <class 'float'>

schemeClr

Values must be of type <class 'openpyxl.drawing.colors.SchemeColor'>

scrgbClr

Values must be of type <class 'openpyxl.drawing.colors.RGBPercent'>

srgbClr

Values must be of type <class 'str'>

sysClr

Values must be of type <class 'openpyxl.drawing.colors.SystemColor'>

class openpyxl.drawing.effect.**GrayscaleEffect**

基类：*openpyxl.descriptors.serialisable.Serialisable*

tagname = 'grayscl'

class openpyxl.drawing.effect.**HSLEffect**(*hue=None*, *sat=None*, *lum=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

hue

Values must be of type <class 'int'>

lum

Values must be of type <class 'int'>

sat

Values must be of type <class 'int'>

class openpyxl.drawing.effect.**InnerShadowEffect**(*blurRad=None*, *dist=None*, *dir=None*, ***kw*)

基类：*openpyxl.drawing.colors.ColorChoice*

blurRad

Values must be of type <class 'float'>

dir

Values must be of type <class 'int'>

dist

Values must be of type <class 'float'>

hslClr

Values must be of type <class 'openpyxl.drawing.colors.HSLColor'>

prstClr

Value must be one of {'darkSlateBlue', 'mediumBlue', 'sandyBrown', 'medAquamarine', 'ltGray', 'deepPink', 'cornsilk', 'cyan', 'dkSalmon', 'darkSalmon', 'ltSlateGrey', 'linen', 'steelBlue', 'lightSeaGreen', 'dkSeaGreen', 'darkSlateGray', 'medSeaGreen', 'hotPink', 'dkMagenta', 'darkGrey', 'rosyBrown', 'white', 'dkViolet', 'ltSkyBlue', 'fuchsia', 'mediumVioletRed', 'snow', 'darkGoldenrod', 'tan', 'lightSteelBlue', 'cornflowerBlue', 'cadetBlue', 'lightYellow', 'orangeRed', 'dkGreen', 'paleGreen', 'sienna', 'lemonChiffon', 'darkSeaGreen', 'blueViolet', 'crimson', 'khaki', 'ltYellow', 'dkOrchid', 'gainsboro', 'ivory', 'mediumAquamarine', 'dkKhaki', 'ltCoral', 'azure', 'indianRed', 'darkOliveGreen', 'aliceBlue', 'dkBlue', 'mediumPurple', 'dkSlateGray', 'dodgerBlue', 'teal', 'medPurple', 'lightSalmon', 'springGreen', 'darkKhaki', 'dkRed', 'chocolate', 'darkGreen', 'ltCyan', 'moccasin', 'paleGoldenrod', 'mediumTurquoise', 'deepSkyBlue', 'grey', 'green', 'peachPuff', 'plum', 'blue', 'firebrick', 'ltSalmon', 'lightGreen', 'darkRed', 'royalBlue', 'saddleBrown', 'medVioletRed', 'darkOrchid', 'floralWhite', 'lightPink', 'lightSkyBlue', 'ltGrey', 'aquamarine', 'medSpringGreen', 'seaShell', 'thistle', 'lightBlue', 'darkMagenta', 'medOrchid', 'ltSeaGreen', 'magenta', 'turquoise', 'dkTurquoise', 'wheat', 'dkOrange', 'lawnGreen', 'oliveDrab', 'skyBlue', 'lightGoldenrodYellow', 'whiteSmoke', 'ltGoldenrodYellow', 'chartreuse', 'lightCoral', 'violet', 'silver', 'black', 'dimGray', 'lavender', 'medSlateBlue', 'salmon', 'navajoWhite', 'gray', 'dkOliveGreen', 'dkSlateGrey', 'lightSlateGrey', 'yellowGreen', 'darkSlateGrey', 'gold', 'mintCream', 'paleVioletRed', 'peru', 'ltBlue', 'ltSteelBlue', 'dkGray', 'seaGreen', 'orchid', 'lightSlateGray', 'ltPink', 'greenYellow', 'midnightBlue', 'red', 'dkCyan', 'lime', 'orange', 'darkTurquoise', 'forestGreen', 'honeydew', 'pink', 'mediumSlateBlue', 'powderBlue', 'darkBlue', 'goldenrod', 'yellow', 'mediumSpringGreen', 'medTurquoise', 'paleTurquoise', 'ltGreen', 'slateBlue', 'darkOrange', 'lightGrey', 'dkGrey', 'darkCyan', 'burlyWood', 'slateGray', 'mistyRose', 'dkGoldenrod', 'slateGrey', 'medBlue', 'lavenderBlush', 'dimGrey', 'coral', 'blanchedAlmond', 'darkViolet', 'aqua', 'limeGreen', 'indigo', 'lightGray', 'lightCyan', 'purple', 'antiqueWhite', 'tomato', 'bisque', 'oldLace', 'beige', 'papayaWhip', 'maroon', 'mediumOrchid', 'ghostWhite', 'dkSlateBlue', 'ltSlateGray', 'mediumSeaGreen', 'brown', 'navy', 'olive', 'darkGray'}

schemeClr

Values must be of type <class 'openpyxl.drawing.colors.SchemeColor'>

scrgbClr

Values must be of type <class 'openpyxl.drawing.colors.RGBPercent'>

srgbClr

Values must be of type <class 'str'>

sysClr

Values must be of type <class 'openpyxl.drawing.colors.SystemColor'>

class openpyxl.drawing.effect.**LuminanceEffect**(*bright=0*, *contrast=0*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

bright

Values must be of type <class 'int' >

contrast

Values must be of type <class 'int' >

tagname = 'lum'

class openpyxl.drawing.effect.**OuterShadow**(*blurRad=None*, *dist=None*, *dir=None*, *sx=None*, *sy=None*, *kx=None*, *ky=None*, *algn=None*, *rotWithShape=None*, ***kw*)

基类：*openpyxl.drawing.colors.ColorChoice*

algn

Value must be one of { 'br' , 'b' , 'r' , 'tr' , 'tl' , 'ctr' , 'l' , 'bl' , 't' }

blurRad

Values must be of type <class 'float' >

dir

Values must be of type <class 'int' >

dist

Values must be of type <class 'float' >

hslClr

Values must be of type <class 'openpyxl.drawing.colors.HSLColor' >

kx

Values must be of type <class 'int' >

ky

Values must be of type <class 'int' >

prstClr

Value must be one of { 'darkSlateBlue' , 'mediumBlue' , 'sandyBrown' , 'medAquamarine' , 'ltGray' , 'deepPink' , 'cornsilk' , 'cyan' , 'dkSalmon' , 'darkSalmon' , 'ltSlateGrey' , 'linen' , 'steelBlue' , 'lightSeaGreen' , 'dkSeaGreen' , 'darkSlateGray' , 'medSeaGreen' , 'hotPink' , 'dkMagenta' , 'darkGrey' , 'rosyBrown' , 'white' , 'dkViolet' , 'ltSkyBlue' , 'fuchsia' , 'mediumVioletRed' , 'snow' , 'darkGoldenrod' , 'tan' , 'lightSteelBlue' , 'cornflowerBlue' , 'cadetBlue' , 'lightYellow' , 'orangeRed' , 'dkGreen' , 'paleGreen' , 'sienna' , 'lemonChiffon' , 'darkSeaGreen' , 'blueViolet' , 'crimson' , 'khaki' , 'ltYellow' , 'dkOrchid' , 'gainsboro' , 'ivory' , 'mediumAquamarine' , 'dkKhaki' , 'ltCoral' , 'azure' , 'indianRed' , 'darkOliveGreen' , 'aliceBlue' , 'dkBlue' , 'mediumPurple' , 'dkSlateGray' , 'dodgerBlue' , 'teal' , 'medPurple' , 'lightSalmon' , 'springGreen' , 'darkKhaki' , 'dkRed' , 'chocolate' , 'darkGreen' , 'ltCyan' , 'moccasin' , 'paleGoldenrod' , 'mediumTurquoise' , 'deepSkyBlue' , 'grey' , 'green' , 'peachPuff' , 'plum' , 'blue' , 'firebrick' , 'ltSalmon'

, 'lightGreen', 'darkRed', 'royalBlue', 'saddleBrown', 'medVioletRed', 'darkOrchid', 'floralWhite', 'lightPink', 'lightSkyBlue', 'ltGrey', 'aquamarine', 'medSpringGreen', 'seaShell', 'thistle', 'lightBlue', 'darkMagenta', 'medOrchid', 'ltSeaGreen', 'magenta', 'turquoise', 'dkTurquoise', 'wheat', 'dkOrange', 'lawnGreen', 'oliveDrab', 'skyBlue', 'lightGoldenrodYellow', 'whiteSmoke', 'ltGoldenrodYellow', 'chartreuse', 'lightCoral', 'violet', 'silver', 'black', 'dimGray', 'lavender', 'medSlateBlue', 'salmon', 'navajoWhite', 'gray', 'dkOliveGreen', 'dkSlateGrey', 'lightSlateGrey', 'yellowGreen', 'darkSlateGrey', 'gold', 'mintCream', 'paleVioletRed', 'peru', 'ltBlue', 'ltSteelBlue', 'dkGray', 'seaGreen', 'orchid', 'lightSlateGray', 'ltPink', 'greenYellow', 'midnightBlue', 'red', 'dkCyan', 'lime', 'orange', 'darkTurquoise', 'forestGreen', 'honeydew', 'pink', 'mediumSlateBlue', 'powderBlue', 'darkBlue', 'goldenrod', 'yellow', 'mediumSpringGreen', 'medTurquoise', 'paleTurquoise', 'ltGreen', 'slateBlue', 'darkOrange', 'lightGrey', 'dkGrey', 'darkCyan', 'burlyWood', 'slateGray', 'mistyRose', 'dkGoldenrod', 'slateGrey', 'medBlue', 'lavenderBlush', 'dimGrey', 'coral', 'blanchedAlmond', 'darkViolet', 'aqua', 'limeGreen', 'indigo', 'lightGray', 'lightCyan', 'purple', 'antiqueWhite', 'tomato', 'bisque', 'oldLace', 'beige', 'papayaWhip', 'maroon', 'mediumOrchid', 'ghostWhite', 'dkSlateBlue', 'ltSlateGray', 'mediumSeaGreen', 'brown', 'navy', 'olive', 'darkGray'}

rotWithShape

Values must be of type <class 'bool'>

schemeClr

Values must be of type <class 'openpyxl.drawing.colors.SchemeColor'>

scrgbClr

Values must be of type <class 'openpyxl.drawing.colors.RGBPercent'>

srgbClr

Values must be of type <class 'str'>

sx

Values must be of type <class 'int'>

sy

Values must be of type <class 'int'>

sysClr

Values must be of type <class 'openpyxl.drawing.colors.SystemColor'>

tagname = 'outerShdw'

class openpyxl.drawing.effect.PresetShadowEffect(*prst=None*, *dist=None*, *dir=None*, ***kw*)

基类: *openpyxl.drawing.colors.ColorChoice*

dir

Values must be of type <class 'int'>

`dist`

Values must be of type <class ‘float’ >

`hslClr`

Values must be of type <class ‘openpyxl.drawing.colors.HSLColor’ >

`prst`

Value must be one of { ‘shdw19’ , ‘shdw6’ , ‘shdw7’ , ‘shdw17’ , ‘shdw20’ , ‘shdw12’ , ‘shdw9’ , ‘shdw18’ , ‘shdw13’ , ‘shdw3’ , ‘shdw14’ , ‘shdw15’ , ‘shdw10’ , ‘shdw1’ , ‘shdw11’ , ‘shdw16’ , ‘shdw2’ , ‘shdw5’ , ‘shdw8’ , ‘shdw4’ }

`prstClr`

Value must be one of { ‘darkSlateBlue’ , ‘mediumBlue’ , ‘sandyBrown’ , ‘medAquamarine’ , ‘ltGray’ , ‘deepPink’ , ‘cornsilk’ , ‘cyan’ , ‘dkSalmon’ , ‘darkSalmon’ , ‘ltSlateGrey’ , ‘linen’ , ‘steelBlue’ , ‘lightSeaGreen’ , ‘dkSeaGreen’ , ‘darkSlateGray’ , ‘medSeaGreen’ , ‘hotPink’ , ‘dkMagenta’ , ‘darkGrey’ , ‘rosyBrown’ , ‘white’ , ‘dkViolet’ , ‘ltSkyBlue’ , ‘fuchsia’ , ‘mediumVioletRed’ , ‘snow’ , ‘darkGoldenrod’ , ‘tan’ , ‘lightSteelBlue’ , ‘cornflowerBlue’ , ‘cadetBlue’ , ‘lightYellow’ , ‘orangeRed’ , ‘dkGreen’ , ‘paleGreen’ , ‘sienna’ , ‘lemonChiffon’ , ‘darkSeaGreen’ , ‘blueViolet’ , ‘crimson’ , ‘khaki’ , ‘ltYellow’ , ‘dkOrchid’ , ‘gainsboro’ , ‘ivory’ , ‘mediumAquamarine’ , ‘dkKhaki’ , ‘ltCoral’ , ‘azure’ , ‘indianRed’ , ‘darkOliveGreen’ , ‘aliceBlue’ , ‘dkBlue’ , ‘mediumPurple’ , ‘dkSlateGray’ , ‘dodgerBlue’ , ‘teal’ , ‘medPurple’ , ‘lightSalmon’ , ‘springGreen’ , ‘darkKhaki’ , ‘dkRed’ , ‘chocolate’ , ‘darkGreen’ , ‘ltCyan’ , ‘moccasin’ , ‘paleGoldenrod’ , ‘mediumTurquoise’ , ‘deepSkyBlue’ , ‘grey’ , ‘green’ , ‘peachPuff’ , ‘plum’ , ‘blue’ , ‘firebrick’ , ‘ltSalmon’ , ‘lightGreen’ , ‘darkRed’ , ‘royalBlue’ , ‘saddleBrown’ , ‘medVioletRed’ , ‘darkOrchid’ , ‘floralWhite’ , ‘lightPink’ , ‘lightSkyBlue’ , ‘ltGrey’ , ‘aquamarine’ , ‘medSpringGreen’ , ‘seaShell’ , ‘thistle’ , ‘lightBlue’ , ‘darkMagenta’ , ‘medOrchid’ , ‘ltSeaGreen’ , ‘magenta’ , ‘turquoise’ , ‘dkTurquoise’ , ‘wheat’ , ‘dkOrange’ , ‘lawnGreen’ , ‘oliveDrab’ , ‘skyBlue’ , ‘lightGoldenrodYellow’ , ‘whiteSmoke’ , ‘ltGoldenrodYellow’ , ‘chartreuse’ , ‘lightCoral’ , ‘violet’ , ‘silver’ , ‘black’ , ‘dimGray’ , ‘lavender’ , ‘medSlateBlue’ , ‘salmon’ , ‘navajoWhite’ , ‘gray’ , ‘dkOliveGreen’ , ‘dkSlateGrey’ , ‘lightSlateGrey’ , ‘yellowGreen’ , ‘darkSlateGrey’ , ‘gold’ , ‘mintCream’ , ‘paleVioletRed’ , ‘peru’ , ‘ltBlue’ , ‘ltSteelBlue’ , ‘dkGray’ , ‘seaGreen’ , ‘orchid’ , ‘lightSlateGray’ , ‘ltPink’ , ‘greenYellow’ , ‘midnightBlue’ , ‘red’ , ‘dkCyan’ , ‘lime’ , ‘orange’ , ‘darkTurquoise’ , ‘forestGreen’ , ‘honeydew’ , ‘pink’ , ‘mediumSlateBlue’ , ‘powderBlue’ , ‘darkBlue’ , ‘goldenrod’ , ‘yellow’ , ‘mediumSpringGreen’ , ‘medTurquoise’ , ‘paleTurquoise’ , ‘ltGreen’ , ‘slateBlue’ , ‘darkOrange’ , ‘lightGrey’ , ‘dkGrey’ , ‘darkCyan’ , ‘burlyWood’ , ‘slateGray’ , ‘mistyRose’ , ‘dkGoldenrod’ , ‘slateGrey’ , ‘medBlue’ , ‘lavenderBlush’ , ‘dimGrey’ , ‘coral’ , ‘blanchedAlmond’ , ‘darkViolet’ , ‘aqua’ , ‘limeGreen’ , ‘indigo’ , ‘lightGray’ , ‘lightCyan’ , ‘purple’ , ‘antiqueWhite’ , ‘tomato’ , ‘bisque’ , ‘oldLace’ , ‘beige’ , ‘papayaWhip’ , ‘maroon’ , ‘mediumOrchid’ , ‘ghostWhite’ , ‘dkSlateBlue’ , ‘ltSlateGray’ , ‘mediumSeaGreen’ , ‘brown’ , ‘navy’ , ‘olive’ , ‘darkGray’ }

`schemeClr`

Values must be of type <class ‘openpyxl.drawing.colors.SchemeColor’ >

scrgbClr

Values must be of type <class ‘openpyxl.drawing.colors.RGBPercent’ >

srgbClr

Values must be of type <class ‘str’ >

sysClr

Values must be of type <class ‘openpyxl.drawing.colors.SystemColor’ >

class openpyxl.drawing.effect.**ReflectionEffect**(*blurRad=None*, *stA=None*, *stPos=None*, *endA=None*, *endPos=None*, *dist=None*, *dir=None*, *fadeDir=None*, *sx=None*, *sy=None*, *kx=None*, *ky=None*, *algn=None*, *rotWithShape=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

algn

Value must be one of { ‘br’ , ‘b’ , ‘r’ , ‘tr’ , ‘tl’ , ‘ctr’ , ‘l’ , ‘bl’ , ‘t’ }

blurRad

Values must be of type <class ‘float’ >

dir

Values must be of type <class ‘int’ >

dist

Values must be of type <class ‘float’ >

endA

Values must be of type <class ‘int’ >

endPos

Values must be of type <class ‘int’ >

fadeDir

Values must be of type <class ‘int’ >

kx

Values must be of type <class ‘int’ >

ky

Values must be of type <class ‘int’ >

rotWithShape

Values must be of type <class ‘bool’ >

stA

Values must be of type <class ‘int’ >

stPos

Values must be of type <class ‘int’ >

sx

Values must be of type <class ‘int’ >

sy

Values must be of type <class ‘int’ >

class openpyxl.drawing.effect.**SoftEdgesEffect**(*rad=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

rad

Values must be of type <class ‘float’ >

class openpyxl.drawing.effect.**TintEffect**(*hue=0*, *amt=0*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

amt

Values must be of type <class ‘int’ >

hue

Values must be of type <class ‘int’ >

tagname = 'tint'

openpyxl.drawing.fill module

class openpyxl.drawing.fill.**Blip**(*cstate=None*, *embed=None*, *link=None*, *noGrp=None*, *noSelect=None*, *noRot=None*, *noChangeAspect=None*, *noMove=None*, *noResize=None*, *noEditPoints=None*, *noAdjustHandles=None*, *noChangeArrowheads=None*, *noChangeShapeType=None*, *extLst=None*, *alphaBiLevel=None*, *alphaCeiling=None*, *alphaFloor=None*, *alphaInv=None*, *alphaMod=None*, *alphaModFix=None*, *alphaRepl=None*, *biLevel=None*, *blur=None*, *clrChange=None*, *clrRepl=None*, *duotone=None*, *fillOverlay=None*, *grayscl=None*, *hsl=None*, *lum=None*, *tint=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

alphaBiLevel

Values must be of type <class ‘openpyxl.drawing.effect.AlphaBiLevelEffect’ >

alphaCeiling

Values must be of type <class ‘openpyxl.drawing.effect.AlphaCeilingEffect’ >

alphaFloor

Values must be of type <class ‘openpyxl.drawing.effect.AlphaFloorEffect’ >

alphaInv
Values must be of type <class 'openpyxl.drawing.effect.AlphaInverseEffect' >

alphaMod
Values must be of type <class 'openpyxl.drawing.effect.AlphaModulateEffect' >

alphaModFix
Values must be of type <class 'openpyxl.drawing.effect.AlphaModulateFixedEffect' >

alphaRepl
Values must be of type <class 'openpyxl.drawing.effect.AlphaReplaceEffect' >

biLevel
Values must be of type <class 'openpyxl.drawing.effect.BiLevelEffect' >

blur
Values must be of type <class 'openpyxl.drawing.effect.BlurEffect' >

clrChange
Values must be of type <class 'openpyxl.drawing.effect.ColorChangeEffect' >

clrRepl
Values must be of type <class 'openpyxl.drawing.effect.ColorReplaceEffect' >

cstate
Value must be one of { 'screen' , 'email' , 'hqprint' , 'print' }

duotone
Values must be of type <class 'openpyxl.drawing.effect.DuotoneEffect' >

embed
Values must be of type <class 'str' >

extLst
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

fillOverlay
Values must be of type <class 'openpyxl.drawing.effect.FillOverlayEffect' >

grayscl
Values must be of type <class 'openpyxl.drawing.effect.GrayscaleEffect' >

hsl
Values must be of type <class 'openpyxl.drawing.effect.HSLEffect' >

link
Values must be of type <class 'str' >

lum
Values must be of type <class 'openpyxl.drawing.effect.LuminanceEffect' >

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

noAdjustHandles

Values must be of type <class 'bool' >

noChangeArrowheads

Values must be of type <class 'bool' >

noChangeAspect

Values must be of type <class 'bool' >

noChangeShapeType

Values must be of type <class 'bool' >

noEditPoints

Values must be of type <class 'bool' >

noGrp

Values must be of type <class 'bool' >

noMove

Values must be of type <class 'bool' >

noResize

Values must be of type <class 'bool' >

noRot

Values must be of type <class 'bool' >

noSelect

Values must be of type <class 'bool' >

tagname = 'blip'

tint

Values must be of type <class 'openpyxl.drawing.effect.TintEffect' >

class openpyxl.drawing.fill.**BlipFillProperties**(*dpi=None, rotWithShape=None, blip=None, tile=None, stretch=<openpyxl.drawing.fill.StretchInfoProperties object> Parameters: fillRect=<openpyxl.drawing.fill.RelativeRect object> Parameters: l=None, t=None, r=None, b=None, srcRect=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

blip

Values must be of type <class 'openpyxl.drawing.fill.Blip' >

dpi

Values must be of type <class 'int' >

rotWithShape

Values must be of type <class 'bool'>

srcRect

Values must be of type <class 'openpyxl.drawing.fill.RelativeRect'>

stretch

Values must be of type <class 'openpyxl.drawing.fill.StretchInfoProperties'>

tagname = 'blipFill'

tile

Values must be of type <class 'openpyxl.drawing.fill.TileInfoProperties'>

class openpyxl.drawing.fill.**GradientFillProperties**(*flip=None*, *rotWithShape=None*, *gsLst=()*, *lin=None*, *path=None*, *tileRect=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

flip

Value must be one of {'y', 'xy', 'x'}

gsLst

Wrap a sequence in an containing object

lin

Values must be of type <class 'openpyxl.drawing.fill.LinearShadeProperties'>

linear

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

path

Values must be of type <class 'openpyxl.drawing.fill.PathShadeProperties'>

rotWithShape

Values must be of type <class 'bool'>

stop_list

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

tagname = 'gradFill'

tileRect

Values must be of type <class 'openpyxl.drawing.fill.RelativeRect'>

class openpyxl.drawing.fill.**GradientStop**(*pos=None*, *scrgbClr=None*, *srgbClr=None*, *hslClr=None*, *sysClr=None*, *schemeClr=None*, *prstClr=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`RGB`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

`RGBPercent`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

`hslClr`

Values must be of type <class ‘openpyxl.drawing.colors.HSLColor’ >

`namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'`

`pos`

Values must be of type <class ‘float’ >

`prstClr`

Value must be one of { ‘darkSlateBlue’ , ‘mediumBlue’ , ‘sandyBrown’ , ‘medAquamarine’ , ‘ltGray’ , ‘deepPink’ , ‘cornsilk’ , ‘cyan’ , ‘dkSalmon’ , ‘darkSalmon’ , ‘ltSlateGrey’ , ‘linen’ , ‘steelBlue’ , ‘lightSeaGreen’ , ‘dkSeaGreen’ , ‘darkSlateGray’ , ‘medSeaGreen’ , ‘hotPink’ , ‘dkMagenta’ , ‘darkGrey’ , ‘rosyBrown’ , ‘white’ , ‘dkViolet’ , ‘ltSkyBlue’ , ‘fuchsia’ , ‘mediumVioletRed’ , ‘snow’ , ‘darkGoldenrod’ , ‘tan’ , ‘lightSteelBlue’ , ‘cornflowerBlue’ , ‘cadetBlue’ , ‘lightYellow’ , ‘orangeRed’ , ‘dkGreen’ , ‘paleGreen’ , ‘sienna’ , ‘lemonChiffon’ , ‘darkSeaGreen’ , ‘blueViolet’ , ‘crimson’ , ‘khaki’ , ‘ltYellow’ , ‘dkOrchid’ , ‘gainsboro’ , ‘ivory’ , ‘mediumAquamarine’ , ‘dkKhaki’ , ‘ltCoral’ , ‘azure’ , ‘indianRed’ , ‘darkOliveGreen’ , ‘aliceBlue’ , ‘dkBlue’ , ‘mediumPurple’ , ‘dkSlateGray’ , ‘dodgerBlue’ , ‘teal’ , ‘medPurple’ , ‘lightSalmon’ , ‘springGreen’ , ‘darkKhaki’ , ‘dkRed’ , ‘chocolate’ , ‘darkGreen’ , ‘ltCyan’ , ‘moccasin’ , ‘paleGoldenrod’ , ‘mediumTurquoise’ , ‘deepSkyBlue’ , ‘grey’ , ‘green’ , ‘peachPuff’ , ‘plum’ , ‘blue’ , ‘firebrick’ , ‘ltSalmon’ , ‘lightGreen’ , ‘darkRed’ , ‘royalBlue’ , ‘saddleBrown’ , ‘medVioletRed’ , ‘darkOrchid’ , ‘floralWhite’ , ‘lightPink’ , ‘lightSkyBlue’ , ‘ltGrey’ , ‘aquamarine’ , ‘medSpringGreen’ , ‘seaShell’ , ‘thistle’ , ‘lightBlue’ , ‘darkMagenta’ , ‘medOrchid’ , ‘ltSeaGreen’ , ‘magenta’ , ‘turquoise’ , ‘dkTurquoise’ , ‘wheat’ , ‘dkOrange’ , ‘lawnGreen’ , ‘oliveDrab’ , ‘skyBlue’ , ‘lightGoldenrodYellow’ , ‘whiteSmoke’ , ‘ltGoldenrodYellow’ , ‘chartreuse’ , ‘lightCoral’ , ‘violet’ , ‘silver’ , ‘black’ , ‘dimGray’ , ‘lavender’ , ‘medSlateBlue’ , ‘salmon’ , ‘navajoWhite’ , ‘gray’ , ‘dkOliveGreen’ , ‘dkSlateGrey’ , ‘lightSlateGrey’ , ‘yellowGreen’ , ‘darkSlateGrey’ , ‘gold’ , ‘mintCream’ , ‘paleVioletRed’ , ‘peru’ , ‘ltBlue’ , ‘ltSteelBlue’ , ‘dkGray’ , ‘seaGreen’ , ‘orchid’ , ‘lightSlateGray’ , ‘ltPink’ , ‘greenYellow’ , ‘midnightBlue’ , ‘red’ , ‘dkCyan’ , ‘lime’ , ‘orange’ , ‘darkTurquoise’ , ‘forestGreen’ , ‘honeydew’ , ‘pink’ , ‘mediumSlateBlue’ , ‘powderBlue’ , ‘darkBlue’ , ‘goldenrod’ , ‘yellow’ , ‘mediumSpringGreen’ , ‘medTurquoise’ , ‘paleTurquoise’ , ‘ltGreen’ , ‘slateBlue’ , ‘darkOrange’ , ‘lightGrey’ , ‘dkGrey’ , ‘darkCyan’ , ‘burlyWood’ , ‘slateGray’ , ‘mistyRose’ , ‘dkGoldenrod’ , ‘slateGrey’ , ‘medBlue’ , ‘lavenderBlush’ , ‘dimGrey’ , 'coral' , 'blanchedAlmond' , 'darkViolet' , 'aqua' , 'limeGreen' , 'indigo' , 'lightGray' , 'lightCyan' , 'purple' , 'antiqueWhite' , 'tomato' , 'bisque' , 'oldLace' , 'beige' , 'papayaWhip' , 'maroon' , 'mediumOrchid' , 'ghostWhite' , 'dkSlateBlue' , 'ltSlateGray' , 'mediumSeaGreen' , 'brown' , 'navy' , 'olive' , 'darkGray' }

`schemeClr`

Values must be of type <class 'openpyxl.drawing.colors.SchemeColor' >

`scrgbClr`

Values must be of type <class 'openpyxl.drawing.colors.RGBPercent' >

`srgbClr`

Values must be of type <class 'str' >

`sysClr`

Values must be of type <class 'openpyxl.drawing.colors.SystemColor' >

`tagname = 'gs'`

`class openpyxl.drawing.fill.LinearShadeProperties(`*ang=None*, *scaled=None*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`ang`

Values must be of type <class 'int' >

`namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'`

`scaled`

Values must be of type <class 'bool' >

`tagname = 'lin'`

`class openpyxl.drawing.fill.PathShadeProperties(`*path=None*, *fillToRect=None*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`fillToRect`

Values must be of type <class 'openpyxl.drawing.fill.RelativeRect' >

`namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'`

`path`

Value must be one of { 'shape' , 'rect' , 'circle' }

`tagname = 'path'`

`class openpyxl.drawing.fill.PatternFillProperties(`*prst=None*, *fgClr=None*, *bgClr=None*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`background`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

bgClr

Values must be of type <class 'openpyxl.drawing.colors.ColorChoice'>

fgClr

Values must be of type <class 'openpyxl.drawing.colors.ColorChoice'>

foreground

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

preset

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

prst

Value must be one of {'pct5', 'pct30', 'dkHorz', 'diagBrick', 'dotDmnd', 'sphere', 'narHorz', 'pct20', 'dkDnDiag', 'dotGrid', 'pct40', 'trellis', 'pct10', 'wdDnDiag', 'pct80', 'solidDmnd', 'smCheck', 'dashDnDiag', 'upDiag', 'ltUpDiag', 'pct90', 'pct75', 'ltDnDiag', 'dashVert', 'plaid', 'shingle', 'horzBrick', 'pct60', 'pct25', 'pct50', 'openDmnd', 'horz', 'zigZag', 'ltVert', 'cross', 'lgGrid', 'dnDiag', 'lgConfetti', 'lgCheck', 'divot', 'wave', 'narVert', 'pct70', 'dashUpDiag', 'weave', 'dashHorz', 'wdUpDiag', 'smGrid', 'smConfetti', 'dkUpDiag', 'diagCross', 'ltHorz', 'dkVert', 'vert'}

tagname = 'pattFill'

class openpyxl.drawing.fill.RelativeRect(*l=None*, *t=None*, *r=None*, *b=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

b

Values must be of type <class 'float'>

bottom

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

l

Values must be of type <class 'float'>

left

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

r

Values must be of type <class 'float'>

right

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

t

Values must be of type <class 'float' >

tagname = 'rect'

top

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

class openpyxl.drawing.fill.SolidColorFillProperties(*scrgbClr=None*, *srgbClr=None*, *hslClr=None*, *sysClr=None*, *schemeClr=None*, *prstClr=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

RGB

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

RGBPercent

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

hslClr

Values must be of type <class 'openpyxl.drawing.colors.HSLColor' >

prstClr

Value must be one of { 'darkSlateBlue' , 'mediumBlue' , 'sandyBrown' , 'medAquamarine' , 'ltGray' , 'deepPink' , 'cornsilk' , 'cyan' , 'dkSalmon' , 'darkSalmon' , 'ltSlateGrey' , 'linen' , 'steelBlue' , 'lightSeaGreen' , 'dkSeaGreen' , 'darkSlateGray' , 'medSeaGreen' , 'hotPink' , 'dkMagenta' , 'darkGrey' , 'rosyBrown' , 'white' , 'dkViolet' , 'ltSkyBlue' , 'fuchsia' , 'mediumVioletRed' , 'snow' , 'darkGoldenrod' , 'tan' , 'lightSteelBlue' , 'cornflowerBlue' , 'cadetBlue' , 'lightYellow' , 'orangeRed' , 'dkGreen' , 'paleGreen' , 'sienna' , 'lemonChiffon' , 'darkSeaGreen' , 'blueViolet' , 'crimson' , 'khaki' , 'ltYellow' , 'dkOrchid' , 'gainsboro' , 'ivory' , 'mediumAquamarine' , 'dkKhaki' , 'ltCoral' , 'azure' , 'indianRed' , 'darkOliveGreen' , 'aliceBlue' , 'dkBlue' , 'mediumPurple' , 'dkSlateGray' , 'dodgerBlue' , 'teal' , 'medPurple' , 'lightSalmon' , 'springGreen' , 'darkKhaki' , 'dkRed' , 'chocolate' , 'darkGreen' , 'ltCyan' , 'moccasin' , 'paleGoldenrod' , 'mediumTurquoise' , 'deepSkyBlue' , 'grey' , 'green' , 'peachPuff' , 'plum' , 'blue' , 'firebrick' , 'ltSalmon' , 'lightGreen' , 'darkRed' , 'royalBlue' , 'saddleBrown' , 'medVioletRed' , 'darkOrchid' , 'floralWhite' , 'lightPink' , 'lightSkyBlue' , 'ltGrey' , 'aquamarine' , 'medSpringGreen' , 'seaShell' , 'thistle' , 'lightBlue' , 'darkMagenta' , 'medOrchid' , 'ltSeaGreen' , 'magenta' , 'turquoise' , 'dkTurquoise' , 'wheat' , 'dkOrange' , 'lawnGreen' , 'oliveDrab' , 'skyBlue' , 'lightGoldenrodYellow' , 'whiteSmoke' , 'ltGoldenrodYellow' , 'chartreuse'

, ‘lightCoral’, ‘violet’, ‘silver’, ‘black’, ‘dimGray’, ‘lavender’, ‘medSlateBlue’, ‘salmon’, ‘navajoWhite’, ‘gray’, ‘dkOliveGreen’, ‘dkSlateGrey’, ‘lightSlateGrey’, ‘yellowGreen’, ‘darkSlateGrey’, ‘gold’, ‘mintCream’, ‘paleVioletRed’, ‘peru’, ‘ltBlue’, ‘ltSteelBlue’, ‘dkGray’, ‘seaGreen’, ‘orchid’, ‘lightSlateGray’, ‘ltPink’, ‘greenYellow’, ‘midnightBlue’, ‘red’, ‘dkCyan’, ‘lime’, ‘orange’, ‘darkTurquoise’, ‘forestGreen’, ‘honeydew’, ‘pink’, ‘mediumSlateBlue’, ‘powderBlue’, ‘darkBlue’, ‘goldenrod’, ‘yellow’, ‘mediumSpringGreen’, ‘medTurquoise’, ‘paleTurquoise’, ‘ltGreen’, ‘slateBlue’, ‘darkOrange’, ‘lightGrey’, ‘dkGrey’, ‘darkCyan’, ‘burlyWood’, ‘slateGray’, ‘mistyRose’, ‘dkGoldenrod’, ‘slateGrey’, ‘medBlue’, ‘lavenderBlush’, ‘dimGrey’, ‘coral’, ‘blanchedAlmond’, ‘darkViolet’, ‘aqua’, ‘limeGreen’, ‘indigo’, ‘lightGray’, ‘lightCyan’, ‘purple’, ‘antiqueWhite’, ‘tomato’, ‘bisque’, ‘oldLace’, ‘beige’, ‘papayaWhip’, ‘maroon’, ‘mediumOrchid’, ‘ghostWhite’, ‘dkSlateBlue’, ‘ltSlateGray’, ‘mediumSeaGreen’, ‘brown’, ‘navy’, ‘olive’, ‘darkGray’ }

schemeClr

Values must be of type <class ‘openpyxl.drawing.colors.SchemeColor’ >

scrgbClr

Values must be of type <class ‘openpyxl.drawing.colors.RGBPercent’ >

srgbClr

Values must be of type <class ‘str’ >

sysClr

Values must be of type <class ‘openpyxl.drawing.colors.SystemColor’ >

tagname = 'solidFill'

class openpyxl.drawing.fill.**StretchInfoProperties**(*fillRect=<openpyxl.drawing.fill.RelativeRect object> Parameters: l=None, t=None, r=None, b=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

fillRect

Values must be of type <class ‘openpyxl.drawing.fill.RelativeRect’ >

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

tagname = 'stretch'

class openpyxl.drawing.fill.**TileInfoProperties**(*tx=None, ty=None, sx=None, sy=None, flip=None, algn=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

algn

Value must be one of { ‘br’, ‘b’, ‘r’, ‘tr’, ‘tl’, ‘ctr’, ‘l’, ‘bl’, ‘t’ }

flip

Value must be one of { ‘y’, ‘xy’, ‘x’ }

sx

Values must be of type <class 'int'>

sy

Values must be of type <class 'int'>

tx

Values must be of type <class 'int'>

ty

Values must be of type <class 'int'>

openpyxl.drawing.geometry module

`class openpyxl.drawing.geometry.AdjPoint2D(x=None, y=None)`

基类: *openpyxl.descriptors.serialisable.Serialisable*

x

Values must be of type <class 'int'>

y

Values must be of type <class 'int'>

`class openpyxl.drawing.geometry.AdjustHandleList`

基类: *openpyxl.descriptors.serialisable.Serialisable*

`class openpyxl.drawing.geometry.Backdrop(anchor=None, norm=None, up=None, extLst=None)`

基类: *openpyxl.descriptors.serialisable.Serialisable*

anchor

Values must be of type <class 'openpyxl.drawing.geometry.Point3D'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

norm

Values must be of type <class 'openpyxl.drawing.geometry.Vector3D'>

up

Values must be of type <class 'openpyxl.drawing.geometry.Vector3D'>

`class openpyxl.drawing.geometry.Bevel(w=None, h=None, prst=None)`

基类: *openpyxl.descriptors.serialisable.Serialisable*

h

Values must be of type <class 'int'>

prst

Value must be one of {'softRound', 'hardEdge', 'circle', 'riblet', 'convex', 'slope'

, 'coolSlant' , 'relaxedInset' , 'artDeco' , 'cross' , 'divot' , 'angle' }

tagname = 'bevel'

w

Values must be of type <class 'int' >

class openpyxl.drawing.geometry.**Camera**(*prst=None*, *fov=None*, *zoom=None*, *rot=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

fov

Values must be of type <class 'int' >

prst

Value must be one of { 'obliqueBottomLeft' , 'isometricBottomUp' , 'legacyPerspectiveTop' , 'legacyObliqueBottom' , 'legacyPerspectiveTopLeft' , 'isometricRightUp' , 'perspectiveContrastingRightFacing' , 'obliqueTopLeft' , 'isometricOffAxis3Left' , 'perspectiveRight' , 'isometricOffAxis4Right' , 'obliqueBottom' , 'isometricRightDown' , 'isometricOffAxis1Right' , 'perspectiveContrastingLeftFacing' , 'perspectiveAboveLeftFacing' , 'isometricLeftUp' , 'legacyObliqueBottomLeft' , 'orthographicFront' , 'obliqueTop' , 'isometricOffAxis1Top' , 'legacyObliqueFront' , 'isometricBottomDown' , 'legacyPerspectiveBottomRight' , 'isometricOffAxis2Top' , 'isometricOffAxis4Left' , 'obliqueTopRight' , 'isometricOffAxis2Left' , 'perspectiveAboveRightFacing' , 'legacyPerspectiveBottom' , 'perspectiveHeroicRightFacing' , 'perspectiveFront' , 'legacyPerspectiveTopRight' , 'perspectiveHeroicExtremeLeftFacing' , 'legacyObliqueTopLeft' , 'isometricOffAxis1Left' , 'legacyObliqueLeft' , 'perspectiveBelow' , 'obliqueBottomRight' , 'isometricOffAxis4Bottom' , 'obliqueRight' , 'legacyPerspectiveFront' , 'perspectiveAbove' , 'legacyObliqueRight' , 'legacyPerspectiveRight' , 'legacyPerspectiveBottomLeft' , 'legacyObliqueTopRight' , 'perspectiveHeroicLeftFacing' , 'perspectiveRelaxed' , 'legacyObliqueTop' , 'legacyPerspectiveLeft' , 'isometricTopDown' , 'perspectiveLeft' , 'isometricOffAxis2Right' , 'isometricOffAxis3Bottom' , 'isometricOffAxis3Right' , 'legacyObliqueBottomRight' , 'perspectiveRelaxedModerately' , 'isometricLeftDown' , 'isometricTopUp' , 'perspectiveHeroicExtremeRightFacing' , 'obliqueLeft' }

rot

Values must be of type <class 'openpyxl.drawing.geometry.SphereCoords' >

tagname = 'camera'

zoom

Values must be of type <class 'openpyxl.descriptors.excel.Percentage' >

class openpyxl.drawing.geometry.**ConnectionSite**(*ang=None*, *pos=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

ang

Values must be of type <class 'float' >

pos

Values must be of type <class ‘openpyxl.drawing.geometry.AdjPoint2D’ >

class openpyxl.drawing.geometry.**ConnectionSiteList**(*cxn=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

cxn

Values must be of type <class ‘openpyxl.drawing.geometry.ConnectionSite’ >

class openpyxl.drawing.geometry.**CustomGeometry2D**(*avLst=None, gdLst=None, ahLst=None, cxnLst=None, rect=None, pathLst=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

ahLst

Values must be of type <class ‘openpyxl.drawing.geometry.AdjustHandleList’ >

avLst

Values must be of type <class ‘openpyxl.drawing.geometry.GeomGuideList’ >

cxnLst

Values must be of type <class ‘openpyxl.drawing.geometry.ConnectionSiteList’ >

gdLst

Values must be of type <class ‘openpyxl.drawing.geometry.GeomGuideList’ >

pathLst

Values must be of type <class ‘openpyxl.drawing.geometry.Path2DList’ >

class openpyxl.drawing.geometry.**FontReference**(*idx=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

idx

Value must be one of { ‘major’ , ‘minor’ }

class openpyxl.drawing.geometry.**GeomGuide**(*name=None, fmla=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

fmla

Values must be of type <class ‘str’ >

name

Values must be of type <class ‘str’ >

class openpyxl.drawing.geometry.**GeomGuideList**(*gd=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

gd

Values must be of type <class ‘openpyxl.drawing.geometry.GeomGuide’ >

class openpyxl.drawing.geometry.**GeomRect**(*l=None, t=None, r=None, b=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

`b`

Values must be of type <class ‘int’ >

`l`

Values must be of type <class ‘int’ >

`r`

Values must be of type <class ‘int’ >

`t`

Values must be of type <class ‘int’ >

class openpyxl.drawing.geometry.GroupTransform2D(*rot=0, flipH=None, flipV=None, off=None, ext=None, chOff=None, chExt=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`chExt`

Values must be of type <class ‘openpyxl.drawing.geometry.PositiveSize2D’ >

`chOff`

Values must be of type <class ‘openpyxl.drawing.geometry.Point2D’ >

`ext`

Values must be of type <class ‘openpyxl.drawing.geometry.PositiveSize2D’ >

`flipH`

Values must be of type <class ‘bool’ >

`flipV`

Values must be of type <class ‘bool’ >

`namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'`

`off`

Values must be of type <class ‘openpyxl.drawing.geometry.Point2D’ >

`rot`

Values must be of type <class ‘int’ >

`tagname = 'xfrm'`

class openpyxl.drawing.geometry.LightRig(*rig=None, dir=None, rot=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`dir`

Value must be one of { ‘br’ , ‘b’ , ‘r’ , ‘tr’ , ‘tl’ , ‘l’ , ‘bl’ , ‘t’ }

`rig`

Value must be one of { ‘legacyNormal2’ , ‘legacyHarsh3’ , ‘twoPt’ , ‘legacyNormal3’ , ‘legacyFlat1’ , ‘legacyFlat2’ , ‘contrasting’ , ‘flat’ , ‘balanced’ , ‘soft’ , ‘harsh’ , ‘sunrise’ , ‘legacyHarsh2’ , ‘sunset’ , ‘freezing’ , ‘flood’ , ‘legacyHarsh1’ , ‘legacyFlat3’

, 'legacyNormal4' , 'threePt' , 'legacyHarsh4' , 'glow' , 'brightRoom' , 'legacyNormal1' , 'morning' , 'legacyFlat4' , 'chilly' }

rot

Values must be of type <class 'openpyxl.drawing.geometry.SphereCoords' >

tagname = 'lightRig'

class openpyxl.drawing.geometry.**Path2D**(*w=None*, *h=None*, *fill=None*, *stroke=None*, *extrusionOk=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

extrusionOk

Values must be of type <class 'bool' >

fill

Value must be one of { 'lighten' , 'norm' , 'lightenLess' , 'darkenLess' , 'darken' }

h

Values must be of type <class 'float' >

stroke

Values must be of type <class 'bool' >

w

Values must be of type <class 'float' >

class openpyxl.drawing.geometry.**Path2DList**(*path=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

path

Values must be of type <class 'openpyxl.drawing.geometry.Path2D' >

class openpyxl.drawing.geometry.**Point2D**(*x=None*, *y=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

tagname = 'off'

x

Values must be of type <class 'int' >

y

Values must be of type <class 'int' >

class openpyxl.drawing.geometry.**Point3D**(*x=None*, *y=None*, *z=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

tagname = 'anchor'

x

Values must be of type <class 'int' >

y

Values must be of type <class 'int' >

z

Values must be of type <class 'int' >

class openpyxl.drawing.geometry.PositiveSize2D(*cx=None*, *cy=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

cx

Values must be of type <class 'int' >

cy

Values must be of type <class 'int' >

height

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

Dimensions in EMUs

tagname = 'ext'

width

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

class openpyxl.drawing.geometry.PresetGeometry2D(*prst=None*, *avLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

avLst

Values must be of type <class 'openpyxl.drawing.geometry.GeomGuideList' >

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

prst

Value must be one of { 'flowChartDecision' , 'flowChartSort' , 'flowChartMagneticTape' , 'foldedCorner' , 'star5' , 'stripedRightArrow' , 'flowChartManualInput' , 'curvedDownArrow' , 'ellipse' , 'round2SameRect' , 'accentBorderCallout3' , 'hexagon' , 'accentBorderCallout2' , 'swooshArrow' , 'noSmoking' , 'pieWedge' , 'halfFrame' , 'star7' , 'callout3' , 'accentBorderCallout1' , 'actionButtonBackPrevious' , 'moon' , 'flowChartPreparation' , 'irregularSeal1' , 'curvedConnector5' , 'flowChartInputOutput' , 'flowChartMultidocument' , 'downArrowCallout' , 'nonIsoscelesTrapezoid' , 'snipRoundRect' , 'plaque' , 'callout2' , 'flowChartDelay' , 'star4' , 'uturnArrow' , 'curvedLeftArrow' , 'wedgeRectCallout' , 'mathMultiply' , 'frame' , 'upDownArrow' , 'ellipseRibbon' , 'actionButtonMovie' , 'actionButtonBlank' , 'triangle' , 'actionButtonDocument' , 'corner' , 'star32' , 'curvedRightArrow' , 'flowChartOr' , 'flowChartOffpageConnector' , 'arc' , 'pie' , 'leftRightArrowCallout' , 'circularArrow' , 'flowChartMagneticDrum' , 'bentConnector3'

, 'quadArrowCallout', 'flowChartMerge', 'homePlate', 'bentUpArrow', 'sun', 'mathNotEqual', 'roundRect', 'star8', 'ribbon', 'leftRightUpArrow', 'flowChartAlternateProcess', 'flowChartPunchedTape', 'rtTriangle', 'star16', 'squareTabs', 'cloudCallout', 'chartStar', 'mathDivide', 'octagon', 'bevel', 'round1Rect', 'flowChartPredefinedProcess', 'mathMinus', 'flowChartDisplay', 'wedgeRoundRectCallout', 'round2DiagRect', 'ellipseRibbon2', 'flowChartPunchedCard', 'actionButtonSound', 'can', 'flowChartDocument', 'rightBrace', 'accentCallout3', 'rightArrow', 'flowChartMagneticDisk', 'actionButtonHome', 'cloud', 'flowChartProcess', 'notchedRightArrow', 'leftRightRibbon', 'borderCallout2', 'leftRightArrow', 'bentArrow', 'wedgeEllipseCallout', 'flowChartSummingJunction', 'decagon', 'trapezoid', 'diagStripe', 'callout1', 'donut', 'chartPlus', 'star10', 'quadArrow', 'snip1Rect', 'curvedConnector2', 'accentCallout2', 'horizontalScroll', 'upArrowCallout', 'verticalScroll', 'heart', 'wave', 'flowChartOnlineStorage', 'plaqueTabs', 'leftRightCircularArrow', 'flowChartInternalStorage', 'bentConnector5', 'rect', 'actionButtonReturn', 'actionButtonHelp', 'actionButtonEnd', 'irregularSeal2', 'lightningBolt', 'bentConnector2', 'pentagon', 'blockArc', 'lineInv', 'actionButtonForwardNext', 'star6', 'snip2SameRect', 'ribbon2', 'rightBracket', 'flowChartConnector', 'chord', 'chevron', 'downArrow', 'bracketPair', 'upDownArrowCallout', 'curvedConnector4', 'leftArrow', 'borderCallout3', 'curvedUpArrow', 'flowChartExtract', 'line', 'chartX', 'leftCircularArrow', 'upArrow', 'gear6', 'mathPlus', 'leftBracket', 'bracePair', 'smileyFace', 'funnel', 'dodecagon', 'curvedConnector3', 'plus', 'snip2DiagRect', 'diamond', 'heptagon', 'star12', 'flowChartOfflineStorage', 'accentCallout1', 'cornerTabs', 'star24', 'actionButtonBeginning', 'leftUpArrow', 'flowChartManualOperation', 'parallelogram', 'flowChartTerminator', 'gear9', 'borderCallout1', 'straightConnector1', 'cube', 'leftArrowCallout', 'bentConnector4', 'flowChartCollate', 'actionButtonInformation', 'mathEqual', 'teardrop', 'leftBrace', 'rightArrowCallout', 'doubleWave'}

`class openpyxl.drawing.geometry.Scene3D`(*camera=None*, *lightRig=None*, *backdrop=None*, *extLst=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

backdrop

Values must be of type <class 'openpyxl.drawing.geometry.Backdrop'>

camera

Values must be of type <class 'openpyxl.drawing.geometry.Camera'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

lightRig

Values must be of type <class 'openpyxl.drawing.geometry.LightRig'>

`class openpyxl.drawing.geometry.Shape3D`(*z=None*, *extrusionH=None*, *contourW=None*, *prstMaterial=None*, *bevelT=None*, *bevelB=None*, *extrusionClr=None*, *contourClr=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

bevelB

Values must be of type <class 'openpyxl.drawing.geometry.Bevel'>

bevelT

Values must be of type <class 'openpyxl.drawing.geometry.Bevel'>

contourClr

Values must be of type <class 'openpyxl.styles.colors.Color'>

contourW

Values must be of type <class 'int'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

extrusionClr

Values must be of type <class 'openpyxl.styles.colors.Color'>

extrusionH

Values must be of type <class 'int'>

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

prstMaterial

Value must be one of {'translucentPowder', 'legacyPlastic', 'legacyMetal', 'powder', 'dkEdge', 'softEdge', 'matte', 'flat', 'metal', 'warmMatte', 'plastic', 'clear', 'legacyWireframe', 'softmetal', 'legacyMatte'}

z

Values must be of type <class 'openpyxl.descriptors.base.Integer'>

class openpyxl.drawing.geometry.**ShapeStyle**(*lnRef=None*, *fillRef=None*, *effectRef=None*, *fontRef=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

effectRef

Values must be of type <class 'openpyxl.drawing.geometry.StyleMatrixReference'>

fillRef

Values must be of type <class 'openpyxl.drawing.geometry.StyleMatrixReference'>

fontRef

Values must be of type <class 'openpyxl.drawing.geometry.FontReference'>

lnRef

Values must be of type <class 'openpyxl.drawing.geometry.StyleMatrixReference'>

class openpyxl.drawing.geometry.**SphereCoords**(*lat=None*, *lon=None*, *rev=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

lat

Values must be of type <class ‘int’ >

lon

Values must be of type <class ‘int’ >

rev

Values must be of type <class ‘int’ >

tagname = 'sphereCoords'

class openpyxl.drawing.geometry.**StyleMatrixReference**(*idx=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

idx

Values must be of type <class ‘int’ >

class openpyxl.drawing.geometry.**Transform2D**(*rot=None, flipH=None, flipV=None, off=None, ext=None, chOff=None, chExt=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

chExt

Values must be of type <class ‘openpyxl.drawing.geometry.PositiveSize2D’ >

chOff

Values must be of type <class ‘openpyxl.drawing.geometry.Point2D’ >

ext

Values must be of type <class ‘openpyxl.drawing.geometry.PositiveSize2D’ >

flipH

Values must be of type <class ‘bool’ >

flipV

Values must be of type <class ‘bool’ >

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

off

Values must be of type <class ‘openpyxl.drawing.geometry.Point2D’ >

rot

Values must be of type <class ‘int’ >

tagname = 'xfrm'

class openpyxl.drawing.geometry.**Vector3D**(*dx=None, dy=None, dz=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

dx

Values must be of type <class ‘int’ >

dy

Values must be of type <class ‘int’ >

dz

Values must be of type <class ‘int’ >

tagname = 'vector'

openpyxl.drawing.graphic module

class openpyxl.drawing.graphic.**GraphicData**(*uri='http://schemas.openxmlformats.org/drawingml/2006/chart'*, *chart=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

chart

Values must be of type <class ‘openpyxl.drawing.relation.ChartRelation’ >

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

tagname = 'graphicData'

uri

Values must be of type <class ‘str’ >

class openpyxl.drawing.graphic.**GraphicFrame**(*nvGraphicFramePr=None*, *xfrm=None*, *graphic=None*, *macro=None*, *fPublished=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

fPublished

Values must be of type <class ‘bool’ >

graphic

Values must be of type <class ‘openpyxl.drawing.graphic.GraphicObject’ >

macro

Values must be of type <class ‘str’ >

nvGraphicFramePr

Values must be of type <class ‘openpyxl.drawing.graphic.NonVisualGraphicFrame’ >

tagname = 'graphicFrame'

xfrm

Values must be of type <class ‘openpyxl.drawing.xdr.XDRTransform2D’ >

class openpyxl.drawing.graphic.**GraphicFrameLocking**(*noGrp=None*, *noDrilldown=None*, *noSelect=None*, *noChangeAspect=None*, *noMove=None*, *noResize=None*, *extLst=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

noChangeAspect

Values must be of type <class ‘bool’ >

noDrilldown

Values must be of type <class ‘bool’ >

noGrp

Values must be of type <class ‘bool’ >

noMove

Values must be of type <class ‘bool’ >

noResize

Values must be of type <class ‘bool’ >

noSelect

Values must be of type <class ‘bool’ >

class openpyxl.drawing.graphic.**GraphicObject**(*graphicData=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

graphicData

Values must be of type <class ‘openpyxl.drawing.graphic.GraphicData’ >

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

tagname = 'graphic'

class openpyxl.drawing.graphic.**GroupShape**(*nvGrpSpPr=None*, *grpSpPr=None*, *pic=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

grpSpPr

Values must be of type <class ‘openpyxl.drawing.properties.GroupShapeProperties’ >

nonVisualProperties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

nvGrpSpPr

Values must be of type <class ‘openpyxl.drawing.properties.NonVisualGroupShape’ >

pic

Values must be of type <class ‘openpyxl.drawing.picture.PictureFrame’ >

visualProperties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

class openpyxl.drawing.graphic.**NonVisualGraphicFrame**(*cNvPr=None*, *cNvGraphicFramePr=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

cNvGraphicFramePr

Values must be of type <class ‘openpyxl.drawing.graphic.NonVisualGraphicFrameProperties’ >

cNvPr

Values must be of type <class ‘openpyxl.drawing.properties.NonVisualDrawingProps’ >

tagname = 'nvGraphicFramePr'

class openpyxl.drawing.graphic.**NonVisualGraphicFrameProperties**(*graphicFrameLocks=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

graphicFrameLocks

Values must be of type <class ‘openpyxl.drawing.graphic.GraphicFrameLocking’ >

tagname = 'cNvGraphicFramePr'

openpyxl.drawing.image module

class openpyxl.drawing.image.**Image**(*img*)

基类：object

Image in a spreadsheet

anchor = 'A1'

path

openpyxl.drawing.line module

class openpyxl.drawing.line.**DashStop**(*d=0*, *sp=0*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

d

Values must be of type <class ‘int’ >

length

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

sp

Values must be of type <class 'int' >

space

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

tagname = 'ds'

class openpyxl.drawing.line.**DashStopList**(*ds=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

ds

A sequence (list or tuple) that may only contain objects of the declared type

class openpyxl.drawing.line.**LineEndProperties**(*type=None*, *w=None*, *len=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

len

Value must be one of { 'med' , 'lg' , 'sm' }

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

tagname = 'end'

type

Value must be one of { 'none' , 'stealth' , 'arrow' , 'oval' , 'diamond' , 'triangle' }

w

Value must be one of { 'med' , 'lg' , 'sm' }

class openpyxl.drawing.line.**LineProperties**(*w=None*, *cap=None*, *cmpd=None*, *algn=None*, *noFill=None*, *solidFill=None*, *gradFill=None*, *pattFill=None*, *prstDash=None*, *custDash=None*, *round=None*, *bevel=None*, *miter=None*, *headEnd=None*, *tailEnd=None*, *extLst=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

algn

Value must be one of { 'ctr' , 'in' }

bevel

Values must be of type <class 'bool' >

cap

Value must be one of { 'rnd' , 'sq' , 'flat' }

cmpd

Value must be one of { 'sng' , 'tri' , 'thickThin' , 'dbl' , 'thinThick' }

custDash

Values must be of type <class 'openpyxl.drawing.line.DashStop' >

dashStyle

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

gradFill

Values must be of type <class ‘openpyxl.drawing.fill.GradientFillProperties’ >

headEnd

Values must be of type <class ‘openpyxl.drawing.line.LineEndProperties’ >

miter

Values must be of type <class ‘int’ >

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

noFill

Values must be of type <class ‘bool’ >

pattFill

Values must be of type <class ‘openpyxl.drawing.fill.PatternFillProperties’ >

prstDash

Value must be one of { ‘lgDashDot’ , ‘dashDot’ , ‘sysDot’ , ‘sysDash’ , ‘sysDashDot’ , ‘lgDash’ , ‘lgDashDotDot’ , ‘dot’ , ‘sysDashDotDot’ , ‘solid’ , ‘dash’ }

round

Values must be of type <class ‘bool’ >

solidFill

Values must be of type <class ‘openpyxl.drawing.colors.ColorChoice’ >

tagname = 'ln'

tailEnd

Values must be of type <class ‘openpyxl.drawing.line.LineEndProperties’ >

w

Values must be of type <class ‘float’ >

width

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

openpyxl.drawing.picture module

class openpyxl.drawing.picture.**NonVisualPictureProperties**(*preferRelativeResize=None, picLocks=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

picLocks

Values must be of type <class 'openpyxl.drawing.picture.PictureLocking'>

preferRelativeResize

Values must be of type <class 'bool'>

tagname = 'cNvPicPr'

class openpyxl.drawing.picture.**PictureFrame**(*macro=None, fPublished=None, nvPicPr=None, blipFill=None, spPr=None, style=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

blipFill

Values must be of type <class 'openpyxl.drawing.fill.BlipFillProperties'>

fPublished

Values must be of type <class 'bool'>

graphicalProperties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

macro

Values must be of type <class 'str'>

nvPicPr

Values must be of type <class 'openpyxl.drawing.picture.PictureNonVisual'>

spPr

Values must be of type <class 'openpyxl.chart.shapes.GraphicalProperties'>

style

Values must be of type <class 'openpyxl.drawing.geometry.ShapeStyle'>

tagname = 'pic'

class openpyxl.drawing.picture.**PictureLocking**(*noCrop=None, noGrp=None, noSelect=None, noRot=None, noChangeAspect=None, noMove=None, noResize=None, noEditPoints=None, noAdjustHandles=None, noChangeArrowheads=None, noChangeShapeType=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

noAdjustHandles

Values must be of type <class 'bool'>

noChangeArrowheads

Values must be of type <class 'bool'>

noChangeAspect

Values must be of type <class 'bool'>

noChangeShapeType

Values must be of type <class 'bool'>

noCrop

Values must be of type <class 'bool'>

noEditPoints

Values must be of type <class 'bool'>

noGrp

Values must be of type <class 'bool'>

noMove

Values must be of type <class 'bool'>

noResize

Values must be of type <class 'bool'>

noRot

Values must be of type <class 'bool'>

noSelect

Values must be of type <class 'bool'>

tagname = 'picLocks'

class openpyxl.drawing.picture.**PictureNonVisual**(*cNvPr=None*, *cNvPicPr=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

cNvPicPr

Values must be of type <class 'openpyxl.drawing.picture.NonVisualPictureProperties'>

cNvPr

Values must be of type <class 'openpyxl.drawing.properties.NonVisualDrawingProps'>

tagname = 'nvPicPr'

openpyxl.drawing.properties module

class openpyxl.drawing.properties.**GroupLocking**(*noGrp=None*, *noUngrp=None*, *noSelect=None*, *noRot=None*, *noChangeAspect=None*, *noChangeArrowheads=None*, *noMove=None*, *noResize=None*, *noEditPoints=None*, *noAdjustHandles=None*, *noChangeShapeType=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

noAdjustHandles

Values must be of type <class ‘bool’ >

noChangeArrowheads

Values must be of type <class ‘bool’ >

noChangeAspect

Values must be of type <class ‘bool’ >

noChangeShapeType

Values must be of type <class ‘bool’ >

noEditPoints

Values must be of type <class ‘bool’ >

noGrp

Values must be of type <class ‘bool’ >

noMove

Values must be of type <class ‘bool’ >

noResize

Values must be of type <class ‘bool’ >

noRot

Values must be of type <class ‘bool’ >

noSelect

Values must be of type <class ‘bool’ >

noUngrp

Values must be of type <class ‘bool’ >

tagname = 'grpSpLocks'

class openpyxl.drawing.properties.GroupShapeProperties(*bwMode=None*, *xfrm=None*, *scene3d=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

bwMode

Value must be one of { ‘black’ , ‘blackGray’ , ‘clr’ , ‘blackWhite’ , ‘auto’ , ‘gray’ , ‘ltGray’ , ‘hidden’ , ‘invGray’ , ‘white’ , ‘grayWhite’ }

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

scene3d

Values must be of type <class ‘openpyxl.drawing.geometry.Scene3D’ >

tagname = 'grpSpPr'

xfrm

Values must be of type <class ‘openpyxl.drawing.geometry.GroupTransform2D’ >

class openpyxl.drawing.properties.NonVisualDrawingProps(*id=None*, *name=None*, *descr=None*, *hidden=None*, *title=None*, *hlinkClick=None*, *hlinkHover=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

descr

Values must be of type <class ‘str’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

hidden

Values must be of type <class ‘bool’ >

hlinkClick

Values must be of type <class ‘openpyxl.drawing.text.Hyperlink’ >

hlinkHover

Values must be of type <class ‘openpyxl.drawing.text.Hyperlink’ >

id

Values must be of type <class ‘int’ >

name

Values must be of type <class ‘str’ >

tagname = 'cNvPr'

title

Values must be of type <class ‘str’ >

class openpyxl.drawing.properties.**NonVisualDrawingShapeProps**(*spLocks=None, txBox=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

spLocks

Values must be of type <class ‘openpyxl.drawing.properties.GroupLocking’ >

tagname = 'cNvSpPr'

txBax

Values must be of type <class ‘bool’ >

class openpyxl.drawing.properties.**NonVisualGroupDrawingShapeProps**(*grpSpLocks=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

grpSpLocks

Values must be of type <class ‘openpyxl.drawing.properties.GroupLocking’ >

tagname = 'cNvGrpSpPr'

class openpyxl.drawing.properties.**NonVisualGroupShape**(*cNvPr=None, cNvGrpSpPr=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

cNvGrpSpPr

Values must be of type <class ‘openpyxl.drawing.properties.NonVisualGroupDrawingShapeProps’ >

cNvPr

Values must be of type <class ‘openpyxl.drawing.properties.NonVisualDrawingProps’ >

tagname = 'nvGrpSpPr'

openpyxl.drawing.relation module

class openpyxl.drawing.relation.**ChartRelation**(*id*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

id

Values must be of type <class ‘str’ >

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/chart'

tagname = 'chart'

openpyxl.drawing.spreadsheet_drawing module

class openpyxl.drawing.spreadsheet_drawing.**AbsoluteAnchor**(*pos=None*, *ext=None*, ***kw*)

基类：openpyxl.drawing.spreadsheet_drawing._AnchorBase

clientData

Values must be of type <class 'openpyxl.drawing.spreadsheet_drawing.AnchorClientData'>

contentPart

Values must be of type <class 'str'>

cxnSp

Values must be of type <class 'openpyxl.drawing.connector.Shape'>

ext

Values must be of type <class 'openpyxl.drawing.xdr.XDRPositiveSize2D'>

graphicFrame

Values must be of type <class 'openpyxl.drawing.graphic.GraphicFrame'>

grpSp

Values must be of type <class 'openpyxl.drawing.graphic.GroupShape'>

pic

Values must be of type <class 'openpyxl.drawing.picture.PictureFrame'>

pos

Values must be of type <class 'openpyxl.drawing.xdr.XDRPoint2D'>

sp

Values must be of type <class 'openpyxl.drawing.connector.Shape'>

tagname = 'absoluteAnchor'

class openpyxl.drawing.spreadsheet_drawing.**AnchorClientData**(*fLocksWithSheet=None*, *fPrintsWithSheet=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

fLocksWithSheet

Values must be of type <class 'bool'>

fPrintsWithSheet

Values must be of type <class 'bool'>

class openpyxl.drawing.spreadsheet_drawing.**AnchorMarker**(*col=0*, *colOff=0*, *row=0*, *rowOff=0*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

col

Values must be of type <class 'int'>

colOff
Values must be of type <class ‘int’ >

row
Values must be of type <class ‘int’ >

rowOff
Values must be of type <class ‘int’ >

tagname = 'marker'

class openpyxl.drawing.spreadsheet_drawing.**OneCellAnchor**(*_from=None*, *ext=None*, ***kw*)
基类：openpyxl.drawing.spreadsheet_drawing._AnchorBase

clientData
Values must be of type <class ‘openpyxl.drawing.spreadsheet_drawing.AnchorClientData’ >

contentPart
Values must be of type <class ‘str’ >

cxnSp
Values must be of type <class ‘openpyxl.drawing.connector.Shape’ >

ext
Values must be of type <class ‘openpyxl.drawing.xdr.XDRPositiveSize2D’ >

graphicFrame
Values must be of type <class ‘openpyxl.drawing.graphic.GraphicFrame’ >

grpSp
Values must be of type <class ‘openpyxl.drawing.graphic.GroupShape’ >

pic
Values must be of type <class ‘openpyxl.drawing.picture.PictureFrame’ >

sp
Values must be of type <class ‘openpyxl.drawing.connector.Shape’ >

tagname = 'oneCellAnchor'

class openpyxl.drawing.spreadsheet_drawing.**SpreadsheetDrawing**(*twoCellAnchor=()*, *oneCellAnchor=()*, *absoluteAnchor=()*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

PartName = '/xl/drawings/drawing{0}.xml'

absoluteAnchor
A sequence (list or tuple) that may only contain objects of the declared type

mime_type = 'application/vnd.openxmlformats-officedocument.drawing+xml'

oneCellAnchor

A sequence (list or tuple) that may only contain objects of the declared type

path

tagname = 'wsDr'

twoCellAnchor

A sequence (list or tuple) that may only contain objects of the declared type

class openpyxl.drawing.spreadsheet_drawing.**TwoCellAnchor**(*editAs=None*, *_from=None*, *to=None*, ***kw*)

基类: openpyxl.drawing.spreadsheet_drawing._AnchorBase

clientData

Values must be of type <class 'openpyxl.drawing.spreadsheet_drawing.AnchorClientData'>

contentPart

Values must be of type <class 'str'>

cxnSp

Values must be of type <class 'openpyxl.drawing.connector.Shape'>

editAs

Value must be one of {'absolute', 'twoCell', 'oneCell'}

graphicFrame

Values must be of type <class 'openpyxl.drawing.graphic.GraphicFrame'>

grpSp

Values must be of type <class 'openpyxl.drawing.graphic.GroupShape'>

pic

Values must be of type <class 'openpyxl.drawing.picture.PictureFrame'>

sp

Values must be of type <class 'openpyxl.drawing.connector.Shape'>

tagname = 'twoCellAnchor'

to

Values must be of type <class 'openpyxl.drawing.spreadsheet_drawing.AnchorMarker'>

openpyxl.drawing.text module

class openpyxl.drawing.text.**AutonumberBullet**(*type=None*, *startAt=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

startAt

Values must be of type <class 'int'>

type

Value must be one of { 'circleNumDbPlain' , 'hindiAlpha1Period' , 'ea1JpnChsDbPeriod' , 'arabicDbPlain' , 'alphaUcParenBoth' , 'ea1JpnKorPeriod' , 'romanLcParenR' , 'alphaLcPeriod' , 'romanUcPeriod' , 'romanUcParenR' , 'romanLcPeriod' , 'thaiNumParenR' , 'arabicParenBoth' , 'romanLcParenBoth' , 'circleNumWdBlackPlain' , 'arabicPeriod' , 'hindiAlphaPeriod' , 'alphaLcParenBoth' , 'hebrew2Minus' , 'circleNumWdWhitePlain' , 'alphaUcParenR' , 'alphaLcParenR' , 'ea1ChtPeriod' , 'thaiNumParenBoth' , 'thaiAlphaParenBoth' , 'arabic2Minus' , 'arabic1Minus' , 'thaiAlphaParenR' , 'hindiNumPeriod' , 'arabicDbPeriod' , 'ea1JpnKorPlain' , 'thaiNumPeriod' , 'thaiAlphaPeriod' , 'alphaUcPeriod' , 'arabicPlain' , 'arabicParenR' , 'ea1ChtPlain' , 'ea1ChsPlain' , 'romanUcParenBoth' , 'hindiNumParenR' , 'ea1ChsPeriod' }

class openpyxl.drawing.text.CharacterProperties(*kumimoji=None*, *lang=None*, *altLang=None*, *sz=None*, *b=None*, *i=None*, *u=None*, *strike=None*, *kern=None*, *cap=None*, *spc=None*, *normalizeH=None*, *baseline=None*, *noProof=None*, *dirty=None*, *err=None*, *smtClean=None*, *smtId=None*, *bmk=None*, *ln=None*, *highlight=None*, *latin=None*, *ea=None*, *cs=None*, *sym=None*, *hlinkClick=None*, *hlinkMouseOver=None*, *rtl=None*, *extLst=None*, *noFill=None*, *solidFill=None*, *gradFill=None*, *blipFill=None*, *pattFill=None*, *grpFill=None*, *effectLst=None*, *effectDag=None*, *uLnTx=None*, *uLn=None*, *uFillTx=None*, *uFill=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

altLang

Values must be of type <class 'str' >

b

Values must be of type <class 'bool' >

baseline

Values must be of type <class 'int' >

blipFill

Values must be of type <class 'openpyxl.drawing.fill.BlipFillProperties' >

bmk

Values must be of type <class 'str' >

cap

Value must be one of { 'small' , 'all' }

cs
Values must be of type <class 'openpyxl.drawing.text.Font'>

dirty
Values must be of type <class 'bool'>

ea
Values must be of type <class 'openpyxl.drawing.text.Font'>

effectDag
Values must be of type <class 'openpyxl.drawing.effect.EffectContainer'>

effectLst
Values must be of type <class 'openpyxl.drawing.effect.EffectList'>

err
Values must be of type <class 'bool'>

extLst
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

gradFill
Values must be of type <class 'openpyxl.drawing.fill.GradientFillProperties'>

grpFill
Values must be of type <class 'bool'>

highlight
Values must be of type <class 'openpyxl.styles.colors.Color'>

hlinkClick
Values must be of type <class 'openpyxl.drawing.text.Hyperlink'>

hlinkMouseOver
Values must be of type <class 'openpyxl.drawing.text.Hyperlink'>

i
Values must be of type <class 'bool'>

kern
Values must be of type <class 'int'>

kumimoji
Values must be of type <class 'bool'>

lang
Values must be of type <class 'str'>

latin
Values must be of type <class 'openpyxl.drawing.text.Font'>

ln

Values must be of type <class 'openpyxl.drawing.line.LineProperties' >

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

noFill

Values must be of type <class 'bool' >

noProof

Values must be of type <class 'bool' >

normalizeH

Values must be of type <class 'bool' >

pattFill

Values must be of type <class 'openpyxl.drawing.fill.PatternFillProperties' >

rtl

Values must be of type <class 'bool' >

smtClean

Values must be of type <class 'bool' >

smtId

Values must be of type <class 'int' >

solidFill

Values must be of type <class 'openpyxl.drawing.colors.ColorChoice' >

spc

Values must be of type <class 'int' >

strike

Value must be one of { 'dblStrike' , 'noStrike' , 'sngStrike' }

sym

Values must be of type <class 'openpyxl.drawing.text.Font' >

sz

Values must be of type <class 'float' >

tagname = 'defRPr'

u

Value must be one of { 'dashLong' , 'dotDotDashHeavy' , 'dashHeavy' , 'sng' , 'heavy' , 'dotDashHeavy' , 'wavyDbl' , 'wavy' , 'dashLongHeavy' , 'dotted' , 'dottedHeavy' , 'words' , 'dotDash' , 'dbl' , 'wavyHeavy' , 'dotDotDash' , 'dash' }

uFill

Values must be of type <class 'bool' >

uFillTx

Values must be of type <class 'bool'>

uLn

Values must be of type <class 'openpyxl.drawing.line.LineProperties'>

uLnTx

Values must be of type <class 'bool'>

class openpyxl.drawing.text.**EmbeddedWAVAudioFile**(*name=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

name

Values must be of type <class 'str'>

class openpyxl.drawing.text.**Font**(*typeface=None*, *panose=None*, *pitchFamily=None*, *charset=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

charset

Values must be of type <class 'int'>

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

panose

pitchFamily

Values must be of type <class 'float'>

tagname = 'latin'

typeface

Values must be of type <class 'str'>

class openpyxl.drawing.text.**GeomGuide**(*name=None*, *fmla=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

fmla

Values must be of type <class 'str'>

name

Values must be of type <class 'str'>

class openpyxl.drawing.text.**GeomGuideList**(*gd=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

gd

A sequence (list or tuple) that may only contain objects of the declared type

class openpyxl.drawing.text.**Hyperlink**(*invalidUrl=None*, *action=None*, *tgtFrame=None*, *tooltip=None*, *history=None*, *highlightClick=None*, *endSnd=None*, *snd=None*, *extLst=None*, *id=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

action
Values must be of type <class 'str'>

endSnd
Values must be of type <class 'bool'>

extLst
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

highlightClick
Values must be of type <class 'bool'>

history
Values must be of type <class 'bool'>

id
Values must be of type <class 'str'>

invalidUrl
Values must be of type <class 'str'>

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

snd
Values must be of type <class 'openpyxl.drawing.text.EmbeddedWAVAudioFile'>

tagname = 'hlinkClick'

tgtFrame
Values must be of type <class 'str'>

tooltip
Values must be of type <class 'str'>

class openpyxl.drawing.text.**LineBreak**(*rPr=None*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

rPr
Values must be of type <class 'openpyxl.drawing.text.CharacterProperties'>

tagname = 'br'

class openpyxl.drawing.text.**ListStyle**(*defPPr=None, lvl1pPr=None, lvl2pPr=None, lvl3pPr=None, lvl4pPr=None, lvl5pPr=None, lvl6pPr=None, lvl7pPr=None, lvl8pPr=None, lvl9pPr=None, extLst=None*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

defPPr
Values must be of type <class 'openpyxl.drawing.text.ParagraphProperties'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

lvl1pPr

Values must be of type <class 'openpyxl.drawing.text.ParagraphProperties'>

lvl2pPr

Values must be of type <class 'openpyxl.drawing.text.ParagraphProperties'>

lvl3pPr

Values must be of type <class 'openpyxl.drawing.text.ParagraphProperties'>

lvl4pPr

Values must be of type <class 'openpyxl.drawing.text.ParagraphProperties'>

lvl5pPr

Values must be of type <class 'openpyxl.drawing.text.ParagraphProperties'>

lvl6pPr

Values must be of type <class 'openpyxl.drawing.text.ParagraphProperties'>

lvl7pPr

Values must be of type <class 'openpyxl.drawing.text.ParagraphProperties'>

lvl8pPr

Values must be of type <class 'openpyxl.drawing.text.ParagraphProperties'>

lvl9pPr

Values must be of type <class 'openpyxl.drawing.text.ParagraphProperties'>

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

tagname = 'lstStyle'

class openpyxl.drawing.text.**Paragraph**(*pPr=None*, *endParaRPr=None*, *r=None*, *br=None*, *fld=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

br

Values must be of type <class 'openpyxl.drawing.text.LineBreak'>

endParaRPr

Values must be of type <class 'openpyxl.drawing.text.CharacterProperties'>

fld

Values must be of type <class 'openpyxl.drawing.text.TextField'>

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

pPr

Values must be of type <class 'openpyxl.drawing.text.ParagraphProperties'>

properties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

r

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'p'

text

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

class openpyxl.drawing.text.**ParagraphProperties**(*marL=None, marR=None, lvl=None, indent=None, algn=None, defTabSz=None, rtl=None, eaLnBrk=None, fontAlgn=None, latinLnBrk=None, hangingPunct=None, lnSpc=None, spcBef=None, spcAft=None, tabLst=None, defRPr=None, extLst=None, buClrTx=None, buClr=None, buSzTx=None, buSzPct=None, buSzPts=None, buFontTx=None, buFont=None, buNone=None, buAutoNum=None, buChar=None, buBlip=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

algn

Value must be one of { ‘dist’ , ‘r’ , ‘justLow’ , ‘just’ , ‘ctr’ , ‘thaiDist’ , ‘l’ }

buAutoNum

Values must be of type <class ‘bool’ >

buBlip

Values must be of type <class ‘openpyxl.drawing.fill.Blip’ >

buChar

Values must be of type <class ‘str’ >

buClr

Values must be of type <class ‘openpyxl.styles.colors.Color’ >

buClrTx

Values must be of type <class ‘bool’ >

buFont

Values must be of type <class ‘openpyxl.drawing.text.Font’ >

buFontTx
Values must be of type <class 'bool'>

buNone
Values must be of type <class 'bool'>

buSzPct
Values must be of type <class 'int'>

buSzPts
Values must be of type <class 'int'>

buSzTx
Values must be of type <class 'bool'>

defRPr
Values must be of type <class 'openpyxl.drawing.text.CharacterProperties'>

defTabSz
Values must be of type <class 'int'>

eaLnBrk
Values must be of type <class 'bool'>

extLst
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

fontAlgn
Value must be one of {'base', 'b', 'auto', 'ctr', 't'}

hangingPunct
Values must be of type <class 'bool'>

indent
Values must be of type <class 'int'>

latinLnBrk
Values must be of type <class 'bool'>

lnSpc
Values must be of type <class 'openpyxl.drawing.text.Spacing'>

lvl
Values must be of type <class 'int'>

marL
Values must be of type <class 'int'>

marR
Values must be of type <class 'int'>

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

rtl

Values must be of type <class 'bool'>

spcAft

Values must be of type <class 'openpyxl.drawing.text.Spacing'>

spcBef

Values must be of type <class 'openpyxl.drawing.text.Spacing'>

tabLst

Values must be of type <class 'openpyxl.drawing.text.TabStopList'>

tagname = 'pPr'

class openpyxl.drawing.text.**PresetTextShape**(*prst=None*, *avLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

avLst

Values must be of type <class 'openpyxl.drawing.text.GeomGuideList'>

prst

Values must be of type <openpyxl.descriptors.base.Set object at 0x7f52416bf8d0>

class openpyxl.drawing.text.**RegularTextRun**(*rPr=None*, *t="*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

properties

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

rPr

Values must be of type <class 'openpyxl.drawing.text.CharacterProperties'>

t

Values must be of type <class 'str'>

tagname = 'r'

value

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`class openpyxl.drawing.text.RichTextProperties`(*rot=None, spcFirstLastPara=None, vertOverflow=None, horzOverflow=None, vert=None, wrap=None, lIns=None, tIns=None, rIns=None, bIns=None, numCol=None, spcCol=None, rtlCol=None, fromWordArt=None, anchor=None, anchorCtr=None, forceAA=None, upright=None, compatLnSpc=None, prstTxWarp=None, scene3d=None, extLst=None, noAutofit=None, normAutofit=None, spAutoFit=None, flatTx=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`anchor`

Value must be one of { ‘dist’ , ‘b’ , ‘just’ , ‘ctr’ , ‘t’ }

`anchorCtr`

Values must be of type <class ‘bool’ >

`bIns`

Values must be of type <class ‘int’ >

`compatLnSpc`

Values must be of type <class ‘bool’ >

`extLst`

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

`flatTx`

Values must be of type <class ‘int’ >

`forceAA`

Values must be of type <class ‘bool’ >

`fromWordArt`

Values must be of type <class ‘bool’ >

`horzOverflow`

Value must be one of { ‘overflow’ , ‘clip’ }

`lIns`

Values must be of type <class ‘int’ >

`namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'`

`noAutofit`

Values must be of type <class ‘bool’ >

`normAutofit`

Values must be of type <class ‘bool’ >

numCol
Values must be of type <class 'int' >

prstTxWarp
Values must be of type <class 'openpyxl.drawing.text.PresetTextShape' >

rIns
Values must be of type <class 'int' >

rot
Values must be of type <class 'int' >

rtlCol
Values must be of type <class 'bool' >

scene3d
Values must be of type <class 'openpyxl.drawing.geometry.Scene3D' >

spAutoFit
Values must be of type <class 'bool' >

spcCol
Values must be of type <class 'int' >

spcFirstLastPara
Values must be of type <class 'bool' >

tIns
Values must be of type <class 'int' >

tagname = 'bodyPr'

upright
Values must be of type <class 'bool' >

vert
Value must be one of { 'wordArtVert' , 'horz' , 'eaVert' , 'vert' , 'mongolianVert' , 'vert270' , 'wordArtVertRtl' }

vertOverflow
Value must be one of { 'overflow' , 'ellipsis' , 'clip' }

wrap
Value must be one of { 'square' , 'none' }

class openpyxl.drawing.text.Spacing(*spcPct=None*, *spcPts=None*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

spcPct
Values must be of type <class 'int' >

spcPts

Values must be of type <class 'int' >

class openpyxl.drawing.text.**TabStop**(*pos=None*, *algn=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

algn

Values must be of type <openpyxl.descriptors.base.Set object at 0x7f52416b5490>

pos

Values must be of type <class 'openpyxl.descriptors.base.Integer' >

class openpyxl.drawing.text.**TabStopList**(*tab=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

tab

Values must be of type <class 'openpyxl.drawing.text.TabStop' >

class openpyxl.drawing.text.**TextField**(*id=None*, *type=None*, *rPr=None*, *pPr=None*, *t=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

id

Values must be of type <class 'str' >

pPr

Values must be of type <class 'openpyxl.drawing.text.ParagraphProperties' >

rPr

Values must be of type <class 'openpyxl.drawing.text.CharacterProperties' >

t

Values must be of type <class 'str' >

type

Values must be of type <class 'str' >

class openpyxl.drawing.text.**TextNormalAutofit**(*fontScale=None*, *lnSpcReduction=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

fontScale

Values must be of type <class 'int' >

lnSpcReduction

Values must be of type <class 'int' >

openpyxl.drawing.xdr module

Spreadsheet Drawing has some copies of Drawing ML elements

class openpyxl.drawing.xdr.**XDRPoint2D**(*x=None*, *y=None*)

基类：*openpyxl.drawing.geometry.Point2D*

`namespace = None`

`x`

Values must be of type <class 'int' >

`y`

Values must be of type <class 'int' >

`class openpyxl.drawing.xdr.XDRPositiveSize2D(`*cx=None*, *cy=None*`)`

基类: *openpyxl.drawing.geometry.PositiveSize2D*

`cx`

Values must be of type <class 'int' >

`cy`

Values must be of type <class 'int' >

`namespace = None`

`class openpyxl.drawing.xdr.XDRTransform2D(`*rot=None*, *flipH=None*, *flipV=None*, *off=None*, *ext=None*, *chOff=None*, *chExt=None*`)`

基类: *openpyxl.drawing.geometry.Transform2D*

`chExt`

Values must be of type <class 'openpyxl.drawing.geometry.PositiveSize2D' >

`chOff`

Values must be of type <class 'openpyxl.drawing.geometry.Point2D' >

`ext`

Values must be of type <class 'openpyxl.drawing.geometry.PositiveSize2D' >

`flipH`

Values must be of type <class 'bool' >

`flipV`

Values must be of type <class 'bool' >

`namespace = None`

`off`

Values must be of type <class 'openpyxl.drawing.geometry.Point2D' >

`rot`

Values must be of type <class 'int' >

openpyxl.formatting package

Submodules

openpyxl.formatting.formatting module

`class openpyxl.formatting.formatting.`**`ConditionalFormatting`**(*sqref=()*, *pivot=None*, *cfRule=()*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`cells`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`cfRule`

A sequence (list or tuple) that may only contain objects of the declared type

`pivot`

Values must be of type <class 'bool' >

`rules`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`sqref`

Values must be of type <class 'openpyxl.worksheet.cell_range.MultiCellRange' >

`tagname = 'conditionalFormatting'`

`class openpyxl.formatting.formatting.`**`ConditionalFormattingList`**

基类：`object`

Conditional formatting rules.

`add`(*range_string*, *cfRule*)

Add a rule such as ColorScaleRule, FormulaRule or CellIsRule

The priority will be added automatically.

openpyxl.formatting.rule module

`openpyxl.formatting.rule.`**`CellIsRule`**(*operator=None*, *formula=None*, *stopIfTrue=None*, *font=None*, *border=None*, *fill=None*)

Conditional formatting rule based on cell contents.

`class openpyxl.formatting.rule.`**`ColorScale`**(*cfvo=None*, *color=None*)

基类：*openpyxl.formatting.rule.RuleType*

`color`

A sequence (list or tuple) that may only contain objects of the declared type

`tagname = 'colorScale'`

`openpyxl.formatting.rule.ColorScaleRule`(*start_type=None, start_value=None, start_color=None, mid_type=None, mid_value=None, mid_color=None, end_type=None, end_value=None, end_color=None*)

Backwards compatibility

class `openpyxl.formatting.rule.DataBar`(*minLength=None, maxLength=None, showValue=None, cfvo=None, color=None*)

基类：*openpyxl.formatting.rule.RuleType*

color

Values must be of type <class 'openpyxl.styles.colors.Color' >

maxLength

Values must be of type <class 'int' >

minLength

Values must be of type <class 'int' >

showValue

Values must be of type <class 'bool' >

tagname = 'dataBar'

`openpyxl.formatting.rule.DataBarRule`(*start_type=None, start_value=None, end_type=None, end_value=None, color=None, showValue=None, minLength=None, maxLength=None*)

class `openpyxl.formatting.rule.FormatObject`(*type, val=None, gte=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

gte

Values must be of type <class 'bool' >

tagname = 'cfvo'

type

Value must be one of { 'percentile' , 'min' , 'percent' , 'max' , 'num' , 'formula' }

val

Values must be of type <class 'float' >

`openpyxl.formatting.rule.FormulaRule`(*formula=None, stopIfTrue=None, font=None, border=None, fill=None*)

Conditional formatting with custom differential style

class `openpyxl.formatting.rule.IconSet`(*iconSet=None, showValue=None, percent=None, reverse=None, cfvo=None*)

基类：*openpyxl.formatting.rule.RuleType*

iconSet

Value must be one of { ‘3Arrows’ , ‘3ArrowsGray’ , ‘3Symbols’ , ‘4Rating’ , ‘5Quarters’ , ‘4RedToBlack’ , ‘5ArrowsGray’ , ‘3Signs’ , ‘4ArrowsGray’ , ‘5Rating’ , ‘3Symbols2’ , ‘3TrafficLights1’ , ‘4TrafficLights’ , ‘5Arrows’ , ‘4Arrows’ , ‘3Flags’ , ‘3TrafficLights2’ }

percent

Values must be of type <class ‘bool’ >

reverse

Values must be of type <class ‘bool’ >

showValue

Values must be of type <class ‘bool’ >

tagname = 'iconSet'

openpyxl.formatting.rule.**IconSetRule**(*icon_style=None*, *type=None*, *values=None*, *showValue=None*, *percent=None*, *reverse=None*)

Convenience function for creating icon set rules

class openpyxl.formatting.rule.**Rule**(*type*, *dxfId=None*, *priority=0*, *stopIfTrue=None*, *aboveAverage=None*, *percent=None*, *bottom=None*, *operator=None*, *text=None*, *timePeriod=None*, *rank=None*, *stdDev=None*, *equalAverage=None*, *formula=()*, *colorScale=None*, *dataBar=None*, *iconSet=None*, *extLst=None*, *dxf=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

aboveAverage

Values must be of type <class ‘bool’ >

bottom

Values must be of type <class ‘bool’ >

colorScale

Values must be of type <class ‘openpyxl.formatting.rule.ColorScale’ >

dataBar

Values must be of type <class ‘openpyxl.formatting.rule.DataBar’ >

dxf

Values must be of type <class ‘openpyxl.styles.differential.DifferentialStyle’ >

dxfId

Values must be of type <class ‘int’ >

equalAverage

Values must be of type <class ‘bool’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

formula
A sequence (list or tuple) that may only contain objects of the declared type

iconSet
Values must be of type <class ‘openpyxl.formatting.rule.IconSet’ >

operator
Value must be one of {‘beginsWith’, ‘lessThanOrEqual’, ‘notBetween’, ‘greaterThanOrEqual’, ‘lessThan’, ‘endsWith’, ‘greaterThan’, ‘equal’, ‘notContains’, ‘between’, ‘containsText’, ‘notEqual’ }

percent
Values must be of type <class ‘bool’ >

priority
Values must be of type <class ‘int’ >

rank
Values must be of type <class ‘int’ >

stdDev
Values must be of type <class ‘int’ >

stopIfTrue
Values must be of type <class ‘bool’ >

tagname = 'cfRule'

text
Values must be of type <class ‘str’ >

timePeriod
Value must be one of { ‘thisWeek’ , ‘last7Days’ , ‘tomorrow’ , ‘yesterday’ , ‘lastWeek’ , ‘nextMonth’ , ‘thisMonth’ , ‘lastMonth’ , ‘nextWeek’ , ‘today’ }

type
Value must be one of { ‘uniqueValues’ , ‘top10’ , ‘notContainsBlanks’ , ‘aboveAverage’ , ‘containsErrors’ , ‘duplicateValues’ , ‘endsWith’ , ‘colorScale’ , ‘timePeriod’ , ‘expression’ , ‘dataBar’ , ‘containsText’ , ‘iconSet’ , ‘cellIs’ , ‘containsBlanks’ , ‘notContainsText’ , ‘beginsWith’ , ‘notContainsErrors’ }

class openpyxl.formatting.rule.**RuleType**
基类：*openpyxl.descriptors.serialisable.Serialisable*

cfvo
A sequence (list or tuple) that may only contain objects of the declared type

class openpyxl.formatting.rule.**ValueDescriptor**(**args*, ***kw*)
基类：*openpyxl.descriptors.base.Float*

Expected type depends upon type attribue of parent :-(

Most values should be numeric BUT they can also be cell references

openpyxl.formula package

Submodules

openpyxl.formula.tokenizer module

This module contains a tokenizer for Excel formulae.

The tokenizer is based on the Javascript tokenizer found at http://ewbi.blogs.com/develops/2004/12/excel_formula_p.html written by Eric Bachtal

class openpyxl.formula.tokenizer.**Token**(*value*, *type_*, *subtype=''*)

基类：object

A token in an Excel formula.

Tokens have three attributes:

- *value*: The string value parsed that led to this token
- *type*: A string identifying the type of token
- ***subtype*: A string identifying subtype of the token (optional, and** defaults to “”)

ARG = 'ARG'

ARRAY = 'ARRAY'

CLOSE = 'CLOSE'

ERROR = 'ERROR'

FUNC = 'FUNC'

LITERAL = 'LITERAL'

LOGICAL = 'LOGICAL'

NUMBER = 'NUMBER'

OPEN = 'OPEN'

OPERAND = 'OPERAND'

OP_IN = 'OPERATOR-INFIX'

OP_POST = 'OPERATOR-POSTFIX'

OP_PRE = 'OPERATOR-PREFIX'

PAREN = 'PAREN'

RANGE = 'RANGE'

`ROW = 'ROW'`

`SEP = 'SEP'`

`TEXT = 'TEXT'`

`WSPACE = 'WHITE-SPACE'`

`get_closer()`

Return a closing token that matches this token’ s type.

`classmethod make_operand(`*value*`)`

Create an operand token.

`classmethod make_separator(`*value*`)`

Create a separator token

`classmethod make_subexp(`*value*, *func=False*`)`

Create a subexpression token.

value: The value of the token *func*: If True, force the token to be of type FUNC

`subtype`

`type`

`value`

`class openpyxl.formula.tokenizer.Tokenizer(`*formula*`)`

基类：`object`

A tokenizer for Excel worksheet formulae.

Converts a str string representing an Excel formula (in A1 notation) into a sequence of *Token* objects.

formula: The str string to tokenize

Tokenizer defines a method *._parse()* to parse the formula into tokens, which can then be accessed through the *.items* attribute.

`ERROR_CODES = ('#NULL!', '#DIV/0!', '#VALUE!', '#REF!', '#NAME?', '#NUM!', '#N/A', '#GETTING_DATA')`

`SN_RE = re.compile('^[1-9](\\.[0-9]+)?[Ee]$')`

`STRING_REGEXES = {'"': re.compile('"(?:[^"]*"")*[^"]*"(?!")'), "'": re.compile("'(?:[^']*'')*[^']*`

`TOKEN_ENDERS = ',;}) +-*/^&=><%'`

`WSPACE_RE = re.compile('[ \\n]+')`

`assert_empty_token(`*can_follow=()*`)`

Ensure that there’ s no token currently being parsed.

Or if there is a token being parsed, it must end with a character in can_follow.

If there are unconsumed token contents, it means we hit an unexpected token transition. In this case, we raise a TokenizerError

`check_scientific_notation()`

Consumes a + or - character if part of a number in sci. notation.

Returns True if the character was consumed and self.offset was updated, False otherwise.

`render()`

Convert the parsed tokens back to a string.

`save_token()`

If there' s a token being parsed, add it to the item list.

`exception openpyxl.formula.tokenizer.TokenizerError`

基类：`Exception`

Base class for all Tokenizer errors.

openpyxl.formula.translate module

This module contains code to translate formulae across cells in a worksheet.

The idea is that if A1 has formula "=B1+C1" , then translating it to cell A2 results in formula "=B2+C2" . The algorithm relies on the formula tokenizer to identify the parts of the formula that need to change.

`class openpyxl.formula.translate.Translator(`*formula*, *origin*`)`

基类：`object`

Modifies a formula so that it can be translated from one cell to another.

***formula*: The str string to translate. Must include the leading '='** character.

***origin*: The cell address (in A1 notation) where this formula was** defined (excluding the worksheet name).

`CELL_REF_RE = re.compile('(\\$?[A-Za-z]{1,3})(\\$?[1-9][0-9]{0,6})$')`

`COL_RANGE_RE = re.compile('(\\$?[A-Za-z]{1,3}):(\\$?[A-Za-z]{1,3})$')`

`ROW_RANGE_RE = re.compile('(\\$?[1-9][0-9]{0,6}):(\\$?[1-9][0-9]{0,6})$')`

`get_tokens()`

Returns a list with the tokens comprising the formula.

`static strip_ws_name(`*range_str*`)`

Splits out the worksheet reference, if any, from a range reference.

`static translate_col(`*col_str*, *cdelta*`)`

Translate a range col-snippet by the given number of columns

translate_formula(*dest=None*, *row_delta=0*, *col_delta=0*)

Convert the formula into A1 notation, or as row and column coordinates

The formula is converted into A1 assuming it is assigned to the cell whose address is *dest* (no worksheet name).

classmethod translate_range(*range_str*, *rdelta*, *cdelta*)

Translate an A1-style range reference to the destination cell.

rdelta: the row offset to add to the range *cdelta*: the column offset to add to the range *range_str*: an A1-style reference to a range. Potentially includes

> the worksheet reference. Could also be a named range.

static translate_row(*row_str*, *rdelta*)

Translate a range row-snippet by the given number of rows.

exception openpyxl.formula.translate.**TranslatorError**

基类：`Exception`

Raised when a formula can’t be translated across cells.

This error arises when a formula’s references would be translated outside the worksheet’s bounds on the top or left. Excel represents these situations with a #REF! literal error. E.g., if the formula at B2 is ‘=A1’, attempting to translate the formula to B1 raises TranslatorError, since there’s no cell above A1. Similarly, translating the same formula from B2 to A2 raises TranslatorError, since there’s no cell to the left of A1.

openpyxl.packaging package

Stuff related to Office OpenXML packaging: relationships, archive, content types.

Submodules

openpyxl.packaging.core module

class openpyxl.packaging.core.**DocumentProperties**(*category=None*, *contentStatus=None*, *keywords=None*, *lastModifiedBy=None*, *lastPrinted=None*, *revision=None*, *version=None*, *created=datetime.datetime(2021, 8, 19, 11, 0, 17, 650940)*, *creator='openpyxl'*, *description=None*, *identifier=None*, *language=None*, *modified=datetime.datetime(2021, 8, 19, 11, 0, 17, 650942)*, *subject=None*, *title=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

High-level properties of the document. Defined in ECMA-376 Par2 Annex D

category

Values must be of type <class 'str'>

contentStatus

Values must be of type <class 'str'>

created

Values must be of type <class 'datetime.datetime'>

creator

Values must be of type <class 'str'>

description

Values must be of type <class 'str'>

identifier

Values must be of type <class 'str'>

keywords

Values must be of type <class 'str'>

language

Values must be of type <class 'str'>

lastModifiedBy

Values must be of type <class 'str'>

lastPrinted

Values must be of type <class 'datetime.datetime'>

last_modified_by

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

modified

Values must be of type <class 'datetime.datetime' >

namespace = 'http://schemas.openxmlformats.org/package/2006/metadata/core-properties'

revision

Values must be of type <class 'str' >

subject

Values must be of type <class 'str' >

tagname = 'coreProperties'

title

Values must be of type <class 'str' >

version

Values must be of type <class 'str' >

class openpyxl.packaging.core.**NestedDateTime**(**args*, ***kw*)

基类: *openpyxl.descriptors.base.DateTime*, *openpyxl.descriptors.nested.NestedText*

expected_type

datetime.datetime 的别名

to_tree(*tagname=None*, *value=None*, *namespace=None*)

class openpyxl.packaging.core.**QualifiedDateTime**(**args*, ***kw*)

基类: *openpyxl.packaging.core.NestedDateTime*

In certain situations Excel will complain if the additional type attribute isn't set

to_tree(*tagname=None*, *value=None*, *namespace=None*)

openpyxl.packaging.core.**tostring**(*element*, ***, *encoding='utf-8'*, *method=None*, *short_empty_elements=True*)

Generate string representation of XML element.

All subelements are included. If encoding is "unicode", a string is returned. Otherwise a bytestring is returned.

element is an Element instance, *encoding* is an optional output encoding defaulting to US-ASCII, *method* is an optional output which can be one of "xml" (default), "html", "text" or "c14n".

Returns an (optionally) encoded string containing the XML data.

openpyxl.packaging.extended module

class openpyxl.packaging.extended.**DigSigBlob**

基类: *openpyxl.descriptors.serialisable.Serialisable*

class openpyxl.packaging.extended.**ExtendedProperties**(*Template=None, Manager=None, Company=None, Pages=None, Words=None, Characters=None, PresentationFormat=None, Lines=None, Paragraphs=None, Slides=None, Notes=None, TotalTime=None, HiddenSlides=None, MMClips=None, ScaleCrop=None, HeadingPairs=None, TitlesOfParts=None, LinksUpToDate=None, CharactersWithSpaces=None, SharedDoc=None, HyperlinkBase=None, HLinks=None, HyperlinksChanged=None, DigSig=None, Application='Microsoft Excel', AppVersion=None, DocSecurity=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

See 22.2

Most of this is irrelevant

AppVersion

Values must be of type <class 'str'>

Application

Values must be of type <class 'str'>

Characters

Values must be of type <class 'int'>

CharactersWithSpaces

Values must be of type <class 'int'>

Company

Values must be of type <class 'str'>

DigSig

Values must be of type <class 'openpyxl.packaging.extended.DigSigBlob'>

DocSecurity

Values must be of type <class 'int'>

HLinks

Values must be of type <class 'openpyxl.packaging.extended.VectorVariant'>

HeadingPairs

Values must be of type <class 'openpyxl.packaging.extended.VectorVariant'>

`HiddenSlides`
Values must be of type <class ‘int’ >

`HyperlinkBase`
Values must be of type <class ‘str’ >

`HyperlinksChanged`
Values must be of type <class ‘bool’ >

`Lines`
Values must be of type <class ‘int’ >

`LinksUpToDate`
Values must be of type <class ‘bool’ >

`MMClips`
Values must be of type <class ‘int’ >

`Manager`
Values must be of type <class ‘str’ >

`Notes`
Values must be of type <class ‘int’ >

`Pages`
Values must be of type <class ‘int’ >

`Paragraphs`
Values must be of type <class ‘int’ >

`PresentationFormat`
Values must be of type <class ‘str’ >

`ScaleCrop`
Values must be of type <class ‘bool’ >

`SharedDoc`
Values must be of type <class ‘bool’ >

`Slides`
Values must be of type <class ‘int’ >

`Template`
Values must be of type <class ‘str’ >

`TitlesOfParts`
Values must be of type <class ‘openpyxl.packaging.extended.VectorLpstr’ >

`TotalTime`
Values must be of type <class ‘int’ >

Words
Values must be of type <class ‘int’ >

tagname = 'Properties'

to_tree()

class openpyxl.packaging.extended.**VectorLpstr**
基类：*openpyxl.descriptors.serialisable.Serialisable*

class openpyxl.packaging.extended.**VectorVariant**
基类：*openpyxl.descriptors.serialisable.Serialisable*

openpyxl.packaging.extended.**get_version()**

openpyxl.packaging.interface module

class openpyxl.packaging.interface.**ISerialisableFile**
基类：abc.ABC

Interface for Serialisable classes that represent files in the archive

id
Object id making it unique

openpyxl.packaging.manifest module

File manifest

class openpyxl.packaging.manifest.**FileExtension**(*Extension*, *ContentType*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

ContentType
Values must be of type <class ‘str’ >

Extension
Values must be of type <class ‘str’ >

tagname = 'Default'

class openpyxl.packaging.manifest.**Manifest**(*Default=()*, *Override=()*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

Default
A sequence (list or tuple) that may only contain objects of the declared type

Override
A sequence (list or tuple) that may only contain objects of the declared type

append(*obj*)

Add content object to the package manifest # needs a contract…

extensions

Map content types to file extensions Skip parts without extensions

filenames

find(*content_type*)

Find specific content-type

findall(*content_type*)

Find all elements of a specific content-type

path = '[Content_Types].xml'

tagname = 'Types'

to_tree()

Custom serialisation method to allow setting a default namespace

class openpyxl.packaging.manifest.**Override**(*PartName*, *ContentType*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

ContentType

Values must be of type <class 'str'>

PartName

Values must be of type <class 'str'>

tagname = 'Override'

openpyxl.packaging.manifest.**tostring**(*element*, ***, *encoding='utf-8'*, *method=None*, *short_empty_elements=True*)

Generate string representation of XML element.

All subelements are included. If encoding is "unicode", a string is returned. Otherwise a bytestring is returned.

element is an Element instance, *encoding* is an optional output encoding defaulting to US-ASCII, *method* is an optional output which can be one of "xml" (default), "html", "text" or "c14n".

Returns an (optionally) encoded string containing the XML data.

openpyxl.packaging.relationship module

class openpyxl.packaging.relationship.**Relationship**(*Id=None*, *Type=None*, *type=None*, *Target=None*, *TargetMode=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Represents many kinds of relationships.

Id
Values must be of type <class 'str'>

Target
Values must be of type <class 'str'>

TargetMode
Values must be of type <class 'str'>

Type
Values must be of type <class 'str'>

id
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

tagname = 'Relationship'

target
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

class openpyxl.packaging.relationship.**RelationshipList**(*Relationship=()*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

Relationship
A sequence (list or tuple) that may only contain objects of the declared type

append(*value*)

find(*content_type*)
Find relationships by content-type NB. these content-types namespaced objects and different to the MIME-types in the package manifest :-(

tagname = 'Relationships'

to_tree()

openpyxl.packaging.relationship.**get_dependents**(*archive*, *filename*)
Normalise dependency file paths to absolute ones

Relative paths are relative to parent object

openpyxl.packaging.relationship.**get_rel**(*archive*, *deps*, *id=None*, *cls=None*)
Get related object based on id or rel_type

openpyxl.packaging.relationship.**get_rels_path**(*path*)
Convert relative path to absolutes that can be loaded from a zip archive. The path to be passed in is that of containing object (workbook, worksheet, etc.)

`openpyxl.packaging.relationship.tostring`(*element*, ***, *encoding='utf-8'*, *method=None*, *short_empty_elements=True*)

Generate string representation of XML element.

All subelements are included. If encoding is "unicode", a string is returned. Otherwise a bytestring is returned.

element is an Element instance, *encoding* is an optional output encoding defaulting to US-ASCII, *method* is an optional output which can be one of "xml" (default), "html", "text" or "c14n".

Returns an (optionally) encoded string containing the XML data.

openpyxl.packaging.workbook module

class `openpyxl.packaging.workbook.ChildSheet`(*name=None*, *sheetId=None*, *state='visible'*, *id=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

Represents a reference to a worksheet or chartsheet in workbook.xml

It contains the title, order and state but only an indirect reference to the objects themselves.

id

Values must be of type <class 'str'>

name

Values must be of type <class 'str'>

sheetId

Values must be of type <class 'int'>

state

Value must be one of {'veryHidden', 'visible', 'hidden'}

tagname = 'sheet'

class `openpyxl.packaging.workbook.FileRecoveryProperties`(*autoRecover=None*, *crashSave=None*, *dataExtractLoad=None*, *repairLoad=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

autoRecover

Values must be of type <class 'bool'>

crashSave

Values must be of type <class 'bool'>

dataExtractLoad

Values must be of type <class 'bool'>

repairLoad

Values must be of type <class 'bool'>

tagname = 'fileRecoveryPr'

class openpyxl.packaging.workbook.**PivotCache**(*cacheId=None*, *id=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

cacheId

Values must be of type <class 'int' >

id

Values must be of type <class 'str' >

tagname = 'pivotCache'

class openpyxl.packaging.workbook.**WorkbookPackage**(*conformance=None*, *fileVersion=None*, *fileSharing=None*, *workbookPr=None*, *workbookProtection=None*, *bookViews=()*, *sheets=()*, *functionGroups=None*, *externalReferences=()*, *definedNames=None*, *calcPr=None*, *oleSize=None*, *customWorkbookViews=()*, *pivotCaches=()*, *smartTagPr=None*, *smartTagTypes=None*, *webPublishing=None*, *fileRecoveryPr=None*, *webPublishObjects=None*, *extLst=None*, *Ignorable=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Represent the workbook file in the archive

Ignorable

Values must be of type <class 'str' >

active

bookViews

Wrap a sequence in an containing object

calcPr

Values must be of type <class 'openpyxl.workbook.properties.CalcProperties' >

conformance

Value must be one of { 'transitional' , 'strict' }

customWorkbookViews

Wrap a sequence in an containing object

definedNames

Values must be of type <class 'openpyxl.workbook.defined_name.DefinedNameList' >

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

externalReferences
Wrap a sequence in an containing object

fileRecoveryPr
Values must be of type <class ‘openpyxl.packaging.workbook.FileRecoveryProperties’ >

fileSharing
Values must be of type <class ‘openpyxl.workbook.protection.FileSharing’ >

fileVersion
Values must be of type <class ‘openpyxl.workbook.properties.FileVersion’ >

functionGroups
Values must be of type <class ‘openpyxl.workbook.function_group.FunctionGroupList’ >

oleSize
Values must be of type <class ‘str’ >

pivotCaches
Wrap a sequence in an containing object

pivot_caches
Get PivotCache objects

properties
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

sheets
Wrap a sequence in an containing object

smartTagPr
Values must be of type <class ‘openpyxl.workbook.smart_tags.SmartTagProperties’ >

smartTagTypes
Values must be of type <class ‘openpyxl.workbook.smart_tags.SmartTagList’ >

tagname = 'workbook'

to_tree()

webPublishObjects
Values must be of type <class ‘openpyxl.workbook.web.WebPublishObjectList’ >

webPublishing
Values must be of type <class ‘openpyxl.workbook.web.WebPublishing’ >

workbookPr
Values must be of type <class ‘openpyxl.workbook.properties.WorkbookProperties’ >

workbookProtection
Values must be of type <class ‘openpyxl.workbook.protection.WorkbookProtection’ >

openpyxl.pivot package

Submodules

openpyxl.pivot.cache module

class openpyxl.pivot.cache.**CacheDefinition**(*invalid=None, saveData=None, refreshOnLoad=None, optimizeMemory=None, enableRefresh=None, refreshedBy=None, refreshedDate=None, refreshedDateIso=None, backgroundQuery=None, missingItemsLimit=None, createdVersion=None, refreshedVersion=None, minRefreshableVersion=None, recordCount=None, upgradeOnRefresh=None, tupleCache=None, supportSubquery=None, supportAdvancedDrill=None, cacheSource=None, cacheFields=(), cacheHierarchies=(), kpis=(), calculatedItems=(), calculatedMembers=(), dimensions=(), measureGroups=(), maps=(), extLst=None, id=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

backgroundQuery

Values must be of type <class 'bool'>

cacheFields

Wrap a sequence in an containing object

cacheHierarchies

Wrap a sequence in an containing object

cacheSource

Values must be of type <class 'openpyxl.pivot.cache.CacheSource'>

calculatedItems

Wrap a sequence in an containing object

calculatedMembers

Wrap a sequence in an containing object

createdVersion

Values must be of type <class 'int'>

dimensions

Wrap a sequence in an containing object

enableRefresh

Values must be of type <class 'bool'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

id

Values must be of type <class 'str'>

invalid

Values must be of type <class 'bool'>

kpis

Wrap a sequence in an containing object

maps

Wrap a sequence in an containing object

measureGroups

Wrap a sequence in an containing object

mime_type = 'application/vnd.openxmlformats-officedocument.spreadsheetml.pivotCacheDefinition+xml'

minRefreshableVersion

Values must be of type <class 'int'>

missingItemsLimit

Values must be of type <class 'int'>

optimizeMemory

Values must be of type <class 'bool'>

path

recordCount

Values must be of type <class 'int'>

records = None

refreshOnLoad

Values must be of type <class 'bool'>

refreshedBy

Values must be of type <class 'str'>

refreshedDate

Values must be of type <class 'float'>

refreshedDateIso

Values must be of type <class 'datetime.datetime'>

refreshedVersion

Values must be of type <class 'int'>

rel_type = 'http://schemas.openxmlformats.org/officeDocument/2006/relationships/pivotCacheDefinitio

saveData

Values must be of type <class 'bool' >

supportAdvancedDrill

Values must be of type <class 'bool' >

supportSubquery

Values must be of type <class 'bool' >

tagname = 'pivotCacheDefinition'

to_tree()

tupleCache

Values must be of type <class 'openpyxl.pivot.cache.TupleCache' >

upgradeOnRefresh

Values must be of type <class 'bool' >

class openpyxl.pivot.cache.**CacheField**(*sharedItems=None, fieldGroup=None, mpMap=None, extLst=None, name=None, caption=None, propertyName=None, serverField=None, uniqueList=True, numFmtId=None, formula=None, sqlType=0, hierarchy=0, level=0, databaseField=True, mappingCount=None, memberPropertyField=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

caption

Values must be of type <class 'str' >

databaseField

Values must be of type <class 'bool' >

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

fieldGroup

Values must be of type <class 'openpyxl.pivot.cache.FieldGroup' >

formula

Values must be of type <class 'str' >

hierarchy

Values must be of type <class 'int' >

level

Values must be of type <class 'int' >

mappingCount

Values must be of type <class 'int' >

memberPropertyField

Values must be of type <class 'bool'>

mpMap

Values must be of type <class 'int'>

name

Values must be of type <class 'str'>

numFmtId

Values must be of type <class 'int'>

propertyName

Values must be of type <class 'str'>

serverField

Values must be of type <class 'bool'>

sharedItems

Values must be of type <class 'openpyxl.pivot.cache.SharedItems'>

sqlType

Values must be of type <class 'int'>

tagname = 'cacheField'

uniqueList

Values must be of type <class 'bool'>

class openpyxl.pivot.cache.**CacheHierarchy**(*uniqueName=", caption=None, measure=None, set=None, parentSet=None, iconSet=0, attribute=None, time=None, keyAttribute=None, defaultMemberUniqueName=None, allUniqueName=None, allCaption=None, dimensionUniqueName=None, displayFolder=None, measureGroup=None, measures=None, count=None, oneField=None, memberValueDatatype=None, unbalanced=None, unbalancedGroup=None, hidden=None, fieldsUsage=None, groupLevels=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

allCaption

Values must be of type <class 'str'>

allUniqueName

Values must be of type <class 'str'>

attribute

Values must be of type <class 'bool'>

caption
Values must be of type <class ‘str’ >

count
Values must be of type <class ‘int’ >

defaultMemberUniqueName
Values must be of type <class ‘str’ >

dimensionUniqueName
Values must be of type <class ‘str’ >

displayFolder
Values must be of type <class ‘str’ >

extLst
Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

fieldsUsage
Values must be of type <class ‘openpyxl.pivot.cache.FieldsUsage’ >

groupLevels
Values must be of type <class ‘openpyxl.pivot.cache.GroupLevels’ >

hidden
Values must be of type <class ‘bool’ >

iconSet
Values must be of type <class ‘int’ >

keyAttribute
Values must be of type <class ‘bool’ >

measure
Values must be of type <class ‘bool’ >

measureGroup
Values must be of type <class ‘str’ >

measures
Values must be of type <class ‘bool’ >

memberValueDatatype
Values must be of type <class ‘int’ >

oneField
Values must be of type <class ‘bool’ >

parentSet
Values must be of type <class ‘int’ >

set
Values must be of type <class 'bool' >

tagname = 'cacheHierarchy'

time
Values must be of type <class 'bool' >

unbalanced
Values must be of type <class 'bool' >

unbalancedGroup
Values must be of type <class 'bool' >

uniqueName
Values must be of type <class 'str' >

class openpyxl.pivot.cache.CacheSource(*type=None*, *connectionId=None*, *worksheetSource=None*, *consolidation=None*, *extLst=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

connectionId
Values must be of type <class 'int' >

consolidation
Values must be of type <class 'openpyxl.pivot.cache.Consolidation' >

extLst
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

tagname = 'cacheSource'

type
Value must be one of { 'scenario' , 'consolidation' , 'external' , 'worksheet' }

worksheetSource
Values must be of type <class 'openpyxl.pivot.cache.WorksheetSource' >

class openpyxl.pivot.cache.CalculatedItem(*field=None*, *formula=None*, *pivotArea=None*, *extLst=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

extLst
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

field
Values must be of type <class 'int' >

formula
Values must be of type <class 'str' >

pivotArea
Values must be of type <class 'openpyxl.pivot.table.PivotArea' >

tagname = 'calculatedItem'

class openpyxl.pivot.cache.**CalculatedMember**(*name=None*, *mdx=None*, *memberName=None*, *hierarchy=None*, *parent=None*, *solveOrder=None*, *set=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

hierarchy

Values must be of type <class ‘str’ >

mdx

Values must be of type <class ‘str’ >

memberName

Values must be of type <class ‘str’ >

name

Values must be of type <class ‘str’ >

parent

Values must be of type <class ‘str’ >

set

Values must be of type <class ‘bool’ >

solveOrder

Values must be of type <class ‘int’ >

tagname = 'calculatedMember'

class openpyxl.pivot.cache.**Consolidation**(*autoPage=None*, *pages=()*, *rangeSets=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

autoPage

Values must be of type <class ‘bool’ >

pages

Wrap a sequence in an containing object

rangeSets

Wrap a sequence in an containing object

tagname = 'consolidation'

class openpyxl.pivot.cache.**DiscretePr**(*count=None*, *x=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

count

Values must be of type <class ‘int’ >

tagname = 'discretePr'

x

Values must be of type <class ‘int’ >

class openpyxl.pivot.cache.**FieldGroup**(*par=None*, *base=None*, *rangePr=None*, *discretePr=None*, *groupItems=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

base

Values must be of type <class ‘int’ >

discretePr

Values must be of type <class ‘openpyxl.pivot.cache.DiscretePr’ >

groupItems

Values must be of type <class ‘openpyxl.pivot.cache.GroupItems’ >

par

Values must be of type <class ‘int’ >

rangePr

Values must be of type <class ‘openpyxl.pivot.cache.RangePr’ >

tagname = 'fieldGroup'

class openpyxl.pivot.cache.**FieldUsage**(*x=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

tagname = 'fieldUsage'

x

Values must be of type <class ‘int’ >

class openpyxl.pivot.cache.**FieldsUsage**(*count=None*, *fieldUsage=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

count

Values must be of type <class ‘int’ >

fieldUsage

Values must be of type <class ‘openpyxl.pivot.cache.FieldUsage’ >

class openpyxl.pivot.cache.**GroupItems**(*count=None*, *m=()*, *n=()*, *b=()*, *e=()*, *s=()*, *d=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

b

A sequence (list or tuple) that may only contain objects of the declared type

count

d

A sequence (list or tuple) that may only contain objects of the declared type

e

A sequence (list or tuple) that may only contain objects of the declared type

m

A sequence (list or tuple) that may only contain objects of the declared type

n

A sequence (list or tuple) that may only contain objects of the declared type

s

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'groupItems'

class openpyxl.pivot.cache.**GroupLevel**(*uniqueName=None*, *caption=None*, *user=None*, *customRollUp=None*, *groups=None*, *extLst=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

caption

Values must be of type <class ‘str’ >

customRollUp

Values must be of type <class ‘bool’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

groups

Values must be of type <class ‘openpyxl.pivot.cache.Groups’ >

tagname = 'groupLevel'

uniqueName

Values must be of type <class ‘str’ >

user

Values must be of type <class ‘bool’ >

class openpyxl.pivot.cache.**GroupLevels**(*count=None*, *groupLevel=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

count

Values must be of type <class ‘int’ >

groupLevel

Values must be of type <class ‘openpyxl.pivot.cache.GroupLevel’ >

class openpyxl.pivot.cache.**GroupMember**(*uniqueName=None*, *group=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

group

Values must be of type <class ‘bool’ >

tagname = 'groupMember'

uniqueName

Values must be of type <class 'str' >

class openpyxl.pivot.cache.**GroupMembers**(*count=None*, *groupMember=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

count

Values must be of type <class 'int' >

groupMember

Values must be of type <class 'openpyxl.pivot.cache.GroupMember' >

class openpyxl.pivot.cache.**Groups**(*count=None*, *group=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

count

Values must be of type <class 'int' >

group

Values must be of type <class 'openpyxl.pivot.cache.LevelGroup' >

tagname = 'groups'

class openpyxl.pivot.cache.**LevelGroup**(*name=None*, *uniqueName=None*, *caption=None*, *uniqueParent=None*, *id=None*, *groupMembers=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

caption

Values must be of type <class 'str' >

groupMembers

Values must be of type <class 'openpyxl.pivot.cache.GroupMembers' >

id

Values must be of type <class 'int' >

name

Values must be of type <class 'str' >

tagname = 'levelGroup'

uniqueName

Values must be of type <class 'str' >

uniqueParent

Values must be of type <class 'str' >

class openpyxl.pivot.cache.**MeasureDimensionMap**(*measureGroup=None*, *dimension=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

dimension

Values must be of type <class 'int' >

measureGroup

Values must be of type <class 'int' >

tagname = 'map'

class openpyxl.pivot.cache.**MeasureGroup**(*name=None*, *caption=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

caption

Values must be of type <class 'str' >

name

Values must be of type <class 'str' >

tagname = 'measureGroup'

class openpyxl.pivot.cache.**OLAPSet**(*count=None*, *maxRank=None*, *setDefinition=None*, *sortType=None*, *queryFailed=None*, *tpls=None*, *sortByTuple=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

count

Values must be of type <class 'int' >

maxRank

Values must be of type <class 'int' >

queryFailed

Values must be of type <class 'bool' >

setDefinition

Values must be of type <class 'str' >

sortByTuple

Values must be of type <class 'openpyxl.pivot.fields.TupleList' >

sortType

Value must be one of { 'descendingAlpha' , 'ascendingNatural' , 'ascendingAlpha' , 'descending' , 'ascending' , 'descendingNatural' }

tagname = 'set'

tpls

Values must be of type <class 'openpyxl.pivot.fields.TupleList' >

class openpyxl.pivot.cache.**OLAPSets**(*count=None*, *set=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

count
Values must be of type <class 'int' >

set
Values must be of type <class 'openpyxl.pivot.cache.OLAPSet' >

class openpyxl.pivot.cache.PCDKPI(*uniqueName=None*, *caption=None*, *displayFolder=None*, *measureGroup=None*, *parent=None*, *value=None*, *goal=None*, *status=None*, *trend=None*, *weight=None*, *time=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

caption
Values must be of type <class 'str' >

displayFolder
Values must be of type <class 'str' >

goal
Values must be of type <class 'str' >

measureGroup
Values must be of type <class 'str' >

parent
Values must be of type <class 'str' >

status
Values must be of type <class 'str' >

tagname = 'pCDKPI'

time
Values must be of type <class 'str' >

trend
Values must be of type <class 'str' >

uniqueName
Values must be of type <class 'str' >

value
Values must be of type <class 'str' >

weight
Values must be of type <class 'str' >

class openpyxl.pivot.cache.PCDSDTCEntries(*count=None*, *m=None*, *n=None*, *e=None*, *s=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

count

Values must be of type <class ‘int’ >

e

Values must be of type <class ‘openpyxl.pivot.fields.Error’ >

m

Values must be of type <class ‘openpyxl.pivot.fields.Missing’ >

n

Values must be of type <class ‘openpyxl.pivot.fields.Number’ >

s

Values must be of type <class ‘openpyxl.pivot.fields.Text’ >

tagname = 'pCDSDTCEntries'

class openpyxl.pivot.cache.**Page**(*count=None*, *pageItem=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

count

pageItem

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'PCDSCPage'

class openpyxl.pivot.cache.**PageItem**(*name=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

name

Values must be of type <class ‘str’ >

tagname = 'pageItem'

class openpyxl.pivot.cache.**PivotDimension**(*measure=None*, *name=None*, *uniqueName=None*, *caption=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

caption

Values must be of type <class ‘str’ >

measure

Values must be of type <class ‘bool’ >

name

Values must be of type <class ‘str’ >

tagname = 'dimension'

uniqueName

Values must be of type <class ‘str’ >

class openpyxl.pivot.cache.**Query**(*mdx=None*, *tpls=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

mdx

Values must be of type <class ‘str’ >

tagname = 'query'

tpls

Values must be of type <class ‘openpyxl.pivot.fields.TupleList’ >

class openpyxl.pivot.cache.**QueryCache**(*count=None*, *query=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

count

Values must be of type <class ‘int’ >

query

Values must be of type <class ‘openpyxl.pivot.cache.Query’ >

tagname = 'queryCache'

class openpyxl.pivot.cache.**RangePr**(*autoStart=True*, *autoEnd=True*, *groupBy=<class 'range'>*, *startNum=None*, *endNum=None*, *startDate=None*, *endDate=None*, *groupInterval=1*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

autoEnd

Values must be of type <class ‘bool’ >

autoStart

Values must be of type <class ‘bool’ >

endDate

Values must be of type <class ‘datetime.datetime’ >

endNum

Values must be of type <class ‘float’ >

groupBy

Value must be one of { ‘years’ , ‘seconds’ , ‘hours’ , ‘range’ , ‘quarters’ , ‘minutes’ , ‘months’ , ‘days’ }

groupInterval

Values must be of type <class ‘float’ >

startDate

Values must be of type <class ‘datetime.datetime’ >

startNum

Values must be of type <class ‘float’ >

tagname = 'rangePr'

class openpyxl.pivot.cache.**RangeSet**(*i1=None*, *i2=None*, *i3=None*, *i4=None*, *ref=None*, *name=None*, *sheet=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

i1

Values must be of type <class 'int'>

i2

Values must be of type <class 'int'>

i3

Values must be of type <class 'int'>

i4

Values must be of type <class 'int'>

name

Values must be of type <class 'str'>

ref

Values must be of type <class 'str'>

sheet

Values must be of type <class 'str'>

tagname = 'rangeSet'

class openpyxl.pivot.cache.**ServerFormat**(*culture=None*, *format=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

culture

Values must be of type <class 'str'>

format

Values must be of type <class 'str'>

tagname = 'serverFormat'

class openpyxl.pivot.cache.**ServerFormatList**(*count=None*, *serverFormat=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

count

serverFormat

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'serverFormats'

class openpyxl.pivot.cache.**SharedItems**(*_fields=()*, *containsSemiMixedTypes=None*, *containsNonDate=None*, *containsDate=None*, *containsString=None*, *containsBlank=None*, *containsMixedTypes=None*, *containsNumber=None*, *containsInteger=None*, *minValue=None*, *maxValue=None*, *minDate=None*, *maxDate=None*, *count=None*, *longText=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

b

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

containsBlank

Values must be of type <class ‘bool’ >

containsDate

Values must be of type <class ‘bool’ >

containsInteger

Values must be of type <class ‘bool’ >

containsMixedTypes

Values must be of type <class ‘bool’ >

containsNonDate

Values must be of type <class ‘bool’ >

containsNumber

Values must be of type <class ‘bool’ >

containsSemiMixedTypes

Values must be of type <class ‘bool’ >

containsString

Values must be of type <class ‘bool’ >

count

d

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

e

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

longText

Values must be of type <class ‘bool’ >

m

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

maxDate

Values must be of type <class 'datetime.datetime'>

maxValue

Values must be of type <class 'float'>

minDate

Values must be of type <class 'datetime.datetime'>

minValue

Values must be of type <class 'float'>

n

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

s

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

tagname = 'sharedItems'

class openpyxl.pivot.cache.**TupleCache**(*entries=None*, *sets=None*, *queryCache=None*, *serverFormats=None*, *extLst=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

entries

Values must be of type <class 'openpyxl.pivot.cache.PCDSDTCEntries'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

queryCache

Values must be of type <class 'openpyxl.pivot.cache.QueryCache'>

serverFormats

Values must be of type <class 'openpyxl.pivot.cache.ServerFormatList'>

sets

Values must be of type <class 'openpyxl.pivot.cache.OLAPSets'>

tagname = 'tupleCache'

class openpyxl.pivot.cache.**WorksheetSource**(*ref=None*, *name=None*, *sheet=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

name
Values must be of type <class 'str' >

ref
Values must be of type <class 'str' >

sheet
Values must be of type <class 'str' >

tagname = 'worksheetSource'

openpyxl.pivot.cache.**tostring**(*element*, ***, *encoding='utf-8'*, *method=None*, *short_empty_elements=True*)

Generate string representation of XML element.

All subelements are included. If encoding is "unicode" , a string is returned. Otherwise a bytestring is returned.

element is an Element instance, *encoding* is an optional output encoding defaulting to US-ASCII, *method* is an optional output which can be one of "xml" (default), "html" , "text" or "c14n" .

Returns an (optionally) encoded string containing the XML data.

openpyxl.pivot.fields module

class openpyxl.pivot.fields.**Boolean**(*x=()*, *v=None*, *u=None*, *f=None*, *c=None*, *cp=None*)
基类： *openpyxl.descriptors.serialisable.Serialisable*

c
Values must be of type <class 'str' >

cp
Values must be of type <class 'int' >

f
Values must be of type <class 'bool' >

tagname = 'b'

u
Values must be of type <class 'bool' >

v
Values must be of type <class 'bool' >

x
A sequence (list or tuple) that may only contain objects of the declared type

class openpyxl.pivot.fields.**DateTimeField**(*x=()*, *v=None*, *u=None*, *f=None*, *c=None*, *cp=None*)
基类： *openpyxl.descriptors.serialisable.Serialisable*

`c`

Values must be of type <class ‘str’ >

`cp`

Values must be of type <class ‘int’ >

`f`

Values must be of type <class ‘bool’ >

`tagname = 'd'`

`u`

Values must be of type <class ‘bool’ >

`v`

Values must be of type <class ‘datetime.datetime’ >

`x`

A sequence (list or tuple) that may only contain objects of the declared type

class openpyxl.pivot.fields.Error(*tpls=None, x=(), v=None, u=None, f=None, c=None, cp=None, _in=None, bc=None, fc=None, i=None, un=None, st=None, b=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`b`

Values must be of type <class ‘bool’ >

`bc`

`c`

Values must be of type <class ‘str’ >

`cp`

Values must be of type <class ‘int’ >

`f`

Values must be of type <class ‘bool’ >

`fc`

`i`

Values must be of type <class ‘bool’ >

`st`

Values must be of type <class ‘bool’ >

`tagname = 'e'`

`tpls`

Values must be of type <class ‘openpyxl.pivot.fields.TupleList’ >

u

Values must be of type <class 'bool' >

un

Values must be of type <class 'bool' >

v

Values must be of type <class 'str' >

x

A sequence (list or tuple) that may only contain objects of the declared type

class openpyxl.pivot.fields.**Index**(*v=0*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

tagname = 'x'

v

Values must be of type <class 'int' >

class openpyxl.pivot.fields.**Missing**(*tpls=()*, *x=()*, *u=None*, *f=None*, *c=None*, *cp=None*, *_in=None*, *bc=None*, *fc=None*, *i=None*, *un=None*, *st=None*, *b=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

b

Values must be of type <class 'bool' >

bc

c

Values must be of type <class 'str' >

cp

Values must be of type <class 'int' >

f

Values must be of type <class 'bool' >

fc

i

Values must be of type <class 'bool' >

st

Values must be of type <class 'bool' >

tagname = 'm'

tpls

A sequence (list or tuple) that may only contain objects of the declared type

u

Values must be of type <class ‘bool’ >

un

Values must be of type <class ‘bool’ >

x

A sequence (list or tuple) that may only contain objects of the declared type

class openpyxl.pivot.fields.Number(*tpls=()*, *x=()*, *v=None*, *u=None*, *f=None*, *c=None*, *cp=None*, *_in=None*, *bc=None*, *fc=None*, *i=None*, *un=None*, *st=None*, *b=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

b

Values must be of type <class ‘bool’ >

bc

c

Values must be of type <class ‘str’ >

cp

Values must be of type <class ‘int’ >

f

Values must be of type <class ‘bool’ >

fc

i

Values must be of type <class ‘bool’ >

st

Values must be of type <class ‘bool’ >

tagname = 'n'

tpls

A sequence (list or tuple) that may only contain objects of the declared type

u

Values must be of type <class ‘bool’ >

un

Values must be of type <class ‘bool’ >

v

Values must be of type <class ‘float’ >

x

A sequence (list or tuple) that may only contain objects of the declared type

class openpyxl.pivot.fields.Text(*tpls=()*, *x=()*, *v=None*, *u=None*, *f=None*, *c=None*, *cp=None*, *_in=None*, *bc=None*, *fc=None*, *i=None*, *un=None*, *st=None*, *b=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

b

Values must be of type <class 'bool' >

bc

c

Values must be of type <class 'str' >

cp

Values must be of type <class 'int' >

f

Values must be of type <class 'bool' >

fc

i

Values must be of type <class 'bool' >

st

Values must be of type <class 'bool' >

tagname = 's'

tpls

A sequence (list or tuple) that may only contain objects of the declared type

u

Values must be of type <class 'bool' >

un

Values must be of type <class 'bool' >

v

Values must be of type <class 'str' >

x

A sequence (list or tuple) that may only contain objects of the declared type

class openpyxl.pivot.fields.Tuple(*fld=None*, *hier=None*, *item=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

fld

Values must be of type <class 'int' >

hier

Values must be of type <class 'int' >

item

Values must be of type <class 'int'>

class openpyxl.pivot.fields.**TupleList**(*c=None*, *tpl=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

c

Values must be of type <class 'int'>

tpl

Values must be of type <class 'openpyxl.pivot.fields.Tuple'>

openpyxl.pivot.record module

class openpyxl.pivot.record.**Record**(*_fields=()*, *m=None*, *n=None*, *b=None*, *e=None*, *s=None*, *d=None*, *x=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

b

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

d

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

e

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

m

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

n

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

s

Allow a multisequence to be built up from parts

Excluded from the instance __elements__ or __attrs__ as is effectively an Alias

tagname = 'r'

x

Allow a multisequence to be built up from parts

Excluded from the instance ___elements___ or ___attrs___ as is effectively an Alias

class openpyxl.pivot.record.**RecordList**(*count=None*, *r=()*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

count

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

mime_type = 'application/vnd.openxmlformats-officedocument.spreadsheetml.pivotCacheRecords+xml'

path

r

A sequence (list or tuple) that may only contain objects of the declared type

rel_type = 'http://schemas.openxmlformats.org/officeDocument/2006/relationships/pivotCacheRecords'

tagname = 'pivotCacheRecords'

to_tree()

openpyxl.pivot.record.**tostring**(*element*, ***, *encoding='utf-8'*, *method=None*, *short_empty_elements=True*)

Generate string representation of XML element.

All subelements are included. If encoding is "unicode", a string is returned. Otherwise a bytestring is returned.

element is an Element instance, *encoding* is an optional output encoding defaulting to US-ASCII, *method* is an optional output which can be one of "xml" (default), "html", "text" or "c14n".

Returns an (optionally) encoded string containing the XML data.

openpyxl.pivot.table module

class openpyxl.pivot.table.**AutoSortScope**(*pivotArea=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

pivotArea

Values must be of type <class 'openpyxl.pivot.table.PivotArea'>

class openpyxl.pivot.table.**ChartFormat**(*chart=None*, *format=None*, *series=None*, *pivotArea=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

chart

Values must be of type <class 'int'>

format

Values must be of type <class 'int'>

pivotArea

Values must be of type <class ‘openpyxl.pivot.table.PivotArea’ >

series

Values must be of type <class ‘bool’ >

tagname = 'chartFormat'

class openpyxl.pivot.table.**ColHierarchiesUsage**(*count=None*, *colHierarchyUsage=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

colHierarchyUsage

A sequence (list or tuple) that may only contain objects of the declared type

count

tagname = 'colHierarchiesUsage'

class openpyxl.pivot.table.**ConditionalFormat**(*scope=None*, *type=None*, *priority=None*, *pivotAreas=()*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

pivotAreas

Wrap a sequence in an containing object

priority

Values must be of type <class ‘int’ >

scope

Value must be one of { ‘data’ , ‘selection’ , ‘field’ }

tagname = 'conditionalFormat'

type

Value must be one of { ‘column’ , ‘row’ , ‘all’ }

class openpyxl.pivot.table.**DataField**(*name=None*, *fld=None*, *subtotal='sum'*, *showDataAs='normal'*, *baseField=-1*, *baseItem=1048832*, *numFmtId=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

baseField

Values must be of type <class ‘int’ >

baseItem

Values must be of type <class ‘int’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

fld

Values must be of type <class 'int' >

name

Values must be of type <class 'str' >

numFmtId

Values must be of type <class 'int' >

showDataAs

Value must be one of { 'percentDiff' , 'difference' , 'percentOfTotal' , 'percent' , 'normal' , 'index' , 'runTotal' , 'percentOfRow' , 'percentOfCol' }

subtotal

Value must be one of { 'sum' , 'countNums' , 'min' , 'stdDev' , 'max' , 'average' , 'count' , 'var' , 'varp' , 'product' , 'stdDevp' }

tagname = 'dataField'

class openpyxl.pivot.table.**FieldItem**(*n=None*, *t='data'*, *h=None*, *s=None*, *sd=True*, *f=None*, *m=None*, *c=None*, *x=None*, *d=None*, *e=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

c

Values must be of type <class 'bool' >

d

Values must be of type <class 'bool' >

e

Values must be of type <class 'bool' >

f

Values must be of type <class 'bool' >

h

Values must be of type <class 'bool' >

m

Values must be of type <class 'bool' >

n

Values must be of type <class 'str' >

s

Values must be of type <class 'bool' >

sd

Values must be of type <class 'bool' >

t

Value must be one of { 'data' , 'sum' , 'grand' , 'min' , 'stdDev' , 'max' , 'var' , 'count', 'stdDevP', 'avg', 'default', 'countA', 'blank', 'varP', 'product'}

tagname = 'item'

x

Values must be of type <class 'int'>

class openpyxl.pivot.table.**Format**(*action='formatting'*, *dxfId=None*, *pivotArea=None*, *extLst=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

action

Value must be one of {'blank', 'formatting', 'formula', 'drill'}

dxfId

Values must be of type <class 'int'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

pivotArea

Values must be of type <class 'openpyxl.pivot.table.PivotArea'>

tagname = 'format'

class openpyxl.pivot.table.**HierarchyUsage**(*hierarchyUsage=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

hierarchyUsage

Values must be of type <class 'int'>

tagname = 'hierarchyUsage'

class openpyxl.pivot.table.**Location**(*ref=None*, *firstHeaderRow=None*, *firstDataRow=None*, *firstDataCol=None*, *rowPageCount=None*, *colPageCount=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

colPageCount

Values must be of type <class 'int'>

firstDataCol

Values must be of type <class 'int'>

firstDataRow

Values must be of type <class 'int'>

firstHeaderRow

Values must be of type <class 'int'>

ref

Values must be of type <class 'str'>

rowPageCount

Values must be of type <class 'int' >

tagname = 'location'

class openpyxl.pivot.table.**MemberList**(*count=None*, *level=None*, *member=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

count

level

Values must be of type <class 'int' >

member

Wrap a sequence in an containing object

tagname = 'members'

class openpyxl.pivot.table.**MemberProperty**(*name=None*, *showCell=None*, *showTip=None*, *showAsCaption=None*, *nameLen=None*, *pPos=None*, *pLen=None*, *level=None*, *field=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

field

Values must be of type <class 'int' >

level

Values must be of type <class 'int' >

name

Values must be of type <class 'str' >

nameLen

Values must be of type <class 'int' >

pLen

Values must be of type <class 'int' >

pPos

Values must be of type <class 'int' >

showAsCaption

Values must be of type <class 'bool' >

showCell

Values must be of type <class 'bool' >

showTip

Values must be of type <class 'bool' >

tagname = 'mps'

class openpyxl.pivot.table.PageField(*fld=None, item=None, hier=None, name=None, cap=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

cap

Values must be of type <class 'str' >

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

fld

Values must be of type <class 'int' >

hier

Values must be of type <class 'int' >

item

Values must be of type <class 'int' >

name

Values must be of type <class 'str' >

tagname = 'pageField'

class openpyxl.pivot.table.PivotArea(*references=(), extLst=None, field=None, type='normal', dataOnly=True, labelOnly=None, grandRow=None, grandCol=None, cacheIndex=None, outline=True, offset=None, collapsedLevelsAreSubtotals=None, axis=None, fieldPosition=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

axis

Value must be one of { 'axisCol' , 'axisRow' , 'axisPage' , 'axisValues' }

cacheIndex

Values must be of type <class 'bool' >

collapsedLevelsAreSubtotals

Values must be of type <class 'bool' >

dataOnly

Values must be of type <class 'bool' >

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

field

Values must be of type <class 'int' >

fieldPosition

Values must be of type <class 'int' >

grandCol
Values must be of type <class ‘bool’ >

grandRow
Values must be of type <class ‘bool’ >

labelOnly
Values must be of type <class ‘bool’ >

offset
Values must be of type <class ‘str’ >

outline
Values must be of type <class ‘bool’ >

references
Wrap a sequence in an containing object

tagname = 'pivotArea'

type
Value must be one of { ‘data’ , ‘button’ , ‘all’ , ‘origin’ , ‘normal’ , ‘topRight’ , ‘topEnd’ }

class openpyxl.pivot.table.**PivotField**(*items=()*, *autoSortScope=None*, *name=None*, *axis=None*, *dataField=None*, *subtotalCaption=None*, *showDropDowns=True*, *hiddenLevel=None*, *uniqueMemberProperty=None*, *compact=True*, *allDrilled=None*, *numFmtId=None*, *outline=True*, *subtotalTop=True*, *dragToRow=True*, *dragToCol=True*, *multipleItemSelectionAllowed=None*, *dragToPage=True*, *dragToData=True*, *dragOff=True*, *showAll=True*, *insertBlankRow=None*, *serverField=None*, *insertPageBreak=None*, *autoShow=None*, *topAutoShow=True*, *hideNewItems=None*, *measureFilter=None*, *includeNewItemsInFilter=None*, *itemPageCount=10*, *sortType=’manual’*, *dataSourceSort=None*, *nonAutoSortDefault=None*, *rankBy=None*, *defaultSubtotal=True*, *sumSubtotal=None*, *countASubtotal=None*, *avgSubtotal=None*, *maxSubtotal=None*, *minSubtotal=None*, *productSubtotal=None*, *countSubtotal=None*, *stdDevSubtotal=None*, *stdDevPSubtotal=None*, *varSubtotal=None*, *varPSubtotal=None*, *showPropCell=None*, *showPropTip=None*, *showPropAsCaption=None*, *defaultAttributeDrillState=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

allDrilled
Values must be of type <class 'bool' >

autoShow
Values must be of type <class 'bool' >

autoSortScope
Values must be of type <class 'openpyxl.pivot.table.AutoSortScope' >

avgSubtotal
Values must be of type <class 'bool' >

axis
Value must be one of { 'axisCol' , 'axisRow' , 'axisPage' , 'axisValues' }

compact
Values must be of type <class 'bool' >

countASubtotal
Values must be of type <class 'bool' >

countSubtotal
Values must be of type <class 'bool' >

dataField
Values must be of type <class 'bool' >

dataSourceSort
Values must be of type <class 'bool' >

defaultAttributeDrillState
Values must be of type <class 'bool' >

defaultSubtotal
Values must be of type <class 'bool' >

dragOff
Values must be of type <class 'bool' >

dragToCol
Values must be of type <class 'bool' >

dragToData
Values must be of type <class 'bool' >

dragToPage
Values must be of type <class 'bool' >

dragToRow
Values must be of type <class 'bool' >

extLst
Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

hiddenLevel
Values must be of type <class ‘bool’ >

hideNewItems
Values must be of type <class ‘bool’ >

includeNewItemsInFilter
Values must be of type <class ‘bool’ >

insertBlankRow
Values must be of type <class ‘bool’ >

insertPageBreak
Values must be of type <class ‘bool’ >

itemPageCount
Values must be of type <class ‘int’ >

items
Wrap a sequence in an containing object

maxSubtotal
Values must be of type <class ‘bool’ >

measureFilter
Values must be of type <class ‘bool’ >

minSubtotal
Values must be of type <class ‘bool’ >

multipleItemSelectionAllowed
Values must be of type <class ‘bool’ >

name
Values must be of type <class ‘str’ >

nonAutoSortDefault
Values must be of type <class ‘bool’ >

numFmtId
Values must be of type <class ‘int’ >

outline
Values must be of type <class ‘bool’ >

productSubtotal
Values must be of type <class ‘bool’ >

rankBy
  Values must be of type <class 'int'>

serverField
  Values must be of type <class 'bool'>

showAll
  Values must be of type <class 'bool'>

showDropDowns
  Values must be of type <class 'bool'>

showPropAsCaption
  Values must be of type <class 'bool'>

showPropCell
  Values must be of type <class 'bool'>

showPropTip
  Values must be of type <class 'bool'>

sortType
  Value must be one of {'manual', 'descending', 'ascending'}

stdDevPSubtotal
  Values must be of type <class 'bool'>

stdDevSubtotal
  Values must be of type <class 'bool'>

subtotalCaption
  Values must be of type <class 'str'>

subtotalTop
  Values must be of type <class 'bool'>

sumSubtotal
  Values must be of type <class 'bool'>

tagname = 'pivotField'

topAutoShow
  Values must be of type <class 'bool'>

uniqueMemberProperty
  Values must be of type <class 'str'>

varPSubtotal
  Values must be of type <class 'bool'>

varSubtotal
  Values must be of type <class 'bool'>

class `openpyxl.pivot.table.PivotFilter`(*fld=None*, *mpFld=None*, *type=None*, *evalOrder=None*, *id=None*, *iMeasureHier=None*, *iMeasureFld=None*, *name=None*, *description=None*, *stringValue1=None*, *stringValue2=None*, *autoFilter=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

autoFilter

Values must be of type <class 'openpyxl.worksheet.filters.AutoFilter'>

description

Values must be of type <class 'str'>

evalOrder

Values must be of type <class 'int'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

fld

Values must be of type <class 'int'>

iMeasureFld

Values must be of type <class 'int'>

iMeasureHier

Values must be of type <class 'int'>

id

Values must be of type <class 'int'>

mpFld

Values must be of type <class 'int'>

name

Values must be of type <class 'str'>

stringValue1

Values must be of type <class 'str'>

stringValue2

Values must be of type <class 'str'>

tagname = 'filter'

type

Value must be one of {'captionNotBeginsWith', 'valueNotBetween', 'thisYear', 'yesterday', 'valueGreaterThan', 'Q2', 'valueEqual', 'dateEqual', 'valueGreaterThanOrEqual', 'thisMonth', 'valueNotEqual', 'dateNewerThan', 'thisQuarter', 'nextWeek', 'M10', 'M2', 'dateNewerThanOrEqual', 'tomorrow', 'captionNotEndsWith', 'captionGreaterThan', 'dateOlderThan', 'nextQuarter', 'percent', 'M3', 'M8', 'dateBetween', 'captionGreaterThanOrEqual', 'dateNotEqual', 'lastMonth', 'captionBetween', 'captionLessThanOrEqual', 'captionBeginsWith', 'lastYear', 'unknown', 'Q4', 'M9', 'captionNotEqual', 'captionEqual', 'thisWeek', 'M11', 'M5', 'Q1', 'valueBetween', 'dateNotBetween', 'count', 'valueLessThan', 'M6', 'valueLessThanOrEqual', 'captionNotBetween', 'M4', 'M1', 'M7', 'today', 'yearToDate', 'M12', 'sum', 'lastWeek', 'captionEndsWith', 'captionContains', 'captionNotContains', 'nextMonth', 'Q3', 'lastQuarter', 'dateOlderThanOrEqual', 'nextYear', 'captionLessThan' }

`class openpyxl.pivot.table.`**`PivotFilters`**(*count=None*, *filter=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

count

Values must be of type <class 'int' >

filter

Values must be of type <class 'openpyxl.pivot.table.PivotFilter' >

`class openpyxl.pivot.table.`**`PivotHierarchy`**(*outline=None*, *multipleItemSelectionAllowed=None*, *subtotalTop=None*, *showInFieldList=None*, *dragToRow=None*, *dragToCol=None*, *dragToPage=None*, *dragToData=None*, *dragOff=None*, *includeNewItemsInFilter=None*, *caption=None*, *mps=()*, *members=None*, *extLst=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

caption

Values must be of type <class 'str' >

dragOff

Values must be of type <class 'bool' >

dragToCol

Values must be of type <class 'bool' >

dragToData

Values must be of type <class 'bool' >

dragToPage

Values must be of type <class 'bool' >

dragToRow

Values must be of type <class 'bool' >

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

includeNewItemsInFilter

Values must be of type <class 'bool' >

members

Values must be of type <class 'openpyxl.pivot.table.MemberList'>

mps

Wrap a sequence in an containing object

multipleItemSelectionAllowed

Values must be of type <class 'bool'>

outline

Values must be of type <class 'bool'>

showInFieldList

Values must be of type <class 'bool'>

subtotalTop

Values must be of type <class 'bool'>

tagname = 'pivotHierarchy'

class openpyxl.pivot.table.**PivotTableStyle**(*name=None*, *showRowHeaders=None*, *showColHeaders=None*, *showRowStripes=None*, *showColStripes=None*, *showLastColumn=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

name

Values must be of type <class 'str'>

showColHeaders

Values must be of type <class 'bool'>

showColStripes

Values must be of type <class 'bool'>

showLastColumn

Values must be of type <class 'bool'>

showRowHeaders

Values must be of type <class 'bool'>

showRowStripes

Values must be of type <class 'bool'>

tagname = 'pivotTableStyleInfo'

class openpyxl.pivot.table.Reference(*field=None, count=None, selected=None, byPosition=None, relative=None, defaultSubtotal=None, sumSubtotal=None, countASubtotal=None, avgSubtotal=None, maxSubtotal=None, minSubtotal=None, productSubtotal=None, countSubtotal=None, stdDevSubtotal=None, stdDevPSubtotal=None, varSubtotal=None, varPSubtotal=None, x=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

avgSubtotal

Values must be of type <class 'bool'>

byPosition

Values must be of type <class 'bool'>

count

countASubtotal

Values must be of type <class 'bool'>

countSubtotal

Values must be of type <class 'bool'>

defaultSubtotal

Values must be of type <class 'bool'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

field

Values must be of type <class 'int'>

maxSubtotal

Values must be of type <class 'bool'>

minSubtotal

Values must be of type <class 'bool'>

productSubtotal

Values must be of type <class 'bool'>

relative

Values must be of type <class 'bool'>

selected

Values must be of type <class 'bool'>

stdDevPSubtotal

Values must be of type <class 'bool'>

stdDevSubtotal
Values must be of type <class ‘bool’ >

sumSubtotal
Values must be of type <class ‘bool’ >

tagname = 'reference'

varPSubtotal
Values must be of type <class ‘bool’ >

varSubtotal
Values must be of type <class ‘bool’ >

x
Values must be of type <class ‘int’ >

class openpyxl.pivot.table.**RowColField**(*x=None*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

tagname = 'field'

x
Values must be of type <class ‘int’ >

class openpyxl.pivot.table.**RowColItem**(*t='data', r=0, i=0, x=()*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

i
Values must be of type <class ‘int’ >

r
Values must be of type <class ‘int’ >

t
Value must be one of { ‘data’ , ‘sum’ , ‘grand’ , ‘min’ , ‘stdDev’ , ‘max’ , ‘var’ , ‘count’ , ‘stdDevP’ , ‘avg’ , ‘default’ , ‘countA’ , ‘blank’ , ‘varP’ , ‘product’ }

tagname = 'i'

x
A sequence (list or tuple) that may only contain objects of the declared type

class openpyxl.pivot.table.**RowHierarchiesUsage**(*count=None, rowHierarchyUsage=()*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

count

rowHierarchyUsage
A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'rowHierarchiesUsage'

`class openpyxl.pivot.table.TableDefinition`(*name=None, cacheId=None, dataOnRows=False, dataPosition=None, dataCaption=None, grandTotalCaption=None, errorCaption=None, showError=False, missingCaption=None, showMissing=True, pageStyle=None, pivotTableStyle=None, vacatedStyle=None, tag=None, updatedVersion=0, minRefreshableVersion=0, asteriskTotals=False, showItems=True, editData=False, disableFieldList=False, showCalcMbrs=True, visualTotals=True, showMultipleLabel=True, showDataDropDown=True, showDrill=True, printDrill=False, showMemberPropertyTips=True, showDataTips=True, enableWizard=True, enableDrill=True, enableFieldProperties=True, preserveFormatting=True, useAutoFormatting=False, pageWrap=0, pageOverThenDown=False, subtotalHiddenItems=False, rowGrandTotals=True, colGrandTotals=True, fieldPrintTitles=False, itemPrintTitles=False, mergeItem=False, showDropZones=True, createdVersion=0, indent=1, showEmptyRow=False, showEmptyCol=False, showHeaders=True, compact=True, outline=False, outlineData=False, compactData=True, published=False, gridDropZones=False, immersive=True, multipleFieldFilters=None, chartFormat=0, rowHeaderCaption=None, colHeaderCaption=None, fieldListSortAscending=None, mdxSubqueries=None, customListSort=None, autoFormatId=None, applyNumberFormats=False, applyBorderFormats=False, applyFontFormats=False, applyPatternFormats=False, applyAlignmentFormats=False, applyWidthHeightFormats=False, location=None, pivotFields=(), rowFields=(), rowItems=(), colFields=(), colItems=(), pageFields=(), dataFields=(), formats=(), conditionalFormats=(), chartFormats=(), pivotHierarchies=(), pivotTableStyleInfo=None, filters=(), rowHierarchiesUsage=None, colHierarchiesUsage=None, extLst=None, id=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

applyAlignmentFormats
Values must be of type <class 'bool'>

applyBorderFormats
Values must be of type <class 'bool'>

applyFontFormats
Values must be of type <class 'bool'>

applyNumberFormats
Values must be of type <class 'bool'>

applyPatternFormats
Values must be of type <class 'bool'>

applyWidthHeightFormats
Values must be of type <class 'bool'>

asteriskTotals
Values must be of type <class 'bool'>

autoFormatId
Values must be of type <class 'int'>

cache = None

cacheId
Values must be of type <class 'int'>

chartFormat
Values must be of type <class 'int'>

chartFormats
Wrap a sequence in an containing object

colFields
Wrap a sequence in an containing object

colGrandTotals
Values must be of type <class 'bool'>

colHeaderCaption
Values must be of type <class 'str'>

colHierarchiesUsage
Values must be of type <class 'openpyxl.pivot.table.ColHierarchiesUsage'>

colItems
Wrap a sequence in an containing object

compact
Values must be of type <class 'bool'>

compactData
Values must be of type <class ‘bool’ >

conditionalFormats
Wrap a sequence in an containing object

createdVersion
Values must be of type <class ‘int’ >

customListSort
Values must be of type <class ‘bool’ >

dataCaption
Values must be of type <class ‘str’ >

dataFields
Wrap a sequence in an containing object

dataOnRows
Values must be of type <class ‘bool’ >

dataPosition
Values must be of type <class ‘int’ >

disableFieldList
Values must be of type <class ‘bool’ >

editData
Values must be of type <class ‘bool’ >

enableDrill
Values must be of type <class ‘bool’ >

enableFieldProperties
Values must be of type <class ‘bool’ >

enableWizard
Values must be of type <class ‘bool’ >

errorCaption
Values must be of type <class ‘str’ >

extLst
Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

fieldListSortAscending
Values must be of type <class ‘bool’ >

fieldPrintTitles
Values must be of type <class ‘bool’ >

filters
Wrap a sequence in an containing object

formats
Wrap a sequence in an containing object

grandTotalCaption
Values must be of type <class 'str' >

gridDropZones
Values must be of type <class 'bool' >

id
Values must be of type <class 'str' >

immersive
Values must be of type <class 'bool' >

indent
Values must be of type <class 'int' >

itemPrintTitles
Values must be of type <class 'bool' >

location
Values must be of type <class 'openpyxl.pivot.table.Location' >

mdxSubqueries
Values must be of type <class 'bool' >

mergeItem
Values must be of type <class 'bool' >

mime_type = 'application/vnd.openxmlformats-officedocument.spreadsheetml.pivotTable+xml'

minRefreshableVersion
Values must be of type <class 'int' >

missingCaption
Values must be of type <class 'str' >

multipleFieldFilters
Values must be of type <class 'bool' >

name
Values must be of type <class 'str' >

outline
Values must be of type <class 'bool' >

outlineData
Values must be of type <class 'bool' >

pageFields
Wrap a sequence in an containing object

pageOverThenDown
Values must be of type <class 'bool'>

pageStyle
Values must be of type <class 'str'>

pageWrap
Values must be of type <class 'int'>

path

pivotFields
Wrap a sequence in an containing object

pivotHierarchies
Wrap a sequence in an containing object

pivotTableStyle
Values must be of type <class 'str'>

pivotTableStyleInfo
Values must be of type <class 'openpyxl.pivot.table.PivotTableStyle'>

preserveFormatting
Values must be of type <class 'bool'>

printDrill
Values must be of type <class 'bool'>

published
Values must be of type <class 'bool'>

rel_type = 'http://schemas.openxmlformats.org/officeDocument/2006/relationships/pivotTable'

rowFields
Wrap a sequence in an containing object

rowGrandTotals
Values must be of type <class 'bool'>

rowHeaderCaption
Values must be of type <class 'str'>

rowHierarchiesUsage
Values must be of type <class 'openpyxl.pivot.table.RowHierarchiesUsage'>

rowItems
Wrap a sequence in an containing object

`showCalcMbrs`

Values must be of type <class 'bool'>

`showDataDropDown`

Values must be of type <class 'bool'>

`showDataTips`

Values must be of type <class 'bool'>

`showDrill`

Values must be of type <class 'bool'>

`showDropZones`

Values must be of type <class 'bool'>

`showEmptyCol`

Values must be of type <class 'bool'>

`showEmptyRow`

Values must be of type <class 'bool'>

`showError`

Values must be of type <class 'bool'>

`showHeaders`

Values must be of type <class 'bool'>

`showItems`

Values must be of type <class 'bool'>

`showMemberPropertyTips`

Values must be of type <class 'bool'>

`showMissing`

Values must be of type <class 'bool'>

`showMultipleLabel`

Values must be of type <class 'bool'>

`subtotalHiddenItems`

Values must be of type <class 'bool'>

`tag`

Values must be of type <class 'str'>

`tagname = 'pivotTableDefinition'`

`to_tree()`

`updatedVersion`

Values must be of type <class 'int'>

useAutoFormatting
Values must be of type <class 'bool'>

vacatedStyle
Values must be of type <class 'str'>

visualTotals
Values must be of type <class 'bool'>

openpyxl.pivot.table.**tostring**(*element*, ***, *encoding='utf-8'*, *method=None*, *short_empty_elements=True*)
Generate string representation of XML element.

All subelements are included. If encoding is "unicode", a string is returned. Otherwise a bytestring is returned.

element is an Element instance, *encoding* is an optional output encoding defaulting to US-ASCII, *method* is an optional output which can be one of "xml" (default), "html", "text" or "c14n".

Returns an (optionally) encoded string containing the XML data.

openpyxl.reader package

Submodules

openpyxl.reader.drawings module

openpyxl.reader.drawings.**find_images**(*archive*, *path*)
Given the path to a drawing file extract charts and images

Ingore errors due to unsupported parts of DrawingML

openpyxl.reader.excel module

Read an xlsx file into Python

class openpyxl.reader.excel.**ExcelReader**(*fn*, *read_only=False*, *keep_vba=False*, *data_only=False*, *keep_links=True*)
基类：object

Read an Excel package and dispatch the contents to the relevant modules

read()

read_chartsheet(*sheet*, *rel*)

read_manifest()

read_properties()

read_strings()

read_theme()

read_workbook()

read_worksheets()

openpyxl.reader.excel.**load_workbook**(*filename*, *read_only=False*, *keep_vba=False*, *data_only=False*, *keep_links=True*)

Open the given filename and return the workbook

参数

- **filename** (string or a file-like object open in binary mode c.f., zipfile.ZipFile) – the path to open or a file-like object
- **read_only** (*bool*) – optimised for reading, content cannot be edited
- **keep_vba** (*bool*) – preseve vba content (this does NOT mean you can use it)
- **data_only** (*bool*) – controls whether cells with formulae have either the formula (default) or the value stored the last time Excel read the sheet
- **keep_links** (*bool*) – whether links to external workbooks should be preserved. The default is True

返回类型 openpyxl.workbook.Workbook

注解: When using lazy load, all worksheets will be openpyxl.worksheet.iter_worksheet.IterableWorksheet and the returned workbook will be read-only.

openpyxl.reader.strings module

openpyxl.reader.strings.**read_string_table**(*xml_source*)

Read in all shared strings in the table

openpyxl.reader.workbook module

class openpyxl.reader.workbook.**WorkbookParser**(*archive*, *workbook_part_name*, *keep_links=True*)

基类：object

assign_names()

Bind reserved names to parsed worksheets

find_sheets()

Find all sheets in the workbook and return the link to the source file.

Older XLSM files sometimes contain invalid sheet elements. Warn user when these are removed.

`parse()`

`pivot_caches`

Get PivotCache objects

`rels`

openpyxl.styles package

Submodules

openpyxl.styles.alignment module

`class openpyxl.styles.alignment.Alignment`(*horizontal=None, vertical=None, textRotation=0, wrapText=None, shrinkToFit=None, indent=0, relativeIndent=0, justifyLastLine=None, readingOrder=0, text_rotation=None, wrap_text=None, shrink_to_fit=None, mergeCell=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Alignment options for use in styles.

`horizontal`

Value must be one of {'left', 'general', 'justify', 'fill', 'right', 'distributed', 'centerContinuous', 'center'}

`indent`

Values must be of type <class 'float'>

`justifyLastLine`

Values must be of type <class 'bool'>

`readingOrder`

Values must be of type <class 'float'>

`relativeIndent`

Values must be of type <class 'float'>

`shrinkToFit`

Values must be of type <class 'bool'>

`shrink_to_fit`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`tagname = 'alignment'`

textRotation

Value must be one of {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180}

text_rotation

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

vertical

Value must be one of { 'top' , 'bottom' , 'justify' , 'distributed' , 'center' }

wrapText

Values must be of type <class 'bool' >

wrap_text

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

openpyxl.styles.borders module

class openpyxl.styles.borders.Border(*left=None, right=None, top=None, bottom=None, diagonal=None, diagonal_direction=None, vertical=None, horizontal=None, diagonalUp=False, diagonalDown=False, outline=True, start=None, end=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Border positioning for use in styles.

bottom

Values must be of type <class 'openpyxl.styles.borders.Side' >

diagonal

Values must be of type <class 'openpyxl.styles.borders.Side' >

diagonalDown

Values must be of type <class 'bool' >

diagonalUp

Values must be of type <class 'bool' >

end

Values must be of type <class 'openpyxl.styles.borders.Side' >

horizontal

Values must be of type <class ‘openpyxl.styles.borders.Side’ >

left

Values must be of type <class ‘openpyxl.styles.borders.Side’ >

outline

Values must be of type <class ‘bool’ >

right

Values must be of type <class ‘openpyxl.styles.borders.Side’ >

start

Values must be of type <class ‘openpyxl.styles.borders.Side’ >

tagname = 'border'

top

Values must be of type <class ‘openpyxl.styles.borders.Side’ >

vertical

Values must be of type <class ‘openpyxl.styles.borders.Side’ >

class openpyxl.styles.borders.**Side**(*style=None*, *color=None*, *border_style=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Border options for use in styles. Caution: if you do not specify a border_style, other attributes will have no effect !

border_style

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

color

Values must be of type <class ‘openpyxl.styles.colors.Color’ >

style

Value must be one of { ‘double’ , ‘medium’ , ‘thin’ , ‘slantDashDot’ , ‘thick’ , ‘dashDot’ , ‘dotted’ , ‘mediumDashDot’ , ‘mediumDashDotDot’ , ‘mediumDashed’ , ‘dashed’ , ‘dashDotDot’ , ‘hair’ }

openpyxl.styles.builtins module

openpyxl.styles.cell_style module

class openpyxl.styles.cell_style.**ArrayDescriptor**(*key*)

基类：object

class openpyxl.styles.cell_style.**CellStyle**(*numFmtId=0*, *fontId=0*, *fillId=0*, *borderId=0*, *xfId=None*, *quotePrefix=None*, *pivotButton=None*, *applyNumberFormat=None*, *applyFont=None*, *applyFill=None*, *applyBorder=None*, *applyAlignment=None*, *applyProtection=None*, *alignment=None*, *protection=None*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

alignment

Values must be of type <class ‘openpyxl.styles.alignment.Alignment’ >

applyAlignment

applyBorder

Values must be of type <class ‘bool’ >

applyFill

Values must be of type <class ‘bool’ >

applyFont

Values must be of type <class ‘bool’ >

applyNumberFormat

Values must be of type <class ‘bool’ >

applyProtection

borderId

Values must be of type <class ‘int’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

fillId

Values must be of type <class ‘int’ >

fontId

Values must be of type <class ‘int’ >

classmethod from_array(*style*)

Convert from StyleArray

numFmtId

Values must be of type <class ‘int’ >

pivotButton

Values must be of type <class ‘bool’ >

protection

Values must be of type <class ‘openpyxl.styles.protection.Protection’ >

quotePrefix
Values must be of type <class ‘bool’ >

tagname = 'xf'

to_array()
Convert to StyleArray

xfId
Values must be of type <class ‘int’ >

class openpyxl.styles.cell_style.**CellStyleList**(*count=None*, *xf=()*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

alignment
A sequence (list or tuple) that may only contain objects of the declared type

count

protection
A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'cellXfs'

xf
A sequence (list or tuple) that may only contain objects of the declared type

class openpyxl.styles.cell_style.**StyleArray**
基类：array.array

Simplified named tuple with an array

alignmentId

borderId

fillId

fontId

numFmtId

pivotButton

protectionId

quotePrefix

tagname = 'xf'

xfId

openpyxl.styles.colors module

class openpyxl.styles.colors.**Color**(*rgb='00000000', indexed=None, auto=None, theme=None, tint=0.0, index=None, type='rgb'*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Named colors for use in styles.

auto

Values must be of type <class 'bool'>

index

indexed

Values must be of type <class 'int'>

rgb

Values must be of type <class 'str'>

tagname = 'color'

theme

Values must be of type <class 'int'>

tint

Values must be of type <class 'float'>

type

Values must be of type <class 'str'>

value

class openpyxl.styles.colors.**ColorDescriptor**(**args, **kw*)

基类：*openpyxl.descriptors.base.Typed*

expected_type

Color 的别名

class openpyxl.styles.colors.**ColorList**(*indexedColors=(), mruColors=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

index

indexedColors

Wrap a sequence in an containing object

mruColors

Wrap a sequence in an containing object

tagname = 'colors'

class openpyxl.styles.colors.**RGB**(**args, **kw*)

基类：*openpyxl.descriptors.base.Typed*

Descriptor for aRGB values If not supplied alpha is 00

expected_type

`builtins.str` 的别名

class openpyxl.styles.colors.**RgbColor**(*rgb=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

rgb

Values must be of type <class 'str'>

tagname = 'rgbColor'

openpyxl.styles.differential module

class openpyxl.styles.differential.**DifferentialStyle**(*font=None, numFmt=None, fill=None, alignment=None, border=None, protection=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

alignment

Values must be of type <class 'openpyxl.styles.alignment.Alignment'>

border

Values must be of type <class 'openpyxl.styles.borders.Border'>

fill

Values must be of type <class 'openpyxl.styles.fills.Fill'>

font

Values must be of type <class 'openpyxl.styles.fonts.Font'>

numFmt

Values must be of type <class 'openpyxl.styles.numbers.NumberFormat'>

protection

Values must be of type <class 'openpyxl.styles.protection.Protection'>

tagname = 'dxf'

class openpyxl.styles.differential.**DifferentialStyleList**(*dxf=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Deduping container for differential styles.

add(*dxf*)

Add a differential style and return its index

append(*dxf*)

Check to see whether style already exists and append it if does not.

`dxf`

A sequence (list or tuple) that may only contain objects of the declared type

`styles`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

`tagname = 'dxfs'`

openpyxl.styles.fills module

`class openpyxl.styles.fills.Fill`

基类：*openpyxl.descriptors.serialisable.Serialisable*

Base class

`classmethod from_tree(el)`

Create object from XML

`tagname = 'fill'`

`class openpyxl.styles.fills.GradientFill(type='linear', degree=0, left=0, right=0, top=0, bottom=0, stop=())`

基类：*openpyxl.styles.fills.Fill*

Fill areas with gradient

Two types of gradient fill are supported:

- A type=’ linear’ gradient interpolates colours between a set of specified Stops, across the length of an area. The gradient is left-to-right by default, but this orientation can be modified with the degree attribute. A list of Colors can be provided instead and they will be positioned with equal distance between them.
- A type=’ path’ gradient applies a linear gradient from each edge of the area. Attributes top, right, bottom, left specify the extent of fill from the respective borders. Thus top=” 0.2” will fill the top 20% of the cell.

`bottom`

Values must be of type <class ‘float’ >

`degree`

Values must be of type <class ‘float’ >

`fill_type`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

`left`

Values must be of type <class ‘float’ >

right

Values must be of type <class 'float'>

stop

tagname = 'gradientFill'

to_tree(*tagname=None*, *namespace=None*, *idx=None*)

top

Values must be of type <class 'float'>

type

Value must be one of {'path', 'linear'}

class openpyxl.styles.fills.PatternFill(*patternType=None*, *fgColor=<openpyxl.styles.colors.Color object> Parameters: rgb='00000000'*, *indexed=None*, *auto=None*, *theme=None*, *tint=0.0*, *type='rgb'*, *bgColor=<openpyxl.styles.colors.Color object> Parameters: rgb='00000000'*, *indexed=None*, *auto=None*, *theme=None*, *tint=0.0*, *type='rgb'*, *fill_type=None*, *start_color=None*, *end_color=None*)

基类：*openpyxl.styles.fills.Fill*

Area fill patterns for use in styles. Caution: if you do not specify a fill_type, other attributes will have no effect !

bgColor

Values must be of type <class 'openpyxl.styles.colors.Color'>

end_color

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

fgColor

Values must be of type <class 'openpyxl.styles.colors.Color'>

fill_type

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

patternType

Value must be one of {'lightDown', 'darkUp', 'lightTrellis', 'lightUp', 'lightVertical', 'mediumGray', 'lightGray', 'darkVertical', 'darkGrid', 'darkTrellis', 'gray0625', 'gray125', 'darkHorizontal', 'solid', 'lightHorizontal', 'lightGrid', 'darkGray', 'darkDown'}

start_color

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

tagname = 'patternFill'

to_tree(*tagname=None*, *idx=None*)

class openpyxl.styles.fills.**Stop**(*color*, *position*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

color

Values must be of type <class ‘openpyxl.styles.colors.Color’ >

position

Values must be of type <class ‘float’ >

tagname = 'stop'

class openpyxl.styles.fills.**StopList**(*name=None*, ***kw*)

基类：*openpyxl.descriptors.sequence.Sequence*

expected_type

Stop 的别名

openpyxl.styles.fonts module

class openpyxl.styles.fonts.**Font**(*name=None*, *sz=None*, *b=None*, *i=None*, *charset=None*, *u=None*, *strike=None*, *color=None*, *scheme=None*, *family=None*, *size=None*, *bold=None*, *italic=None*, *strikethrough=None*, *underline=None*, *vertAlign=None*, *outline=None*, *shadow=None*, *condense=None*, *extend=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Font options used in styles.

UNDERLINE_DOUBLE = 'double'

UNDERLINE_DOUBLE_ACCOUNTING = 'doubleAccounting'

UNDERLINE_SINGLE = 'single'

UNDERLINE_SINGLE_ACCOUNTING = 'singleAccounting'

b

Values must be of type <class ‘bool’ >

bold

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

charset

Values must be of type <class ‘int’ >

color

Values must be of type <class 'openpyxl.styles.colors.Color'>

condense

Values must be of type <class 'bool'>

extend

Values must be of type <class 'bool'>

family

Values must be of type <class 'float'>

classmethod from_tree(*node*)

Set default value for underline if child element is present

i

Values must be of type <class 'bool'>

italic

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

name

Values must be of type <class 'str'>

outline

Values must be of type <class 'bool'>

scheme

Value must be one of {'major', 'minor'}

shadow

Values must be of type <class 'bool'>

size

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

strike

Values must be of type <class 'bool'>

strikethrough

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

sz

Values must be of type <class 'float'>

tagname = 'font'

u

Value must be one of {‘doubleAccounting’, ‘double’, ‘singleAccounting’, ‘single’}

underline

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. “type”) or a more descriptve name is desired (eg. “underline” for “u”)

vertAlign

Value must be one of {‘subscript’, ‘baseline’, ‘superscript’}

openpyxl.styles.named_styles module

`class openpyxl.styles.named_styles.NamedStyle`(*name='Normal', font=<openpyxl.styles.fonts.Font object> Parameters: name=None, charset=None, family=None, b=False, i=False, strike=None, outline=None, shadow=None, condense=None, color=None, extend=None, sz=None, u=None, vertAlign=None, scheme=None, fill=<openpyxl.styles.fills.PatternFill object> Parameters: patternType=None, fgColor=<openpyxl.styles.colors.Color object> Parameters: rgb='00000000', indexed=None, auto=None, theme=None, tint=0.0, type='rgb', bgColor=<openpyxl.styles.colors.Color object> Parameters: rgb='00000000', indexed=None, auto=None, theme=None, tint=0.0, type='rgb', border=<openpyxl.styles.borders.Border object> Parameters: outline=True, diagonalUp=False, diagonalDown=False, start=None, end=None, left=None, right=None, top=None, bottom=None, diagonal=None, vertical=None, horizontal=None, alignment=<openpyxl.styles.alignment.Alignment object> Parameters: horizontal=None, vertical=None, textRotation=0, wrapText=None, shrinkToFit=None, indent=0.0, relativeIndent=0.0, justifyLastLine=None, readingOrder=0.0, number_format=None, protection=<openpyxl.styles.protection.Protection object> Parameters: locked=True, hidden=False, builtinId=None, hidden=False, xfId=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Named and editable styles

alignment

Values must be of type <class 'openpyxl.styles.alignment.Alignment' >

as_name()

Return relevant named style

as_tuple()

Return a style array representing the current style

as_xf()

Return equivalent XfStyle

bind(*wb*)

Bind a named style to a workbook

border

Values must be of type <class 'openpyxl.styles.borders.Border'>

builtinId

Values must be of type <class 'int'>

fill

Values must be of type <class 'openpyxl.styles.fills.Fill'>

font

Values must be of type <class 'openpyxl.styles.fonts.Font'>

hidden

Values must be of type <class 'bool'>

name

Values must be of type <class 'str'>

number_format

Values must be of type <class 'str'>

protection

Values must be of type <class 'openpyxl.styles.protection.Protection'>

xfId

Index of the style in the list of named styles

class openpyxl.styles.named_styles.**NamedStyleList**

基类：list

Named styles are editable and can be applied to multiple objects

As only the index is stored in referencing objects the order mus be preserved.

append(*style*)

Append object to the end of the list.

names

openpyxl.styles.numbers module

class openpyxl.styles.numbers.**NumberFormat**(*numFmtId=None*, *formatCode=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

formatCode

Values must be of type <class ‘str’ >

numFmtId

Values must be of type <class ‘int’ >

class openpyxl.styles.numbers.**NumberFormatDescriptor**(**args*, ***kw*)

基类：*openpyxl.descriptors.base.String*

class openpyxl.styles.numbers.**NumberFormatList**(*count=None*, *numFmt=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

count

numFmt

A sequence (list or tuple) that may only contain objects of the declared type

openpyxl.styles.numbers.**builtin_format_code**(*index*)

Return one of the standard format codes by index.

openpyxl.styles.numbers.**builtin_format_id**(*fmt*)

Return the id of a standard style.

openpyxl.styles.numbers.**is_builtin**(*fmt*)

openpyxl.styles.numbers.**is_date_format**(*fmt*)

openpyxl.styles.numbers.**is_datetime**(*fmt*)

Return date, time or datetime

openpyxl.styles.numbers.**is_timedelta_format**(*fmt*)

openpyxl.styles.protection module

class openpyxl.styles.protection.**Protection**(*locked=True*, *hidden=False*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Protection options for use in styles.

hidden

Values must be of type <class ‘bool’ >

locked

Values must be of type <class ‘bool’ >

tagname = 'protection'

openpyxl.styles.proxy module

class openpyxl.styles.proxy.**StyleProxy**(*target*)

基类：object

Proxy formatting objects so that they cannot be altered

copy(***kw*)

Return a copy of the proxied object. Keyword args will be passed through

注解： Deprecated: Use copy(obj) or cell.obj = cell.obj + other

openpyxl.styles.styleable module

class openpyxl.styles.styleable.**NamedStyleDescriptor**

基类：object

collection = '_named_styles'

key = 'xfId'

class openpyxl.styles.styleable.**NumberFormatDescriptor**

基类：object

collection = '_number_formats'

key = 'numFmtId'

class openpyxl.styles.styleable.**StyleArrayDescriptor**(*key*)

基类：object

class openpyxl.styles.styleable.**StyleDescriptor**(*collection*, *key*)

基类：object

class openpyxl.styles.styleable.**StyleableObject**(*sheet*, *style_array=None*)

基类：object

Base class for styleble objects implementing proxy and lookup functions

alignment

border

fill

font

has_style

number_format

`parent`

`pivotButton`

`protection`

`quotePrefix`

`style`

`style_id`

openpyxl.styles.stylesheet module

class `openpyxl.styles.stylesheet.Stylesheet`(*numFmts=None, fonts=(), fills=(), borders=(), cellStyleXfs=None, cellXfs=None, cellStyles=None, dxfs=(), tableStyles=None, colors=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

borders

Wrap a sequence in an containing object

cellStyleXfs

Values must be of type <class 'openpyxl.styles.cell_style.CellStyleList'>

cellStyles

Values must be of type <class 'openpyxl.styles.named_styles._NamedCellStyleList'>

cellXfs

Values must be of type <class 'openpyxl.styles.cell_style.CellStyleList'>

colors

Values must be of type <class 'openpyxl.styles.colors.ColorList'>

custom_formats

dxfs

Wrap a sequence in an containing object

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

fills

Wrap a sequence in an containing object

fonts

Wrap a sequence in an containing object

classmethod from_tree(*node*)

Create object from XML

numFmts
Values must be of type <class 'openpyxl.styles.numbers.NumberFormatList'>

tableStyles
Values must be of type <class 'openpyxl.styles.table.TableStyleList'>

tagname = 'styleSheet'

to_tree(*tagname=None*, *idx=None*, *namespace=None*)

openpyxl.styles.stylesheet.**apply_stylesheet**(*archive*, *wb*)
Add styles to workbook if present

openpyxl.styles.stylesheet.**write_stylesheet**(*wb*)

openpyxl.styles.table module

class openpyxl.styles.table.**TableStyle**(*name=None*, *pivot=None*, *table=None*, *count=None*, *tableStyleElement=()*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

count
Values must be of type <class 'int'>

name
Values must be of type <class 'str'>

pivot
Values must be of type <class 'bool'>

table
Values must be of type <class 'bool'>

tableStyleElement
A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'tableStyle'

class openpyxl.styles.table.**TableStyleElement**(*type=None*, *size=None*, *dxfId=None*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

dxfId
Values must be of type <class 'int'>

size
Values must be of type <class 'int'>

tagname = 'tableStyleElement'

type
Value must be one of {'totalRow', 'lastColumn', 'lastHeaderCell', 'thirdSubtotalColumn'

, 'secondSubtotalRow', 'wholeTable', 'firstColumnStripe', 'thirdSubtotalRow', 'firstRowStripe', 'blankRow', 'thirdColumnSubheading', 'secondRowSubheading', 'lastTotalCell', 'firstHeaderCell', 'firstColumn', 'pageFieldLabels', 'firstTotalCell', 'thirdRowSubheading', 'secondColumnStripe', 'firstColumnSubheading', 'firstSubtotalColumn', 'headerRow', 'secondColumnSubheading', 'secondRowStripe', 'firstRowSubheading', 'secondSubtotalColumn', 'pageFieldValues', 'firstSubtotalRow'}

`class openpyxl.styles.table.TableStyleList`(*count=None, defaultTableStyle='TableStyleMedium9', defaultPivotStyle='PivotStyleLight16', tableStyle=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`count`

`defaultPivotStyle`

Values must be of type <class 'str'>

`defaultTableStyle`

Values must be of type <class 'str'>

`tableStyle`

A sequence (list or tuple) that may only contain objects of the declared type

`tagname = 'tableStyles'`

openpyxl.utils package

Submodules

openpyxl.utils.bound_dictionary module

`class openpyxl.utils.bound_dictionary.BoundDictionary`(*reference=None, *args, **kw*)

基类：`collections.defaultdict`

A default dictionary where elements are tightly coupled.

The factory method is responsible for binding the parent object to the child.

If a reference attribute is assigned then child objects will have the key assigned to this.

Otherwise it's just a defaultdict.

openpyxl.utils.cell module

Collection of utilities used within the package and also available for client code

`openpyxl.utils.cell.absolute_coordinate`(*coord_string*)

Convert a coordinate to an absolute coordinate string (B12 -> $B$12) `openpyxl.utils.cell.`**`cols_from_range`**`(`*range_string*`)`

Get individual addresses for every cell in a range. Yields one row at a time.

`openpyxl.utils.cell.`**`column_index_from_string`**`(`*str_col*`)`

Convert a column name into a numerical index (‘A’ -> 1)

`openpyxl.utils.cell.`**`coordinate_from_string`**`(`*coord_string*`)`

Convert a coordinate string like ‘B12’ to a tuple (‘B’ , 12)

`openpyxl.utils.cell.`**`coordinate_to_tuple`**`(`*coordinate*`)`

Convert an Excel style coordinate to (row, colum) tuple

`openpyxl.utils.cell.`**`get_column_interval`**`(`*start*, *end*`)`

Given the start and end columns, return all the columns in the series.

The start and end columns can be either column letters or 1-based indexes.

`openpyxl.utils.cell.`**`get_column_letter`**`(`*idx*`)`

Convert a column index into a column letter (3 -> ‘C’)

`openpyxl.utils.cell.`**`quote_sheetname`**`(`*sheetname*`)`

Add quotes around sheetnames if they contain spaces.

`openpyxl.utils.cell.`**`range_boundaries`**`(`*range_string*`)`

Convert a range string into a tuple of boundaries: (min_col, min_row, max_col, max_row) Cell coordinates will be converted into a range with the cell at both end

`openpyxl.utils.cell.`**`range_to_tuple`**`(`*range_string*`)`

Convert a worksheet range to the sheetname and maximum and minimum coordinate indices

`openpyxl.utils.cell.`**`rows_from_range`**`(`*range_string*`)`

Get individual addresses for every cell in a range. Yields one row at a time.

openpyxl.utils.dataframe module

`openpyxl.utils.dataframe.`**`dataframe_to_rows`**`(`*df*, *index=True*, *header=True*`)`

Convert a Pandas dataframe into something suitable for passing into a worksheet. If index is True then the index will be included, starting one row below the header. If header is True then column headers will be included starting one column to the right. Formatting should be done by client code.

`openpyxl.utils.dataframe.`**`expand_index`**`(`*index*, *header=False*`)`

Expand axis or column Multiindex For columns use header = True For axes use header = False (default)

openpyxl.utils.datetime module

`openpyxl.utils.datetime.`**`days_to_time`**`(`*value*`)` `openpyxl.utils.datetime.from_ISO8601`(*formatted_string*)

Convert from a timestamp string to a datetime object. According to 18.17.4 in the specification the following ISO 8601 formats are supported.

Dates B.1.1 and B.2.1 Times B.1.2 and B.2.2 Datetimes B.1.3 and B.2.3

There is no concept of timedeltas in the specification, but Excel writes them (in strict OOXML mode), so these are also understood.

`openpyxl.utils.datetime.from_excel`(*value*, *epoch=datetime.datetime(1899*, *12*, *30*, *0*, *0)*, *timedelta=False*)

Convert Excel serial to Python datetime

`openpyxl.utils.datetime.time_to_days`(*value*)

Convert a time value to fractions of day

`openpyxl.utils.datetime.timedelta_to_days`(*value*)

Convert a timedelta value to fractions of a day

`openpyxl.utils.datetime.to_ISO8601`(*dt*)

Convert from a datetime to a timestamp string.

`openpyxl.utils.datetime.to_excel`(*dt*, *epoch=datetime.datetime(1899*, *12*, *30*, *0*, *0)*)

Convert Python datetime to Excel serial

openpyxl.utils.escape module

OOXML has non-standard escaping for characters <

`openpyxl.utils.escape.escape`(*value*)

Convert ASCII < 31 to OOXML: n == _x + hex(ord(n)) + _

`openpyxl.utils.escape.unescape`(*value*)

Convert escaped strings to ASCIII: _x000a_ == n

openpyxl.utils.exceptions module

Definitions for openpyxl shared exception classes.

exception `openpyxl.utils.exceptions.CellCoordinatesException`

基类：`Exception`

Error for converting between numeric and A1-style cell references.

exception `openpyxl.utils.exceptions.IllegalCharacterError`

基类：`Exception`

The data submitted which cannot be used directly in Excel files. It must be removed or escaped.

`exception openpyxl.utils.exceptions.InvalidFileException`

基类：`Exception`

Error for trying to open a non-ooxml file.

`exception openpyxl.utils.exceptions.NamedRangeException`

基类：`Exception`

Error for badly formatted named ranges.

`exception openpyxl.utils.exceptions.ReadOnlyWorkbookException`

基类：`Exception`

Error for trying to modify a read-only workbook

`exception openpyxl.utils.exceptions.SheetTitleException`

基类：`Exception`

Error for bad sheet names.

`exception openpyxl.utils.exceptions.WorkbookAlreadySaved`

基类：`Exception`

Error when attempting to perform operations on a dump workbook while it has already been dumped once

openpyxl.utils.formulas module

List of builtin formulae

openpyxl.utils.indexed_list module

`class openpyxl.utils.indexed_list.IndexedList(`*iterable=None*`)`

基类：`list`

List with optimised access by value Based on Alex Martelli’s recipe

http://code.activestate.com/recipes/52303-the-auxiliary-dictionary-idiom-for-sequences-with-/

add(*value*)

append(*value*)

Append object to the end of the list.

index(*value*)

Return first index of value.

Raises ValueError if the value is not present.

openpyxl.utils.inference module

Type inference functions

openpyxl.utils.inference.**cast_numeric**(*value*)

Explicity convert a string to a numeric value

openpyxl.utils.inference.**cast_percentage**(*value*)

Explicitly convert a string to numeric value and format as a percentage

openpyxl.utils.inference.**cast_time**(*value*)

Explicitly convert a string to a number and format as datetime or time

openpyxl.utils.protection module

openpyxl.utils.protection.**hash_password**(*plaintext_password=”*)

Create a password hash from a given string for protecting a worksheet only. This will not work for encrypting a workbook.

This method is based on the algorithm provided by Daniel Rentz of OpenOffice and the PEAR package Spreadsheet_Excel_Writer by Xavier Noguer <xnoguer@rezebra.com>. See also http://blogs.msdn.com/b/ericwhite/archive/2008/02/23/the-legacy-hashing-algorithm-in-open-xml.aspx

openpyxl.utils.units module

openpyxl.utils.units.**DEFAULT_HEADER = 0.3**

From the ECMA Spec (4th Edition part 1) Page setup: "Left Page Margin in inches" p. 1647

Docs from http://startbigthinksmall.wordpress.com/2010/01/04/points-inches-and-emus-measuring-units-in-office-ope

See also http://msdn.microsoft.com/en-us/library/dd560821(v=office.12).aspx

dxa: The main unit in OOXML is a twentieth of a point. Also called twips. pt: point. In Excel there are 72 points to an inch hp: half-points are used to specify font sizes. A font-size of 12pt equals 24 half points pct: Half-points are used to specify font sizes. A font-size of 12pt equals 24 half points

EMU: English Metric Unit, EMUs are used for coordinates in vector-based drawings and embedded pictures. One inch equates to 914400 EMUs and a centimeter is 360000. For bitmaps the default resolution is 96 dpi (known as PixelsPerInch in Excel). Spec p. 1122

For radial geometry Excel uses integert units of 1/60000th of a degree.

openpyxl.utils.units.**EMU_to_cm**(*value*)

openpyxl.utils.units.**EMU_to_inch**(*value*)

openpyxl.utils.units.**EMU_to_pixels**(*value*)

openpyxl.utils.units.**angle_to_degrees**(*value*) `openpyxl.utils.units.cm_to_EMU`(*value*)

1 cm = 360000 EMUs

`openpyxl.utils.units.cm_to_dxa`(*value*)

`openpyxl.utils.units.degrees_to_angle`(*value*)

1 degree = 60000 angles

`openpyxl.utils.units.dxa_to_cm`(*value*)

`openpyxl.utils.units.dxa_to_inch`(*value*)

`openpyxl.utils.units.inch_to_EMU`(*value*)

1 inch = 914400 EMUs

`openpyxl.utils.units.inch_to_dxa`(*value*)

1 inch = 72 * 20 dxa

`openpyxl.utils.units.pixels_to_EMU`(*value*)

1 pixel = 9525 EMUs

`openpyxl.utils.units.pixels_to_points`(*value*, *dpi=96*)

96 dpi, 72i

`openpyxl.utils.units.points_to_pixels`(*value*, *dpi=96*)

`openpyxl.utils.units.short_color`(*color*)

format a color to its short size

openpyxl.workbook package

Subpackages

openpyxl.workbook.external_link package

Submodules

openpyxl.workbook.external_link.external module

`class openpyxl.workbook.external_link.external.ExternalBook`(*sheetNames=None*, *definedNames=()*, *sheetDataSet=None*, *id=None*)

基类： *openpyxl.descriptors.serialisable.Serialisable*

`definedNames`

Wrap a sequence in an containing object

`id`

Values must be of type <class ‘str’ >

sheetDataSet

Values must be of type <class ‘openpyxl.workbook.external_link.external.ExternalSheetDataSet’ >

sheetNames

Values must be of type <class ‘openpyxl.workbook.external_link.external.ExternalSheetNames’ >

tagname = 'externalBook'

class openpyxl.workbook.external_link.external.**ExternalCell**(*r=None*, *t=None*, *vm=None*, *v=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

r

Values must be of type <class ‘str’ >

t

Value must be one of { ‘d’ , ‘b’ , ‘e’ , ‘s’ , ‘n’ , ‘str’ , ‘inlineStr’ }

v

Values must be of type <class ‘str’ >

vm

Values must be of type <class ‘int’ >

class openpyxl.workbook.external_link.external.**ExternalDefinedName**(*name=None*, *refersTo=None*, *sheetId=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

name

Values must be of type <class ‘str’ >

refersTo

Values must be of type <class ‘str’ >

sheetId

Values must be of type <class ‘int’ >

tagname = 'definedName'

class openpyxl.workbook.external_link.external.**ExternalLink**(*externalBook=None*, *ddeLink=None*, *oleLink=None*, *extLst=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

externalBook

Values must be of type <class ‘openpyxl.workbook.external_link.external.ExternalBook’ >

file_link

Values must be of type <class ‘openpyxl.packaging.relationship.Relationship’ >

`mime_type = 'application/vnd.openxmlformats-officedocument.spreadsheetml.externalLink+xml'`

`path`

`tagname = 'externalLink'`

`to_tree()`

`class openpyxl.workbook.external_link.external.`**`ExternalRow`**`(`*r=(), cell=None*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`cell`

A sequence (list or tuple) that may only contain objects of the declared type

`r`

Values must be of type <class ‘int’ >

`class openpyxl.workbook.external_link.external.`**`ExternalSheetData`**`(`*sheetId=None, refreshError=None, row=()*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`refreshError`

Values must be of type <class ‘bool’ >

`row`

A sequence (list or tuple) that may only contain objects of the declared type

`sheetId`

Values must be of type <class ‘int’ >

`class openpyxl.workbook.external_link.external.`**`ExternalSheetDataSet`**`(`*sheetData=None*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`sheetData`

A sequence (list or tuple) that may only contain objects of the declared type

`class openpyxl.workbook.external_link.external.`**`ExternalSheetNames`**`(`*sheetName=()*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`sheetName`

A sequence of primitive types that are stored as a single attribute. “val” is the default attribute

`openpyxl.workbook.external_link.external.`**`read_external_link`**`(`*archive, book_path*`)`

Submodules

openpyxl.workbook.child module

`openpyxl.workbook.child.`**`avoid_duplicate_name`**`(`*names, value*`)`

Naive check to see whether name already exists. If name does exist suggest a name using an incrementer Duplicates are case insensitive

openpyxl.workbook.defined_name module

class openpyxl.workbook.defined_name.**DefinedName**(*name=None, comment=None, customMenu=None, description=None, help=None, statusBar=None, localSheetId=None, hidden=None, function=None, vbProcedure=None, xlm=None, functionGroupId=None, shortcutKey=None, publishToServer=None, workbookParameter=None, attr_text=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

attr_text

comment

Values must be of type <class 'str' >

customMenu

Values must be of type <class 'str' >

description

Values must be of type <class 'str' >

destinations

function

Values must be of type <class 'bool' >

functionGroupId

Values must be of type <class 'int' >

help

Values must be of type <class 'str' >

hidden

Values must be of type <class 'bool' >

is_external

is_reserved

localSheetId

Values must be of type <class 'int' >

name

Values must be of type <class 'str' >

publishToServer

Values must be of type <class 'bool' >

shortcutKey
Values must be of type <class 'str' >

statusBar
Values must be of type <class 'str' >

tagname = 'definedName'

type

value
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

vbProcedure
Values must be of type <class 'bool' >

workbookParameter
Values must be of type <class 'bool' >

xlm
Values must be of type <class 'bool' >

class openpyxl.workbook.defined_name.**DefinedNameList**(*definedName=()*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

append(*defn*)

definedName
A sequence (list or tuple) that may only contain objects of the declared type

delete(*name, scope=None*)
Delete a name assigned to a specific or global

get(*name, scope=None*)
Get the name assigned to a specicic sheet or global

localnames(*scope*)
Provide a list of all names for a particular worksheet

tagname = 'definedNames'

openpyxl.workbook.external_reference module

class openpyxl.workbook.external_reference.**ExternalReference**(*id*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

id
Values must be of type <class 'str' >

tagname = 'externalReference'

openpyxl.workbook.function_group module

class openpyxl.workbook.function_group.**FunctionGroup**(*name=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

name

Values must be of type <class 'str'>

tagname = 'functionGroup'

class openpyxl.workbook.function_group.**FunctionGroupList**(*builtInGroupCount=16*, *functionGroup=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

builtInGroupCount

Values must be of type <class 'int'>

functionGroup

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'functionGroups'

openpyxl.workbook.properties module

class openpyxl.workbook.properties.**CalcProperties**(*calcId=124519*, *calcMode=None*, *fullCalcOnLoad=True*, *refMode=None*, *iterate=None*, *iterateCount=None*, *iterateDelta=None*, *fullPrecision=None*, *calcCompleted=None*, *calcOnSave=None*, *concurrentCalc=None*, *concurrentManualCount=None*, *forceFullCalc=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

calcCompleted

Values must be of type <class 'bool'>

calcId

Values must be of type <class 'int'>

calcMode

Value must be one of {'autoNoTable', 'manual', 'auto'}

calcOnSave

Values must be of type <class 'bool'>

concurrentCalc

Values must be of type <class 'bool'>

concurrentManualCount

Values must be of type <class 'int' >

forceFullCalc

Values must be of type <class 'bool' >

fullCalcOnLoad

Values must be of type <class 'bool' >

fullPrecision

Values must be of type <class 'bool' >

iterate

Values must be of type <class 'bool' >

iterateCount

Values must be of type <class 'int' >

iterateDelta

Values must be of type <class 'float' >

refMode

Value must be one of { 'A1' , 'R1C1' }

tagname = 'calcPr'

class openpyxl.workbook.properties.**FileVersion**(*appName=None*, *lastEdited=None*, *lowestEdited=None*, *rupBuild=None*, *codeName=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

appName

Values must be of type <class 'str' >

codeName

lastEdited

Values must be of type <class 'str' >

lowestEdited

Values must be of type <class 'str' >

rupBuild

Values must be of type <class 'str' >

tagname = 'fileVersion'

class openpyxl.workbook.properties.WorkbookProperties(*date1904=None, dateCompatibility=None, showObjects=None, showBorderUnselectedTables=None, filterPrivacy=None, promptedSolutions=None, showInkAnnotation=None, backupFile=None, saveExternalLinkValues=None, updateLinks=None, codeName=None, hidePivotFieldList=None, showPivotChartFilter=None, allowRefreshQuery=None, publishItems=None, checkCompatibility=None, autoCompressPictures=None, refreshAllConnections=None, defaultThemeVersion=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

allowRefreshQuery

Values must be of type <class ‘bool’ >

autoCompressPictures

Values must be of type <class ‘bool’ >

backupFile

Values must be of type <class ‘bool’ >

checkCompatibility

Values must be of type <class ‘bool’ >

codeName

Values must be of type <class ‘str’ >

date1904

Values must be of type <class ‘bool’ >

dateCompatibility

Values must be of type <class ‘bool’ >

defaultThemeVersion

Values must be of type <class ‘int’ >

filterPrivacy

Values must be of type <class ‘bool’ >

hidePivotFieldList

Values must be of type <class ‘bool’ >

promptedSolutions

Values must be of type <class ‘bool’ >

publishItems
Values must be of type <class 'bool'>

refreshAllConnections
Values must be of type <class 'bool'>

saveExternalLinkValues
Values must be of type <class 'bool'>

showBorderUnselectedTables
Values must be of type <class 'bool'>

showInkAnnotation
Values must be of type <class 'bool'>

showObjects
Value must be one of {'all', 'placeholders'}

showPivotChartFilter
Values must be of type <class 'bool'>

tagname = 'workbookPr'

updateLinks
Value must be one of {'userSet', 'always', 'never'}

openpyxl.workbook.protection module

openpyxl.workbook.protection.**DocumentSecurity**
openpyxl.workbook.protection.WorkbookProtection 的别名

class openpyxl.workbook.protection.**FileSharing**(*readOnlyRecommended=None, userName=None, reservationPassword=None, algorithmName=None, hashValue=None, saltValue=None, spinCount=None*)
基类: *openpyxl.descriptors.serialisable.Serialisable*

algorithmName
Values must be of type <class 'str'>

hashValue

readOnlyRecommended
Values must be of type <class 'bool'>

reservationPassword

saltValue

spinCount
Values must be of type <class 'int'>

tagname = 'fileSharing'

userName

Values must be of type <class 'str' >

class openpyxl.workbook.protection.WorkbookProtection(*workbookPassword=None*, *workbookPasswordCharacterSet=None*, *revisionsPassword=None*, *revisionsPasswordCharacterSet=None*, *lockStructure=None*, *lockWindows=None*, *lockRevision=None*, *revisionsAlgorithmName=None*, *revisionsHashValue=None*, *revisionsSaltValue=None*, *revisionsSpinCount=None*, *workbookAlgorithmName=None*, *workbookHashValue=None*, *workbookSaltValue=None*, *workbookSpinCount=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

classmethod from_tree(*node*)

Don't hash passwords when deserialising from XML

lockRevision

Values must be of type <class 'bool' >

lockStructure

Values must be of type <class 'bool' >

lockWindows

Values must be of type <class 'bool' >

lock_revision

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

lock_structure

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

lock_windows

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

revision_password

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

revisionsAlgorithmName

Values must be of type <class 'str' >

revisionsHashValue

revisionsPassword

Return the revisions password value, regardless of hash.

revisionsPasswordCharacterSet

Values must be of type <class 'str' >

revisionsSaltValue

revisionsSpinCount

Values must be of type <class 'int' >

set_revisions_password(*value=", already_hashed=False*)

Set a revision password on this workbook.

set_workbook_password(*value=", already_hashed=False*)

Set a password on this workbook.

tagname = 'workbookPr'

workbookAlgorithmName

Values must be of type <class 'str' >

workbookHashValue

workbookPassword

Return the workbook password value, regardless of hash.

workbookPasswordCharacterSet

Values must be of type <class 'str' >

workbookSaltValue

workbookSpinCount

Values must be of type <class 'int' >

workbook_password

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

openpyxl.workbook.smart_tags module

class openpyxl.workbook.smart_tags.**SmartTag**(*namespaceUri=None, name=None, url=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

name

Values must be of type <class 'str' >

namespaceUri
Values must be of type <class 'str' >

tagname = 'smartTagType'

url
Values must be of type <class 'str' >

class openpyxl.workbook.smart_tags.**SmartTagList**(*smartTagType=()*)
基类: *openpyxl.descriptors.serialisable.Serialisable*

smartTagType
A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'smartTagTypes'

class openpyxl.workbook.smart_tags.**SmartTagProperties**(*embed=None*, *show=None*)
基类: *openpyxl.descriptors.serialisable.Serialisable*

embed
Values must be of type <class 'bool' >

show
Value must be one of { 'all' , 'noIndicator' }

tagname = 'smartTagPr'

openpyxl.workbook.views module

class openpyxl.workbook.views.**BookView**(*visibility='visible'*, *minimized=False*, *showHorizontalScroll=True*, *showVerticalScroll=True*, *showSheetTabs=True*, *xWindow=None*, *yWindow=None*, *windowWidth=None*, *windowHeight=None*, *tabRatio=600*, *firstSheet=0*, *activeTab=0*, *autoFilterDateGrouping=True*, *extLst=None*)
基类: *openpyxl.descriptors.serialisable.Serialisable*

activeTab
Values must be of type <class 'int' >

autoFilterDateGrouping
Values must be of type <class 'bool' >

extLst
Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

firstSheet
Values must be of type <class 'int' >

minimized

Values must be of type <class 'bool'>

showHorizontalScroll

Values must be of type <class 'bool'>

showSheetTabs

Values must be of type <class 'bool'>

showVerticalScroll

Values must be of type <class 'bool'>

tabRatio

Values must be of type <class 'int'>

tagname = 'workbookView'

visibility

Value must be one of {'veryHidden', 'visible', 'hidden'}

windowHeight

Values must be of type <class 'int'>

windowWidth

Values must be of type <class 'int'>

xWindow

Values must be of type <class 'int'>

yWindow

Values must be of type <class 'int'>

class openpyxl.workbook.views.**CustomWorkbookView**(*name=None, guid=None, autoUpdate=None, mergeInterval=None, changesSavedWin=None, onlySync=None, personalView=None, includePrintSettings=None, includeHiddenRowCol=None, maximized=None, minimized=None, showHorizontalScroll=None, showVerticalScroll=None, showSheetTabs=None, xWindow=None, yWindow=None, windowWidth=None, windowHeight=None, tabRatio=None, activeSheetId=None, showFormulaBar=None, showStatusbar=None, showComments='commIndicator', showObjects='all', extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

activeSheetId

Values must be of type <class 'int' >

autoUpdate

Values must be of type <class 'bool' >

changesSavedWin

Values must be of type <class 'bool' >

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

guid

includeHiddenRowCol

Values must be of type <class 'bool' >

includePrintSettings

Values must be of type <class 'bool' >

maximized

Values must be of type <class 'bool' >

mergeInterval

Values must be of type <class 'int' >

minimized

Values must be of type <class 'bool' >

name

Values must be of type <class 'str' >

onlySync

Values must be of type <class 'bool' >

personalView

Values must be of type <class 'bool' >

showComments

Value must be one of { 'commNone' , 'commIndAndComment' , 'commIndicator' }

showFormulaBar

Values must be of type <class 'bool' >

showHorizontalScroll

Values must be of type <class 'bool' >

showObjects

Value must be one of { 'all' , 'placeholders' }

showSheetTabs

Values must be of type <class 'bool' >

showStatusbar

Values must be of type <class 'bool'>

showVerticalScroll

Values must be of type <class 'bool'>

tabRatio

Values must be of type <class 'int'>

tagname = 'customWorkbookView'

windowHeight

Values must be of type <class 'int'>

windowWidth

Values must be of type <class 'int'>

xWindow

Values must be of type <class 'int'>

yWindow

Values must be of type <class 'int'>

openpyxl.workbook.web module

class openpyxl.workbook.web.**WebPublishObject**(*id=None*, *divId=None*, *sourceObject=None*, *destinationFile=None*, *title=None*, *autoRepublish=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

autoRepublish

Values must be of type <class 'bool'>

destinationFile

Values must be of type <class 'str'>

divId

Values must be of type <class 'str'>

id

Values must be of type <class 'int'>

sourceObject

Values must be of type <class 'str'>

tagname = 'webPublishingObject'

title

Values must be of type <class 'str'>

class openpyxl.workbook.web.**WebPublishObjectList**(*count=None*, *webPublishObject=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

count

tagname = 'webPublishingObjects'

webPublishObject

A sequence (list or tuple) that may only contain objects of the declared type

class openpyxl.workbook.web.**WebPublishing**(*css=None*, *thicket=None*, *longFileNames=None*, *vml=None*, *allowPng=None*, *targetScreenSize='800x600'*, *dpi=None*, *codePage=None*, *characterSet=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

allowPng

Values must be of type <class ‘bool’ >

characterSet

Values must be of type <class ‘str’ >

codePage

Values must be of type <class ‘int’ >

css

Values must be of type <class ‘bool’ >

dpi

Values must be of type <class ‘int’ >

longFileNames

Values must be of type <class ‘bool’ >

tagname = 'webPublishing'

targetScreenSize

Value must be one of { ‘1920x1200’ , ‘544x376’ , ‘1152x900’ , ‘1024x768’ , ‘640x480’ , ‘720x512’ , ‘1600x1200’ , ‘1800x1440’ , ‘800x600’ , ‘1152x882’ , ‘1280x1024’ }

thicket

Values must be of type <class ‘bool’ >

vml

Values must be of type <class ‘bool’ >

openpyxl.workbook.workbook module

Workbook is the top-level container for all document information.

class openpyxl.workbook.workbook.**Workbook**(*write_only=False*, *iso_dates=False*)

基类：object

Workbook is the container for all other parts of the document.

active

Get the currently active sheet or None

Type *openpyxl.worksheet.worksheet.Worksheet*

add_named_range(*named_range*)

Add an existing named_range to the list of named_ranges.

注解： Deprecated: Use workbook.defined_names.append

add_named_style(*style*)

Add a named style

chartsheets

A list of Chartsheets in this workbook

Type list of *openpyxl.chartsheet.chartsheet.Chartsheet*

close()

Close workbook file if open. Only affects read-only and write-only modes.

copy_worksheet(*from_worksheet*)

Copy an existing worksheet in the current workbook

警告： This function cannot copy worksheets between workbooks. worksheets can only be copied within the workbook that they belong

参数 **from_worksheet** – the worksheet to be copied from

返回 copy of the initial worksheet

create_chartsheet(*title=None*, *index=None*)

create_named_range(*name*, *worksheet=None*, *value=None*, *scope=None*)

Create a new named_range on a worksheet

create_sheet(*title=None*, *index=None*)

Create a worksheet (at an optional index).

参数

- **title** (*str*) – optional title of the sheet
- **index** (*int*) – optional position at which the sheet will be inserted

data_only

epoch

excel_base_date

get_index(*worksheet*)

Return the index of the worksheet.

注解： Deprecated: Use wb.index(worksheet)

get_named_range(*name*)

Return the range specified by name.

注解： Deprecated: Use workbook.defined_names[name]

get_named_ranges()

Return all named ranges

注解： Deprecated: Use workbook.defined_names.definedName

get_sheet_by_name(*name*)

Returns a worksheet by its name.

param name the name of the worksheet to look for

type name string

注解： Deprecated: Use wb[sheetname]

get_sheet_names()

注解： Deprecated: Use wb.sheetnames

index(*worksheet*)

Return the index of a worksheet.

mime_type

The mime type is determined by whether a workbook is a template or not and whether it contains macros or not. Excel requires the file extension to match but openpyxl does not enforce this.

`move_sheet`(*sheet*, *offset=0*)

Move a sheet or sheetname

`named_styles`

List available named styles

`path = '/xl/workbook.xml'`

`read_only`

`remove`(*worksheet*)

Remove *worksheet* from this workbook.

`remove_named_range`(*named_range*)

Remove a named_range from this workbook.

注解： Deprecated: Use del workbook.defined_names[name]

`remove_sheet`(*worksheet*)

Remove *worksheet* from this workbook.

注解： Deprecated: Use wb.remove(worksheet) or del wb[sheetname]

`save`(*filename*)

Save the current workbook under the given *filename*. Use this function instead of using an *ExcelWriter*.

警告： When creating your workbook using *write_only* set to True, you will only be able to call this function once. Subsequents attempts to modify or save the file will raise an `openpyxl.shared.exc.WorkbookAlreadySaved` exception.

`sheetnames`

Returns the list of the names of worksheets in this workbook.

Names are returned in the worksheets order.

Type list of strings

`style_names`

List of named styles

`template = False`

`worksheets`

A list of sheets in this workbook

Type list of *openpyxl.worksheet.worksheet.Worksheet*

`write_only`

openpyxl.worksheet package

Submodules

openpyxl.worksheet.cell_range module

`class openpyxl.worksheet.cell_range.CellRange`(*range_string=None*, *min_col=None*, *min_row=None*, *max_col=None*, *max_row=None*, *title=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Represents a range in a sheet: title and coordinates.

This object is used to perform operations on ranges, like:

- shift, expand or shrink
- union/intersection with another sheet range,

We can check whether a range is:

- equal or not equal to another,
- disjoint of another,
- contained in another.

We can get:

- the size of a range.
- the range bounds (vertices)
- the coordinates,
- the string representation,

`bottom`

A list of cell coordinates that comprise the bottom of the range

`bounds`

Vertices of the range as a tuple

`cells`

`cols`

Return cell coordinates as columns

`coord`
Excel-style representation of the range

`expand`(*right=0*, *down=0*, *left=0*, *up=0*)
Expand the range by the dimensions provided.

参数

- **right** (*int*) – expand range to the right by this number of cells
- **down** (*int*) – expand range down by this number of cells
- **left** (*int*) – expand range to the left by this number of cells
- **up** (*int*) – expand range up by this number of cells

`intersection`(*other*)
Return a new range with cells common to this range and *other*

参数 **other** (`openpyxl.worksheet.cell_range.CellRange`) – Other sheet range.

返回 the intersecting sheet range.

Raise `ValueError` if the *other* range doesn’ t intersect with this range.

`isdisjoint`(*other*)
Return `True` if this range has no cell in common with *other*. Ranges are disjoint if and only if their intersection is the empty range.

参数 **other** (`openpyxl.worksheet.cell_range.CellRange`) – Other sheet range.

返回 `True` if the range has no cells in common with other.

`issubset`(*other*)
Test whether every cell in this range is also in *other*.

参数 **other** (`openpyxl.worksheet.cell_range.CellRange`) – Other sheet range

返回 `True` if *range <= other*.

`issuperset`(*other*)
Test whether every cell in *other* is in this range.

参数 **other** (`openpyxl.worksheet.cell_range.CellRange`) – Other sheet range

返回 `True` if *range >= other* (or *other* in *range*).

`left`
A list of cell coordinates that comprise the left-side of the range

`max_col`
Values must be of type <class ‘int’ >

`max_row`
Values must be of type <class ‘int’ >

min_col

Values must be of type <class 'int'>

min_row

Values must be of type <class 'int'>

right

A list of cell coordinates that comprise the right-side of the range

rows

Return cell coordinates as rows

shift(*col_shift=0*, *row_shift=0*)

Shift the focus of the range according to the shift values (*col_shift*, *row_shift*).

参数

- **col_shift** (*int*) – number of columns to be moved by, can be negative
- **row_shift** (*int*) – number of rows to be moved by, can be negative

Raise `ValueError` if any row or column index < 1

shrink(*right=0*, *bottom=0*, *left=0*, *top=0*)

Shrink the range by the dimensions provided.

参数

- **right** (*int*) – shrink range from the right by this number of cells
- **down** (*int*) – shrink range from the top by this number of cells
- **left** (*int*) – shrink range from the left by this number of cells
- **up** (*int*) – shrink range from the bottown by this number of cells

size

Return the size of the range as a dictionary of rows and columns.

top

A list of cell coordinates that comprise the top of the range

union(*other*)

Return the minimal superset of this range and *other*. This new range will contain all cells from this range, *other*, and any additional cells required to form a rectangular `CellRange`.

参数 **other** (`openpyxl.worksheet.cell_range.CellRange`) – Other sheet range.

返回 a `CellRange` that is a superset of this and *other*.

class openpyxl.worksheet.cell_range.**MultiCellRange**(*ranges=()*)

基类：*openpyxl.descriptors.Strict*

add(*coord*)

Add a cell coordinate or CellRange

ranges

A sequence (list or tuple) that may only contain objects of the declared type

remove(*coord*)

openpyxl.worksheet.cell_watch module

class openpyxl.worksheet.cell_watch.**CellWatch**(*r=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

r

Values must be of type <class ‘str’ >

tagname = 'cellWatch'

class openpyxl.worksheet.cell_watch.**CellWatches**(*cellWatch=()*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

cellWatch

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'cellWatches'

openpyxl.worksheet.controls module

class openpyxl.worksheet.controls.**Control**(*controlPr=None, shapeId=None, name=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

controlPr

Values must be of type <class ‘openpyxl.worksheet.controls.ControlProperty’ >

name

Values must be of type <class ‘str’ >

shapeId

Values must be of type <class ‘int’ >

tagname = 'control'

class openpyxl.worksheet.controls.**ControlProperty**(*anchor=None, locked=True, defaultSize=True, _print=True, disabled=False, recalcAlways=False, uiObject=False, autoFill=True, autoLine=True, autoPict=True, macro=None, altText=None, linkedCell=None, listFillRange=None, cf='pict', id=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`altText`
Values must be of type <class ‘str’ >

`anchor`
Values must be of type <class ‘openpyxl.worksheet.ole.ObjectAnchor’ >

`autoFill`
Values must be of type <class ‘bool’ >

`autoLine`
Values must be of type <class ‘bool’ >

`autoPict`
Values must be of type <class ‘bool’ >

`cf`
Values must be of type <class ‘str’ >

`defaultSize`
Values must be of type <class ‘bool’ >

`disabled`
Values must be of type <class ‘bool’ >

`id`
Values must be of type <class ‘str’ >

`linkedCell`
Values must be of type <class ‘str’ >

`listFillRange`
Values must be of type <class ‘str’ >

`locked`
Values must be of type <class ‘bool’ >

`macro`
Values must be of type <class ‘str’ >

`recalcAlways`
Values must be of type <class ‘bool’ >

`tagname = 'controlPr'`

`uiObject`
Values must be of type <class ‘bool’ >

`class openpyxl.worksheet.controls.Controls(`*control=()*`)`
基类：*openpyxl.descriptors.serialisable.Serialisable*

`control`

A sequence (list or tuple) that may only contain objects of the declared type

`tagname = 'controls'`

openpyxl.worksheet.copier module

`class openpyxl.worksheet.copier.WorksheetCopy(`*source_worksheet, target_worksheet*`)`

基类：`object`

Copy the values, styles, dimensions, merged cells, margins, and print/page setup from one worksheet to another within the same workbook.

`copy_worksheet()`

openpyxl.worksheet.custom module

`class openpyxl.worksheet.custom.CustomProperties(`*customPr=()*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`customPr`

A sequence (list or tuple) that may only contain objects of the declared type

`tagname = 'customProperties'`

`class openpyxl.worksheet.custom.CustomProperty(`*name=None*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`name`

Values must be of type <class ‘str’ >

`tagname = 'customProperty'`

openpyxl.worksheet.datavalidation module

`class openpyxl.worksheet.datavalidation.DataValidation(`*type=None, formula1=None, formula2=None, allow_blank=False, showErrorMessage=True, showInputMessage=True, showDropDown=None, allowBlank=None, sqref=(), promptTitle=None, errorStyle=None, error=None, prompt=None, errorTitle=None, imeMode=None, operator=None*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`add(`*cell*`)`
Adds a cell or cell coordinate to this validator

`allowBlank`
Values must be of type <class 'bool'>

`allow_blank`
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`cells`
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`error`
Values must be of type <class 'str'>

`errorStyle`
Value must be one of {'information', 'stop', 'warning'}

`errorTitle`
Values must be of type <class 'str'>

`formula1`
Values must be of type <class 'str'>

`formula2`
Values must be of type <class 'str'>

`hide_drop_down`
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`imeMode`
Value must be one of {'fullHangul', 'on', 'noControl', 'disabled', 'halfAlpha', 'halfHangul', 'halfKatakana', 'fullKatakana', 'off', 'hiragana', 'fullAlpha'}

`operator`
Value must be one of {'lessThanOrEqual', 'notBetween', 'lessThan', 'greaterThanOrEqual', 'greaterThan', 'equal', 'between', 'notEqual'}

`prompt`
Values must be of type <class 'str'>

`promptTitle`
Values must be of type <class 'str'>

`ranges`
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

showDropDown

Values must be of type <class 'bool'>

showErrorMessage

Values must be of type <class 'bool'>

showInputMessage

Values must be of type <class 'bool'>

sqref

Values must be of type <class 'openpyxl.worksheet.cell_range.MultiCellRange'>

tagname = 'dataValidation'

type

Value must be one of {'custom', 'date', 'list', 'time', 'decimal', 'textLength', 'whole'}

validation_type

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

class openpyxl.worksheet.datavalidation.**DataValidationList**(*disablePrompts=None, xWindow=None, yWindow=None, count=None, dataValidation=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

append(*dv*)

count

dataValidation

A sequence (list or tuple) that may only contain objects of the declared type

disablePrompts

Values must be of type <class 'bool'>

tagname = 'dataValidations'

to_tree(*tagname=None*)

Need to skip validations that have no cell ranges

xWindow

Values must be of type <class 'int'>

yWindow

Values must be of type <class 'int'>

openpyxl.worksheet.datavalidation.**collapse_cell_addresses**(*cells, input_ranges=()*)

Collapse a collection of cell co-ordinates down into an optimal range or collection of ranges.

E.g. Cells A1, A2, A3, B1, B2 and B3 should have the data-validation object applied, attempt to collapse down to a single range, A1:B3.

Currently only collapsing contiguous vertical ranges (i.e. above example results in A1:A3 B1:B3).

`openpyxl.worksheet.datavalidation.expand_cell_ranges(`*range_string*`)`

Expand cell ranges to a sequence of addresses. Reverse of collapse_cell_addresses Eg. converts "A1:A2 B1:B2" to (A1, A2, B1, B2)

openpyxl.worksheet.dimensions module

`class openpyxl.worksheet.dimensions.ColumnDimension(`*worksheet, index='A', width=13, bestFit=False, hidden=False, outlineLevel=0, outline_level=None, collapsed=False, style=None, min=None, max=None, customWidth=False, visible=None, auto_size=None*`)`

基类：*openpyxl.worksheet.dimensions.Dimension*

Information about the display properties of a column.

auto_size

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

bestFit

Values must be of type <class 'bool' >

collapsed

Values must be of type <class 'bool' >

customWidth

Always true if there is a width for the column

index

Values must be of type <class 'str' >

max

Values must be of type <class 'int' >

min

Values must be of type <class 'int' >

reindex()

Set boundaries for column definition

to_tree()

width

Values must be of type <class 'float' >

class openpyxl.worksheet.dimensions.Dimension(*index*, *hidden*, *outlineLevel*, *collapsed*, *worksheet*, *visible=True*, *style=None*)

基类：*openpyxl.descriptors.Strict*, *openpyxl.styles.styleable.StyleableObject*

Information about the display properties of a row or column.

collapsed

Values must be of type <class 'bool' >

hidden

Values must be of type <class 'bool' >

index

Values must be of type <class 'int' >

outlineLevel

Values must be of type <class 'int' >

outline_level

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

style

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

class openpyxl.worksheet.dimensions.DimensionHolder(*worksheet*, *reference='index'*, *default_factory=None*)

基类：*openpyxl.utils.bound_dictionary.BoundDictionary*

Allow columns to be grouped

group(*start*, *end=None*, *outline_level=1*, *hidden=False*)

allow grouping a range of consecutive rows or columns together

参数

- **start** – first row or column to be grouped (mandatory)
- **end** – last row or column to be grouped (optional, default to start)
- **outline_level** – outline level
- **hidden** – should the group be hidden on workbook open or not

to_tree()

`class openpyxl.worksheet.dimensions.RowDimension`(*worksheet, index=0, ht=None, customHeight=None, s=None, customFormat=None, hidden=False, outlineLevel=0, outline_level=None, collapsed=False, visible=None, height=None, r=None, spans=None, thickBot=None, thickTop=None, **kw*)

基类：*openpyxl.worksheet.dimensions.Dimension*

Information about the display properties of a row.

customFormat

Always true if there is a style for the row

customHeight

Always true if there is a height for the row

height

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

ht

Values must be of type <class 'float' >

r

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

s

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

thickBot

Values must be of type <class 'bool' >

thickTop

Values must be of type <class 'bool' >

`class openpyxl.worksheet.dimensions.SheetDimension`(*ref=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

boundaries

ref

Values must be of type <class 'str' >

`tagname = 'dimension'`

class openpyxl.worksheet.dimensions.**SheetFormatProperties**(*baseColWidth=8, defaultColWidth=None, defaultRowHeight=15, customHeight=None, zeroHeight=None, thickTop=None, thickBottom=None, outlineLevelRow=None, outlineLevelCol=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

baseColWidth

Values must be of type <class 'int'>

customHeight

Values must be of type <class 'bool'>

defaultColWidth

Values must be of type <class 'float'>

defaultRowHeight

Values must be of type <class 'float'>

outlineLevelCol

Values must be of type <class 'int'>

outlineLevelRow

Values must be of type <class 'int'>

tagname = 'sheetFormatPr'

thickBottom

Values must be of type <class 'bool'>

thickTop

Values must be of type <class 'bool'>

zeroHeight

Values must be of type <class 'bool'>

openpyxl.worksheet.drawing module

class openpyxl.worksheet.drawing.**Drawing**(*id=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

id

Values must be of type <class 'str'>

tagname = 'drawing'

openpyxl.worksheet.errors module

class openpyxl.worksheet.errors.**Extension**(*uri=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

tagname = 'extension'

uri

Values must be of type <class 'str' >

class openpyxl.worksheet.errors.**ExtensionList**(*ext=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

ext

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'extensionList'

class openpyxl.worksheet.errors.**IgnoredError**(*sqref=None, evalError=False, twoDigitTextYear=False, numberStoredAsText=False, formula=False, formulaRange=False, unlockedFormula=False, emptyCellReference=False, listDataValidation=False, calculatedColumn=False*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

calculatedColumn

Values must be of type <class 'bool' >

emptyCellReference

Values must be of type <class 'bool' >

evalError

Values must be of type <class 'bool' >

formula

Values must be of type <class 'bool' >

formulaRange

Values must be of type <class 'bool' >

listDataValidation

Values must be of type <class 'bool' >

numberStoredAsText

Values must be of type <class 'bool' >

sqref

openpyxl.descriptors.excel.CellRange 的别名

tagname = 'ignoredError'

twoDigitTextYear

Values must be of type <class 'bool'>

unlockedFormula

Values must be of type <class 'bool'>

class openpyxl.worksheet.errors.**IgnoredErrors**(*ignoredError=()*, *extLst=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

extLst

Values must be of type <class 'openpyxl.worksheet.errors.ExtensionList'>

ignoredError

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'ignoredErrors'

openpyxl.worksheet.filters module

class openpyxl.worksheet.filters.**AutoFilter**(*ref=None*, *filterColumn=()*, *sortState=None*, *extLst=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

add_filter_column(*col_id*, *vals*, *blank=False*)

Add row filter for specified column.

参数

- **col_id** (*int*) – Zero-origin column id. 0 means first column.
- **vals** (*str[]*) – Value list to show.
- **blank** (*bool*) – Show rows that have blank cell if True (default=``False``)

add_sort_condition(*ref*, *descending=False*)

Add sort condition for cpecified range of cells.

参数

- **ref** (*string*) – range of the cells (e.g. 'A2:A150')
- **descending** (*bool*) – Descending sort order (default=``False``)

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

filterColumn

A sequence (list or tuple) that may only contain objects of the declared type

ref

sortState

Values must be of type <class 'openpyxl.worksheet.filters.SortState'>

tagname = 'autoFilter'

class openpyxl.worksheet.filters.**ColorFilter**(*dxfId=None*, *cellColor=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

cellColor

Values must be of type <class ‘bool’ >

dxfId

Values must be of type <class ‘int’ >

tagname = 'colorFilter'

class openpyxl.worksheet.filters.**CustomFilter**(*operator=None*, *val=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

operator

Value must be one of { ‘lessThanOrEqual’ , ‘greaterThanOrEqual’ , ‘lessThan’ , ‘greaterThan’ , ‘equal’ , ‘notEqual’ }

tagname = 'customFilter'

val

Values must be of type <class ‘str’ >

class openpyxl.worksheet.filters.**CustomFilters**(*_and=None*, *customFilter=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

customFilter

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'customFilters'

class openpyxl.worksheet.filters.**DateGroupItem**(*year=None*, *month=None*, *day=None*, *hour=None*, *minute=None*, *second=None*, *dateTimeGrouping=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

dateTimeGrouping

Value must be one of { ‘minute’ , ‘second’ , ‘year’ , ‘hour’ , ‘month’ , ‘day’ }

day

Values must be of type <class ‘float’ >

hour

Values must be of type <class ‘float’ >

minute

Values must be of type <class ‘float’ >

month

Values must be of type <class ‘float’ >

`second`

Values must be of type <class 'int'>

`tagname = 'dateGroupItem'`

`year`

Values must be of type <class 'int'>

`class openpyxl.worksheet.filters.DynamicFilter`(*type=None*, *val=None*, *valIso=None*, *maxVal=None*, *maxValIso=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`maxVal`

Values must be of type <class 'float'>

`maxValIso`

Values must be of type <class 'datetime.datetime'>

`tagname = 'dynamicFilter'`

`type`

Value must be one of {'thisYear', 'aboveAverage', 'belowAverage', 'yesterday', 'Q2', 'thisMonth', 'null', 'thisQuarter', 'nextWeek', 'M10', 'M2', 'tomorrow', 'nextQuarter', 'M3', 'M8', 'lastMonth', 'M9', 'lastYear', 'Q4', 'thisWeek', 'M11', 'M5', 'Q1', 'M6', 'M4', 'M1', 'M7', 'yearToDate', 'today', 'M12', 'lastWeek', 'nextMonth', 'Q3', 'lastQuarter', 'nextYear'}

`val`

Values must be of type <class 'float'>

`valIso`

Values must be of type <class 'datetime.datetime'>

`class openpyxl.worksheet.filters.FilterColumn`(*colId=None*, *hiddenButton=None*, *showButton=None*, *filters=None*, *top10=None*, *customFilters=None*, *dynamicFilter=None*, *colorFilter=None*, *iconFilter=None*, *extLst=None*, *blank=None*, *vals=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`colId`

Values must be of type <class 'int'>

`col_id`

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`colorFilter`

Values must be of type <class 'openpyxl.worksheet.filters.ColorFilter'>

customFilters

Values must be of type <class ‘openpyxl.worksheet.filters.CustomFilters’ >

dynamicFilter

Values must be of type <class ‘openpyxl.worksheet.filters.DynamicFilter’ >

extLst

Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

filters

Values must be of type <class ‘openpyxl.worksheet.filters.Filters’ >

hiddenButton

Values must be of type <class ‘bool’ >

iconFilter

Values must be of type <class ‘openpyxl.worksheet.filters.IconFilter’ >

showButton

Values must be of type <class ‘bool’ >

tagname = 'filterColumn'

top10

Values must be of type <class ‘openpyxl.worksheet.filters.Top10’ >

class openpyxl.worksheet.filters.**Filters**(*blank=None*, *calendarType=None*, *filter=()*, *dateGroupItem=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

blank

Values must be of type <class ‘bool’ >

calendarType

Value must be one of { ‘gregorianXlitFrench’ , ‘gregorianArabic’ , ‘gregorianXlitEnglish’ , ‘saka’ , ‘korea’ , ‘gregorianMeFrench’ , ‘gregorian’ , ‘thai’ , ‘gregorianUs’ , ‘taiwan’ , ‘japan’ , ‘hijri’ , ‘hebrew’ }

dateGroupItem

A sequence (list or tuple) that may only contain objects of the declared type

filter

A sequence of primitive types that are stored as a single attribute. “val” is the default attribute

tagname = 'filters'

class openpyxl.worksheet.filters.**IconFilter**(*iconSet=None*, *iconId=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

iconId

Values must be of type <class ‘int’ >

iconSet

Value must be one of { '3Arrows', '3ArrowsGray', '3Symbols', '4Rating', '5Quarters', '4RedToBlack', '5ArrowsGray', '3Signs', '4ArrowsGray', '5Rating', '3Symbols2', '3TrafficLights1', '4TrafficLights', '5Arrows', '4Arrows', '3Flags', '3TrafficLights2' }

tagname = 'iconFilter'

class openpyxl.worksheet.filters.**SortCondition**(*ref=None*, *descending=None*, *sortBy=None*, *customList=None*, *dxfId=None*, *iconSet=None*, *iconId=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

customList

Values must be of type <class 'str'>

descending

Values must be of type <class 'bool'>

dxfId

Values must be of type <class 'int'>

iconId

Values must be of type <class 'int'>

iconSet

Value must be one of { '3Arrows', '3ArrowsGray', '3Symbols', '4Rating', '5Quarters', '4RedToBlack', '5ArrowsGray', '3Signs', '4ArrowsGray', '5Rating', '3Symbols2', '3TrafficLights1', '4TrafficLights', '5Arrows', '4Arrows', '3Flags', '3TrafficLights2' }

ref

sortBy

Value must be one of { 'icon', 'fontColor', 'value', 'cellColor' }

tagname = 'sortCondition'

class openpyxl.worksheet.filters.**SortState**(*columnSort=None*, *caseSensitive=None*, *sortMethod=None*, *ref=None*, *sortCondition=()*, *extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

caseSensitive

Values must be of type <class 'bool'>

columnSort

Values must be of type <class 'bool'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

ref

sortCondition

A sequence (list or tuple) that may only contain objects of the declared type

sortMethod

Value must be one of { ‘pinYin’ , ‘stroke’ }

tagname = 'sortState'

class openpyxl.worksheet.filters.**Top10**(*top=None*, *percent=None*, *val=None*, *filterVal=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

filterVal

Values must be of type <class ‘float’ >

percent

Values must be of type <class ‘bool’ >

tagname = 'top10'

top

Values must be of type <class ‘bool’ >

val

Values must be of type <class ‘float’ >

openpyxl.worksheet.header_footer module

class openpyxl.worksheet.header_footer.**HeaderFooter**(*differentOddEven=None*, *differentFirst=None*, *scaleWithDoc=None*, *alignWithMargins=None*, *oddHeader=None*, *oddFooter=None*, *evenHeader=None*, *evenFooter=None*, *firstHeader=None*, *firstFooter=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

alignWithMargins

Values must be of type <class ‘bool’ >

differentFirst

Values must be of type <class ‘bool’ >

differentOddEven

Values must be of type <class ‘bool’ >

evenFooter

Values must be of type <class ‘openpyxl.worksheet.header_footer.HeaderFooterItem’ >

evenHeader

Values must be of type <class ‘openpyxl.worksheet.header_footer.HeaderFooterItem’ >

firstFooter
Values must be of type <class 'openpyxl.worksheet.header_footer.HeaderFooterItem'>

firstHeader
Values must be of type <class 'openpyxl.worksheet.header_footer.HeaderFooterItem'>

oddFooter
Values must be of type <class 'openpyxl.worksheet.header_footer.HeaderFooterItem'>

oddHeader
Values must be of type <class 'openpyxl.worksheet.header_footer.HeaderFooterItem'>

scaleWithDoc
Values must be of type <class 'bool'>

tagname = 'headerFooter'

class openpyxl.worksheet.header_footer.**HeaderFooterItem**(*left=None*, *right=None*, *center=None*)

基类：*openpyxl.descriptors.Strict*

Header or footer item

center
Values must be of type <class 'openpyxl.worksheet.header_footer._HeaderFooterPart'>

centre
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

classmethod **from_tree**(*node*)

left
Values must be of type <class 'openpyxl.worksheet.header_footer._HeaderFooterPart'>

right
Values must be of type <class 'openpyxl.worksheet.header_footer._HeaderFooterPart'>

to_tree(*tagname*)
Return as XML node

openpyxl.worksheet.hyperlink module

class openpyxl.worksheet.hyperlink.**Hyperlink**(*ref=None*, *location=None*, *tooltip=None*, *display=None*, *id=None*, *target=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

display
Values must be of type <class 'str'>

id

Values must be of type <class 'str'>

location

Values must be of type <class 'str'>

ref

Values must be of type <class 'str'>

tagname = 'hyperlink'

target

Values must be of type <class 'str'>

tooltip

Values must be of type <class 'str'>

class openpyxl.worksheet.hyperlink.**HyperlinkList**(*hyperlink=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

append(*value*)

hyperlink

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'hyperlinks'

openpyxl.worksheet.merge module

class openpyxl.worksheet.merge.**MergeCell**(*ref=None*)

基类：*openpyxl.worksheet.cell_range.CellRange*

ref

Excel-style representation of the range

tagname = 'mergeCell'

class openpyxl.worksheet.merge.**MergeCells**(*count=None, mergeCell=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

count

mergeCell

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'mergeCells'

class openpyxl.worksheet.merge.**MergedCellRange**(*worksheet, coord*)

基类：*openpyxl.worksheet.cell_range.CellRange*

MergedCellRange stores the border information of a merged cell in the top left cell of the merged cell. The remaining cells in the merged cell are stored as MergedCell objects and get their border information from the upper left cell.

format()

Each cell of the merged cell is created as MergedCell if it does not already exist.

The MergedCells at the edge of the merged cell gets its borders from the upper left cell.

- The top MergedCells get the top border from the top left cell.
- The bottom MergedCells get the bottom border from the top left cell.
- The left MergedCells get the left border from the top left cell.
- The right MergedCells get the right border from the top left cell.

openpyxl.worksheet.ole module

class openpyxl.worksheet.ole.**ObjectAnchor**(*_from=None, to=None, moveWithCells=False, sizeWithCells=False, z_order=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

moveWithCells

Values must be of type <class 'bool'>

sizeWithCells

Values must be of type <class 'bool'>

tagname = 'anchor'

to

Values must be of type <class 'openpyxl.drawing.spreadsheet_drawing.AnchorMarker'>

z_order

Values must be of type <class 'int'>

class openpyxl.worksheet.ole.**ObjectPr**(*anchor=None, locked=True, defaultSize=True, _print=True, disabled=False, uiObject=False, autoFill=True, autoLine=True, autoPict=True, macro=None, altText=None, dde=False*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

altText

Values must be of type <class 'str'>

anchor

Values must be of type <class 'openpyxl.worksheet.ole.ObjectAnchor'>

autoFill

Values must be of type <class 'bool'>

autoLine

Values must be of type <class 'bool'>

autoPict

Values must be of type <class 'bool'>

dde

Values must be of type <class 'bool'>

defaultSize

Values must be of type <class 'bool'>

disabled

Values must be of type <class 'bool'>

locked

Values must be of type <class 'bool'>

macro

Values must be of type <class 'str'>

tagname = 'objectPr'

uiObject

Values must be of type <class 'bool'>

class openpyxl.worksheet.ole.**OleObject**(*objectPr=None, progId=None, dvAspect='DVASPECT_CONTENT', link=None, oleUpdate=None, autoLoad=False, shapeId=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

autoLoad

Values must be of type <class 'bool'>

dvAspect

Value must be one of {'DVASPECT_CONTENT', 'DVASPECT_ICON'}

link

Values must be of type <class 'str'>

objectPr

Values must be of type <class 'openpyxl.worksheet.ole.ObjectPr'>

oleUpdate

Value must be one of {'OLEUPDATE_ALWAYS', 'OLEUPDATE_ONCALL'}

progId

Values must be of type <class 'str'>

shapeId

Values must be of type <class 'int'>

tagname = 'oleObject'

class openpyxl.worksheet.ole.**OleObjects**(*oleObject=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

oleObject

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'oleObjects'

openpyxl.worksheet.page module

class openpyxl.worksheet.page.**PageMargins**(*left=0.75*, *right=0.75*, *top=1*, *bottom=1*, *header=0.5*, *footer=0.5*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Information about page margins for view/print layouts. Standard values (in inches) left, right = 0.75 top, bottom = 1 header, footer = 0.5

bottom

Values must be of type <class 'float' >

footer

Values must be of type <class 'float' >

header

Values must be of type <class 'float' >

left

Values must be of type <class 'float' >

right

Values must be of type <class 'float' >

tagname = 'pageMargins'

top

Values must be of type <class 'float' >

class openpyxl.worksheet.page.**PrintOptions**(*horizontalCentered=None*, *verticalCentered=None*, *headings=None*, *gridLines=None*, *gridLinesSet=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Worksheet print options

gridLines

Values must be of type <class 'bool' >

gridLinesSet

Values must be of type <class 'bool' >

headings

Values must be of type <class 'bool' >

horizontalCentered

Values must be of type <class 'bool' >

tagname = 'printOptions'

verticalCentered

Values must be of type <class 'bool' >

class openpyxl.worksheet.page.**PrintPageSetup**(*worksheet=None, orientation=None, paperSize=None, scale=None, fitToHeight=None, fitToWidth=None, firstPageNumber=None, useFirstPageNumber=None, paperHeight=None, paperWidth=None, pageOrder=None, usePrinterDefaults=None, blackAndWhite=None, draft=None, cellComments=None, errors=None, horizontalDpi=None, verticalDpi=None, copies=None, id=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Worksheet print page setup

autoPageBreaks

blackAndWhite

Values must be of type <class 'bool' >

cellComments

Value must be one of { 'atEnd' , 'asDisplayed' }

copies

Values must be of type <class 'int' >

draft

Values must be of type <class 'bool' >

errors

Value must be one of { 'blank' , 'displayed' , 'dash' , 'NA' }

firstPageNumber

Values must be of type <class 'int' >

fitToHeight

Values must be of type <class 'int' >

fitToPage

fitToWidth

Values must be of type <class 'int' >

classmethod from_tree(*node*)

Create object from XML

horizontalDpi

Values must be of type <class 'int'>

id

Values must be of type <class 'str'>

orientation

Value must be one of { 'portrait', 'default', 'landscape'}

pageOrder

Value must be one of { 'overThenDown', 'downThenOver'}

paperHeight

paperSize

Values must be of type <class 'int'>

paperWidth

scale

Values must be of type <class 'int'>

sheet_properties

Proxy property

tagname = 'pageSetup'

useFirstPageNumber

Values must be of type <class 'bool'>

usePrinterDefaults

Values must be of type <class 'bool'>

verticalDpi

Values must be of type <class 'int'>

openpyxl.worksheet.pagebreak module

class openpyxl.worksheet.pagebreak.Break(*id=0*, *min=0*, *max=16383*, *man=True*, *pt=None*)

基类: *openpyxl.descriptors.serialisable.Serialisable*

id

Values must be of type <class 'int'>

man

Values must be of type <class 'bool'>

max
Values must be of type <class 'int' >

min
Values must be of type <class 'int' >

pt
Values must be of type <class 'bool' >

tagname = 'brk'

class openpyxl.worksheet.pagebreak.**ColBreak**(*count=None*, *manualBreakCount=None*, *brk=()*)
基类: *openpyxl.worksheet.pagebreak.RowBreak*

brk
A sequence (list or tuple) that may only contain objects of the declared type

count

manualBreakCount

tagname = 'colBreaks'

openpyxl.worksheet.pagebreak.**PageBreak**
openpyxl.worksheet.pagebreak.RowBreak 的别名

class openpyxl.worksheet.pagebreak.**RowBreak**(*count=None*, *manualBreakCount=None*, *brk=()*)
基类: *openpyxl.descriptors.serialisable.Serialisable*

append(*brk=None*)
Add a page break

brk
A sequence (list or tuple) that may only contain objects of the declared type

count

manualBreakCount

tagname = 'rowBreaks'

openpyxl.worksheet.picture module

class openpyxl.worksheet.picture.**SheetBackgroundPicture**
基类: *openpyxl.descriptors.serialisable.Serialisable*

tagname = 'sheetBackgroundPicture'

openpyxl.worksheet.properties module

Worksheet Properties

class openpyxl.worksheet.properties.**Outline**(*applyStyles=None*, *summaryBelow=None*, *summaryRight=None*, *showOutlineSymbols=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

applyStyles

Values must be of type <class 'bool' >

showOutlineSymbols

Values must be of type <class 'bool' >

summaryBelow

Values must be of type <class 'bool' >

summaryRight

Values must be of type <class 'bool' >

tagname = 'outlinePr'

class openpyxl.worksheet.properties.**PageSetupProperties**(*autoPageBreaks=None*, *fitToPage=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

autoPageBreaks

Values must be of type <class 'bool' >

fitToPage

Values must be of type <class 'bool' >

tagname = 'pageSetUpPr'

class openpyxl.worksheet.properties.**WorksheetProperties**(*codeName=None*, *enableFormatConditionsCalculation=None*, *filterMode=None*, *published=None*, *syncHorizontal=None*, *syncRef=None*, *syncVertical=None*, *transitionEvaluation=None*, *transitionEntry=None*, *tabColor=None*, *outlinePr=None*, *pageSetUpPr=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

codeName

Values must be of type <class 'str' >

enableFormatConditionsCalculation

Values must be of type <class 'bool' >

filterMode

Values must be of type <class 'bool' >

outlinePr
Values must be of type <class 'openpyxl.worksheet.properties.Outline' >

pageSetUpPr
Values must be of type <class 'openpyxl.worksheet.properties.PageSetupProperties' >

published
Values must be of type <class 'bool' >

syncHorizontal
Values must be of type <class 'bool' >

syncRef
Values must be of type <class 'str' >

syncVertical
Values must be of type <class 'bool' >

tabColor
Values must be of type <class 'openpyxl.styles.colors.Color' >

tagname = 'sheetPr'

transitionEntry
Elements

transitionEvaluation
Values must be of type <class 'bool' >

openpyxl.worksheet.protection module

class openpyxl.worksheet.protection.**SheetProtection**(*sheet=False, objects=False, scenarios=False, formatCells=True, formatRows=True, formatColumns=True, insertColumns=True, insertRows=True, insertHyperlinks=True, deleteColumns=True, deleteRows=True, selectLockedCells=False, selectUnlockedCells=False, sort=True, autoFilter=True, pivotTables=True, password=None, algorithmName=None, saltValue=None, spinCount=None, hashValue=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*, openpyxl.worksheet.protection._Protected

Information about protection of various aspects of a sheet. True values mean that protection for the object or action is active This is the **default** when protection is active, ie. users cannot do something

`algorithmName`
Values must be of type <class 'str'>

`autoFilter`
Values must be of type <class 'bool'>

`deleteColumns`
Values must be of type <class 'bool'>

`deleteRows`
Values must be of type <class 'bool'>

`disable()`

`enable()`

`enabled`
Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

`formatCells`
Values must be of type <class 'bool'>

`formatColumns`
Values must be of type <class 'bool'>

`formatRows`
Values must be of type <class 'bool'>

`hashValue`

`insertColumns`
Values must be of type <class 'bool'>

`insertHyperlinks`
Values must be of type <class 'bool'>

`insertRows`
Values must be of type <class 'bool'>

`objects`
Values must be of type <class 'bool'>

`pivotTables`
Values must be of type <class 'bool'>

`saltValue`

`scenarios`
Values must be of type <class 'bool'>

`selectLockedCells`

Values must be of type <class ' bool' >

`selectUnlockedCells`

Values must be of type <class ' bool' >

`set_password`(*value=", already_hashed=False*)

Set a password on this sheet.

`sheet`

Values must be of type <class ' bool' >

`sort`

Values must be of type <class ' bool' >

`spinCount`

Values must be of type <class ' int' >

`tagname = 'sheetProtection'`

openpyxl.worksheet.related module

`class openpyxl.worksheet.related.Related`(*id=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`id`

Values must be of type <class ' str' >

`to_tree`(*tagname, idx=None*)

openpyxl.worksheet.scenario module

`class openpyxl.worksheet.scenario.InputCells`(*r=None, deleted=False, undone=False, val=None, numFmtId=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

`deleted`

Values must be of type <class ' bool' >

`numFmtId`

Values must be of type <class ' int' >

`r`

Values must be of type <class ' str' >

`tagname = 'inputCells'`

`undone`

Values must be of type <class ' bool' >

val

Values must be of type <class 'str' >

class openpyxl.worksheet.scenario.**Scenario**(*inputCells=()*, *name=None*, *locked=False*, *hidden=False*, *count=None*, *user=None*, *comment=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

comment

Values must be of type <class 'str' >

count

hidden

Values must be of type <class 'bool' >

inputCells

A sequence (list or tuple) that may only contain objects of the declared type

locked

Values must be of type <class 'bool' >

name

Values must be of type <class 'str' >

tagname = 'scenario'

user

Values must be of type <class 'str' >

class openpyxl.worksheet.scenario.**ScenarioList**(*scenario=()*, *current=None*, *show=None*, *sqref=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

append(*scenario*)

current

Values must be of type <class 'int' >

scenario

A sequence (list or tuple) that may only contain objects of the declared type

show

Values must be of type <class 'int' >

sqref

Values must be of type <class 'openpyxl.worksheet.cell_range.MultiCellRange' >

tagname = 'scenarios'

openpyxl.worksheet.smart_tag module

`class openpyxl.worksheet.smart_tag.`**`CellSmartTag`**`(`*cellSmartTagPr=(), type=None, deleted=False, xmlBased=False*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`cellSmartTagPr`

A sequence (list or tuple) that may only contain objects of the declared type

`deleted`

Values must be of type <class 'bool'>

`tagname = 'cellSmartTag'`

`type`

Values must be of type <class 'int'>

`xmlBased`

Values must be of type <class 'bool'>

`class openpyxl.worksheet.smart_tag.`**`CellSmartTagPr`**`(`*key=None, val=None*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`key`

Values must be of type <class 'str'>

`tagname = 'cellSmartTagPr'`

`val`

Values must be of type <class 'str'>

`class openpyxl.worksheet.smart_tag.`**`CellSmartTags`**`(`*cellSmartTag=(), r=None*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`cellSmartTag`

A sequence (list or tuple) that may only contain objects of the declared type

`r`

Values must be of type <class 'str'>

`tagname = 'cellSmartTags'`

`class openpyxl.worksheet.smart_tag.`**`SmartTags`**`(`*cellSmartTags=()*`)`

基类：*openpyxl.descriptors.serialisable.Serialisable*

`cellSmartTags`

A sequence (list or tuple) that may only contain objects of the declared type

`tagname = 'smartTags'`

openpyxl.worksheet.table module

class openpyxl.worksheet.table.**Table**(*id=1, displayName=None, ref=None, name=None, comment=None, tableType=None, headerRowCount=1, insertRow=None, insertRowShift=None, totalsRowCount=None, totalsRowShown=None, published=None, headerRowDxfId=None, dataDxfId=None, totalsRowDxfId=None, headerRowBorderDxfId=None, tableBorderDxfId=None, totalsRowBorderDxfId=None, headerRowCellStyle=None, dataCellStyle=None, totalsRowCellStyle=None, connectionId=None, autoFilter=None, sortState=None, tableColumns=(), tableStyleInfo=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

autoFilter

Values must be of type <class 'openpyxl.worksheet.filters.AutoFilter'>

column_names

comment

Values must be of type <class 'str'>

connectionId

Values must be of type <class 'int'>

dataCellStyle

Values must be of type <class 'str'>

dataDxfId

Values must be of type <class 'int'>

displayName

Values must be of type <class 'str'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

headerRowBorderDxfId

Values must be of type <class 'int'>

headerRowCellStyle

Values must be of type <class 'str'>

headerRowCount

Values must be of type <class 'int'>

headerRowDxfId

Values must be of type <class 'int'>

id

Values must be of type <class ‘int’ >

insertRow

Values must be of type <class ‘bool’ >

insertRowShift

Values must be of type <class ‘bool’ >

mime_type = 'application/vnd.openxmlformats-officedocument.spreadsheetml.table+xml'

name

Values must be of type <class ‘str’ >

path

Return path within the archive

published

Values must be of type <class ‘bool’ >

ref

sortState

Values must be of type <class ‘openpyxl.worksheet.filters.SortState’ >

tableBorderDxfId

Values must be of type <class ‘int’ >

tableColumns

Wrap a sequence in an containing object

tableStyleInfo

Values must be of type <class ‘openpyxl.worksheet.table.TableStyleInfo’ >

tableType

Value must be one of { ‘xml’ , ‘queryTable’ , ‘worksheet’ }

tagname = 'table'

to_tree()

totalsRowBorderDxfId

Values must be of type <class ‘int’ >

totalsRowCellStyle

Values must be of type <class ‘str’ >

totalsRowCount

Values must be of type <class ‘int’ >

totalsRowDxfId

Values must be of type <class ‘int’ >

totalsRowShown

Values must be of type <class 'bool'>

class openpyxl.worksheet.table.TableColumn(*id=None, uniqueName=None, name=None, totalsRowFunction=None, totalsRowLabel=None, queryTableFieldId=None, headerRowDxfId=None, dataDxfId=None, totalsRowDxfId=None, headerRowCellStyle=None, dataCellStyle=None, totalsRowCellStyle=None, calculatedColumnFormula=None, totalsRowFormula=None, xmlColumnPr=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

calculatedColumnFormula

Values must be of type <class 'openpyxl.worksheet.table.TableFormula'>

dataCellStyle

Values must be of type <class 'str'>

dataDxfId

Values must be of type <class 'int'>

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList'>

classmethod from_tree(*node*)

Create object from XML

headerRowCellStyle

Values must be of type <class 'str'>

headerRowDxfId

Values must be of type <class 'int'>

id

Values must be of type <class 'int'>

name

Values must be of type <class 'str'>

queryTableFieldId

Values must be of type <class 'int'>

tagname = 'tableColumn'

totalsRowCellStyle

Values must be of type <class 'str'>

totalsRowDxfId

Values must be of type <class 'int'>

totalsRowFormula

Values must be of type <class 'openpyxl.worksheet.table.TableFormula'>

totalsRowFunction

Value must be one of { 'custom', 'sum', 'countNums', 'min', 'stdDev', 'max', 'average', 'count', 'var' }

totalsRowLabel

Values must be of type <class 'str'>

uniqueName

Values must be of type <class 'str'>

xmlColumnPr

Values must be of type <class 'openpyxl.worksheet.table.XMLColumnProps'>

class openpyxl.worksheet.table.**TableFormula**(*array=None*, *attr_text=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

array

Values must be of type <class 'bool'>

attr_text

tagname = 'tableFormula'

text

Aliases can be used when either the desired attribute name is not allowed or confusing in Python (eg. "type") or a more descriptve name is desired (eg. "underline" for "u")

class openpyxl.worksheet.table.**TableList**

基类：dict

add(*table*)

get(*name=None*, *table_range=None*)

Return the value for key if key is in the dictionary, else default.

items() → a set-like object providing a view on D's items

class openpyxl.worksheet.table.**TableNameDescriptor**(**args*, ***kw*)

基类：*openpyxl.descriptors.base.String*

Table names cannot have spaces in them

class openpyxl.worksheet.table.**TablePartList**(*count=None*, *tablePart=()*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

append(*part*)

count

tablePart

A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'tableParts'

class openpyxl.worksheet.table.**TableStyleInfo**(*name=None, showFirstColumn=None, showLastColumn=None, showRowStripes=None, showColumnStripes=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

name

Values must be of type <class 'str' >

showColumnStripes

Values must be of type <class 'bool' >

showFirstColumn

Values must be of type <class 'bool' >

showLastColumn

Values must be of type <class 'bool' >

showRowStripes

Values must be of type <class 'bool' >

tagname = 'tableStyleInfo'

class openpyxl.worksheet.table.**XMLColumnProps**(*mapId=None, xpath=None, denormalized=None, xmlDataType=None, extLst=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

denormalized

Values must be of type <class 'bool' >

extLst

Values must be of type <class 'openpyxl.descriptors.excel.ExtensionList' >

mapId

Values must be of type <class 'int' >

tagname = 'xmlColumnPr'

xmlDataType

Values must be of type <class 'str' >

xpath

Values must be of type <class 'str' >

openpyxl.worksheet.table.**tostring**(*element, *, encoding='utf-8', method=None, short_empty_elements=True*)

Generate string representation of XML element.

All subelements are included. If encoding is "unicode", a string is returned. Otherwise a bytestring is returned.

element is an Element instance, *encoding* is an optional output encoding defaulting to US-ASCII, *method* is an optional output which can be one of "xml" (default), "html", "text" or "c14n".

Returns an (optionally) encoded string containing the XML data.

openpyxl.worksheet.views module

class openpyxl.worksheet.views.**Pane**(*xSplit=None*, *ySplit=None*, *topLeftCell=None*, *activePane='topLeft'*, *state='split'*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

activePane

Value must be one of {'bottomRight', 'topLeft', 'topRight', 'bottomLeft'}

state

Value must be one of {'frozen', 'frozenSplit', 'split'}

topLeftCell

Values must be of type <class 'str'>

xSplit

Values must be of type <class 'float'>

ySplit

Values must be of type <class 'float'>

class openpyxl.worksheet.views.**Selection**(*pane=None*, *activeCell='A1'*, *activeCellId=None*, *sqref='A1'*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

activeCell

Values must be of type <class 'str'>

activeCellId

Values must be of type <class 'int'>

pane

Value must be one of {'bottomRight', 'topLeft', 'topRight', 'bottomLeft'}

sqref

Values must be of type <class 'str'>

`class openpyxl.worksheet.views.SheetView`(*windowProtection=None, showFormulas=None, showGridLines=None, showRowColHeaders=None, showZeros=None, rightToLeft=None, tabSelected=None, showRuler=None, showOutlineSymbols=None, defaultGridColor=None, showWhiteSpace=None, view=None, topLeftCell=None, colorId=None, zoomScale=None, zoomScaleNormal=None, zoomScaleSheetLayoutView=None, zoomScalePageLayoutView=None, zoomToFit=None, workbookViewId=0, selection=None, pane=None*)

基类：*openpyxl.descriptors.serialisable.Serialisable*

Information about the visible portions of this sheet.

`colorId`

Values must be of type <class 'int' >

`defaultGridColor`

Values must be of type <class 'bool' >

`pane`

Values must be of type <class 'openpyxl.worksheet.views.Pane' >

`rightToLeft`

Values must be of type <class 'bool' >

`selection`

A sequence (list or tuple) that may only contain objects of the declared type

`showFormulas`

Values must be of type <class 'bool' >

`showGridLines`

Values must be of type <class 'bool' >

`showOutlineSymbols`

Values must be of type <class 'bool' >

`showRowColHeaders`

Values must be of type <class 'bool' >

`showRuler`

Values must be of type <class 'bool' >

`showWhiteSpace`

Values must be of type <class 'bool' >

`showZeros`

Values must be of type <class 'bool' >

tabSelected
Values must be of type <class ‘bool’ >

tagname = 'sheetView'

topLeftCell
Values must be of type <class ‘str’ >

view
Value must be one of { ‘pageLayout’ , ‘normal’ , ‘pageBreakPreview’ }

windowProtection
Values must be of type <class ‘bool’ >

workbookViewId
Values must be of type <class ‘int’ >

zoomScale
Values must be of type <class ‘int’ >

zoomScaleNormal
Values must be of type <class ‘int’ >

zoomScalePageLayoutView
Values must be of type <class ‘int’ >

zoomScaleSheetLayoutView
Values must be of type <class ‘int’ >

zoomToFit
Values must be of type <class ‘bool’ >

class openpyxl.worksheet.views.**SheetViewList**(*sheetView=None*, *extLst=None*)
基类：*openpyxl.descriptors.serialisable.Serialisable*

extLst
Values must be of type <class ‘openpyxl.descriptors.excel.ExtensionList’ >

sheetView
A sequence (list or tuple) that may only contain objects of the declared type

tagname = 'sheetViews'

openpyxl.worksheet.worksheet module

Worksheet is the 2nd-level container in Excel.

class openpyxl.worksheet.worksheet.**Worksheet**(*parent*, *title=None*)
基类：openpyxl.workbook.child._WorkbookChild

Represents a worksheet.

Do not create worksheets yourself, use `openpyxl.workbook.Workbook.create_sheet()` instead

`BREAK_COLUMN = 2`

`BREAK_NONE = 0`

`BREAK_ROW = 1`

`ORIENTATION_LANDSCAPE = 'landscape'`

`ORIENTATION_PORTRAIT = 'portrait'`

`PAPERSIZE_A3 = '8'`

`PAPERSIZE_A4 = '9'`

`PAPERSIZE_A4_SMALL = '10'`

`PAPERSIZE_A5 = '11'`

`PAPERSIZE_EXECUTIVE = '7'`

`PAPERSIZE_LEDGER = '4'`

`PAPERSIZE_LEGAL = '5'`

`PAPERSIZE_LETTER = '1'`

`PAPERSIZE_LETTER_SMALL = '2'`

`PAPERSIZE_STATEMENT = '6'`

`PAPERSIZE_TABLOID = '3'`

`SHEETSTATE_HIDDEN = 'hidden'`

`SHEETSTATE_VERYHIDDEN = 'veryHidden'`

`SHEETSTATE_VISIBLE = 'visible'`

`active_cell`

add_chart(*chart*, *anchor=None*)

Add a chart to the sheet Optionally provide a cell for the top-left anchor

add_data_validation(*data_validation*)

Add a data-validation object to the sheet. The data-validation object defines the type of data-validation to be applied and the cell or range of cells it should apply to.

add_image(*img*, *anchor=None*)

Add an image to the sheet. Optionally provide a cell for the top-left anchor

add_pivot(*pivot*)

add_table(*table*)

Check for duplicate name in definedNames and other worksheet tables before adding table.

append(*iterable*)

Appends a group of values at the bottom of the current sheet.

- If it' s a list: all values are added in order, starting from the first column
- If it' s a dict: values are assigned to the columns indicated by the keys (numbers or letters)

参数 iterable (*list/tuple/range/generator or dict*) – list, range or generator, or dict containing values to append

Usage:

- append(['This is A1' , 'This is B1' , 'This is C1'])
- **or** append({ 'A' : 'This is A1' , 'C' : 'This is C1' })
- **or** append({1 : 'This is A1' , 3 : 'This is C1' })

Raise TypeError when iterable is neither a list/tuple nor a dict

calculate_dimension()

Return the minimum bounding range for all cells containing data (ex. 'A1:M24')

返回类型 string

cell(*row*, *column*, *value=None*)

Returns a cell object based on the given coordinates.

Usage: cell(row=15, column=1, value=5)

Calling *cell* creates cells in memory when they are first accessed.

参数

- **row** (*int*) – row index of the cell (e.g. 4)
- **column** (*int*) – column index of the cell (e.g. 3)
- **value** (*numeric or time or string or bool or none*) – value of the cell (e.g. 5)

返回类型 *openpyxl.cell.cell.Cell*

columns

Produces all cells in the worksheet, by column (see *iter_cols()*)

delete_cols(*idx*, *amount=1*)

Delete column or columns from col==idx

delete_rows(*idx*, *amount=1*)

Delete row or rows from row==idx

dimensions

Returns the result of *calculate_dimension()*

freeze_panes

insert_cols(*idx*, *amount=1*)

Insert column or columns before col==idx

insert_rows(*idx*, *amount=1*)

Insert row or rows before row==idx

iter_cols(*min_col=None*, *max_col=None*, *min_row=None*, *max_row=None*, *values_only=False*)

Produces cells from the worksheet, by column. Specify the iteration range using indices of rows and columns.

If no indices are specified the range starts at A1.

If no cells are in the worksheet an empty tuple will be returned.

参数

- **min_col** (*int*) – smallest column index (1-based index)
- **min_row** (*int*) – smallest row index (1-based index)
- **max_col** (*int*) – largest column index (1-based index)
- **max_row** (*int*) – largest row index (1-based index)
- **values_only** (*bool*) – whether only cell values should be returned

返回类型 generator

iter_rows(*min_row=None*, *max_row=None*, *min_col=None*, *max_col=None*, *values_only=False*)

Produces cells from the worksheet, by row. Specify the iteration range using indices of rows and columns.

If no indices are specified the range starts at A1.

If no cells are in the worksheet an empty tuple will be returned.

参数

- **min_col** (*int*) – smallest column index (1-based index)
- **min_row** (*int*) – smallest row index (1-based index)
- **max_col** (*int*) – largest column index (1-based index)
- **max_row** (*int*) – largest row index (1-based index)
- **values_only** (*bool*) – whether only cell values should be returned

返回类型 generator

max_column

The maximum column index containing data (1-based)

Type int

`max_row`

The maximum row index containing data (1-based)

Type int

`merge_cells`(*range_string=None*, *start_row=None*, *start_column=None*, *end_row=None*, *end_column=None*)

Set merge on a cell range. Range is a cell range (e.g. A1:E1)

`merged_cell_ranges`

Return a copy of cell ranges

注解： Deprecated: Use ws.merged_cells.ranges

`mime_type = 'application/vnd.openxmlformats-officedocument.spreadsheetml.worksheet+xml'`

`min_column`

The minimum column index containing data (1-based)

Type int

`min_row`

The minimium row index containing data (1-based)

Type int

`move_range`(*cell_range*, *rows=0*, *cols=0*, *translate=False*)

Move a cell range by the number of rows and/or columns: down if rows > 0 and up if rows < 0 right if cols > 0 and left if cols < 0 Existing cells will be overwritten. Formulae and references will not be updated.

`page_breaks`

`print_area`

The print area for the worksheet, or None if not set. To set, supply a range like ‘A1:D4’ or a list of ranges.

`print_title_cols`

Columns to be printed at the left side of every page (ex: ‘A:C’)

`print_title_rows`

Rows to be printed at the top of every page (ex: ‘1:3’)

`print_titles`

`rows`

Produces all cells in the worksheet, by row (see *`iter_rows()`*)

Type generator

`selected_cell`

`set_printer_settings`(*paper_size*, *orientation*)

Set printer settings

`sheet_view`

`show_gridlines`

`show_summary_below`

`show_summary_right`

`tables`

`unmerge_cells`(*range_string=None*, *start_row=None*, *start_column=None*, *end_row=None*, *end_column=None*)

Remove merge on a cell range. Range is a cell range (e.g. A1:E1)

`values`

Produces all cell values in the worksheet, by row

Type generator

openpyxl.writer package

Submodules

openpyxl.writer.excel module

Write a .xlsx file.

`class openpyxl.writer.excel.ExcelWriter`(*workbook*, *archive*)

基类：`object`

Write a workbook object to an Excel file.

`save()`

Write data into the archive.

`write_data()`

Write the various xml files into the zip archive.

`write_worksheet`(*ws*)

`openpyxl.writer.excel.save_virtual_workbook`(*workbook*)

Return an in-memory workbook, suitable for a Django response.

注解： Deprecated: Use a NamedTemporaryFile `openpyxl.writer.excel.save_workbook`(*workbook*, *filename*)

Save the given workbook on the filesystem under the name filename.

参数

- **`workbook`** (`openpyxl.workbook.Workbook`) – the workbook to save
- **`filename`** (*string*) – the path to which save the workbook

返回类型 bool

`openpyxl.writer.excel.tostring`(*element*, ***, *encoding='utf-8'*, *method=None*, *short_empty_elements=True*)

Generate string representation of XML element.

All subelements are included. If encoding is "unicode", a string is returned. Otherwise a bytestring is returned.

element is an Element instance, *encoding* is an optional output encoding defaulting to US-ASCII, *method* is an optional output which can be one of "xml" (default), "html", "text" or "c14n".

Returns an (optionally) encoded string containing the XML data.

openpyxl.writer.theme module

Write the theme xml based on a fixed string.

`openpyxl.writer.theme.write_theme`()

Write the theme xml.

openpyxl.xml package

Collection of XML resources compatible across different Python versions

`openpyxl.xml.defusedxml_available`()

`openpyxl.xml.defusedxml_env_set`()

`openpyxl.xml.lxml_available`()

`openpyxl.xml.lxml_env_set`()

Submodules

openpyxl.xml.constants module

Constants for fixed paths in a file and xml namespace urls.

openpyxl.xml.functions module

XML compatability functions

openpyxl.xml.functions.**localname**(*node*)

openpyxl.xml.functions.**tostring**(*element*, ***, *encoding='utf-8'*, *method=None*, *short_empty_elements=True*)

Generate string representation of XML element.

All subelements are included. If encoding is "unicode", a string is returned. Otherwise a bytestring is returned.

element is an Element instance, *encoding* is an optional output encoding defaulting to US-ASCII, *method* is an optional output which can be one of "xml" (default), "html", "text" or "c14n".

Returns an (optionally) encoded string containing the XML data.

openpyxl.xml.functions.**whitespace**(*node*)

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

CHAPTER 10

发布说明

10.1 3.0.4 (2020-06-24)

10.1.1 Bugfixes

- #844 Find tables by name
- #1414 Worksheet protection missing in existing files
- #1439 Exception when reading files with external images
- #1452 Reading lots of merged cells is very slow.
- #1455 Read support for Bubble Charts.
- #1458 Preserve any indexed colours
- #1473 Reading many thousand of merged cells is really slow.
- #1474 Adding tables in write-only mode raises an exception.

10.1.2 Pull Requests

- PR377 Add support for finding tables by name or range.

10.2 3.0.3 (2020-01-20)

10.2.1 Bugfixes

- #1260 Exception when handling merged cells with hyperlinks
- #1373 Problems when both lxml and defusedxml are installed
- #1385 CFVO with incorrect values cannot be processed

10.3 3.0.2 (2019-11-25)

10.3.1 Bug fixes

- #1267 DeprecationError if both defusedxml and lxml are installed
- #1345 ws._current_row is higher than ws.max_row
- #1365 Border bottom style is not optional when it should be
- #1367 Empty cells in read-only, values-only mode are sometimes returned as ReadOnlyCells
- #1368 Cannot add page breaks to existing worksheets if none exist already

10.3.2 Pull Requests

- PR359 Improvements to the documentation

10.4 3.0.1 (2019-11-14)

10.4.1 Bugfixes

- #1250 Cannot read empty charts.

10.4.2 Pull Requests

- PR354 Fix for #1250
- PR352 TableStyleElement is a sequence

10.5 3.0.0 (2019-09-25)

10.5.1 Python 3.6+ only release

10.6 2.6.4 (2019-09-25)

10.6.1 Final release for Python 2.7 and 3.5

10.6.2 Bugfixes

- ‘ #1330 <https://bitbucket.org/openpyxl/openpyxl/issues/1330>‘_ Cannot save workbooks with comments more than once.

10.7 2.6.3 (2019-08-19)

10.7.1 Bugfixes

- #1237 Fix 3D charts.
- #1290 Minimum for holeSize in Doughnut charts too high
- #1291 Warning for MergedCells with comments
- #1296 Pagebreaks duplicated
- #1309 Workbook has no default CellStyle
- #1330 Workbooks with comments cannot be saved multiple times

10.7.2 Pull Requests

- PR344 Make sure NamedStyles number formats are correctly handled

10.8 2.6.2 (2019-03-29)

10.8.1 Bugfixes

- #1173 Workbook has no _date_formats attribute
- #1190 Cannot create charts for worksheets with quotes in the title
- #1228 MergedCells not removed when range is unmerged

- #1232 Link to pivot table lost from charts
- #1233 Chart colours change after saving
- #1236 Cannot use ws.cell in read-only mode with Python 2.7

10.9 2.6.1 (2019-03-04)

10.9.1 Bugfixes

- #1174 ReadOnlyCell.is_date does not work properly
- #1175 Cannot read Google Docs spreadsheet with a Pivot Table
- #1180 Charts created with openpyxl cannot be styled
- #1181 Cannot handle some numpy number types
- #1182 Exception when reading unknowable number formats
- #1186 Only last formatting rule for a range loaded
- #1191 Give MergedCell a *value* attribute
- #1193 Cannot process worksheets with comments
- #1197 Cannot process worksheets with both row and page breaks
- #1204 Cannot reset dimensions in ReadOnlyWorksheets
- #1211 Incorrect descriptor in ParagraphProperties
- #1213 Missing *hier* attribute in PageField raises an exception

10.10 2.6.0 (2019-02-06)

10.10.1 Bugfixes

- #1162 Exception on tables with names containing spaces.
- #1170 Cannot save files with existing images.

10.11 2.6.-b1 (2019-01-08)

10.11.1 Bugfixes

- #1141 Cannot use read-only mode with stream

- #1143 Hyperlinks always set on A1
- #1151 Internal row counter not initialised when reading files
- #1152 Exception raised on out of bounds date

10.12 2.6-a1 (2018-11-21)

10.12.1 Major changes

- Implement robust for merged cells so that these can be formatted the way Excel does without confusion. Thanks to Magnus Schieder.

10.12.2 Minor changes

- Add support for worksheet scenarios
- Add read support for chartsheets
- Add method for moving ranges of cells on a worksheet
- Drop support for Python 3.4
- Last version to support Python 2.7

10.12.3 Deprecations

- Type inference and coercion for cell values

10.13 2.5.14 (2019-01-23)

10.13.1 Bugfixes

- #1150 Correct typo in LineProperties
- #1142 Exception raised for unsupported image files
- #1159 Exception raised when cannot find source for non-local cache object

10.13.2 Pull Requests

- PR301 Add support for nested brackets to the tokeniser
- PR303 Improvements on handling nested brackets in the tokeniser

10.14 2.5.13 (brown bag)

10.15 2.5.12 (2018-11-29)

10.15.1 Bugfixes

- #1130 Overwriting default font in Normal style affects library default
- #1133 Images not added to anchors.
- #1134 Cannot read pivot table formats without dxId
- #1138 Repeated registration of simple filter could lead to memory leaks

10.15.2 Pull Requests

- PR300 Use defusedxml if available

10.16 2.5.11 (2018-11-21)

10.16.1 Pull Requests

- PR295 Improved handling of missing rows
- PR296 Add support for defined names to tokeniser

10.17 2.5.10 (2018-11-13)

10.17.1 Bugfixes

- #1114 Empty column dimensions should not be saved.

10.17.2 Pull Requests

- PR285 Tokenizer failure for quoted sheet name in second half of range
- PR289 Improved error detection in ranges.

10.18 2.5.9 (2018-10-19)

10.18.1 Bugfixes

- #1000 Clean AutoFilter name definitions
- #1106 Attribute missing from Shape object
- #1109 Failure to read all DrawingML means workbook can’t be read

10.18.2 Pull Requests

- PR281 Allow newlines in formulae
- PR284 Fix whitespace in front of infix operator in formulae

10.19 2.5.8 (2018-09-25)

- #877 Cannot control how missing values are displayed in charts.
- #948 Cell references can’t be used for chart titles
- #1095 Params in iter_cols and iter_rows methods are slightly wrong.

10.20 2.5.7 (2018-09-13)

- #954 Sheet title containing % need quoting in references
- #1047 Cannot set quote prefix
- #1093 Pandas timestamps raise KeyError

10.21 2.5.6 (2018-08-30)

- #832 Read-only mode can leave find-handles open when reading dimensions
- #933 Set a worksheet directly as active
- #1086 Internal row counter not adjusted when rows are deleted or inserted

10.22 2.5.5 (2018-08-04)

10.22.1 Bugfixes

- #1049 Files with Mac epoch are read incorrectly
- #1058 Cannot copy merged cells
- #1066 Cannot access ws.active_cell

10.22.2 Pull Requests

- PR267 Introduce read-support for images

10.23 2.5.4 (2018-06-07)

10.23.1 Bugfixes

- #1025 Cannot read files with 3D charts.
- #1030 Merged cells take a long time to parse

10.23.2 Minor changes

- Improve read support for pivot tables and don’t always create a Filters child for filterColumn objects.
- *Support folding rows* <https://bitbucket.org/openpyxl/openpyxl/pull-requests/259/fold-rows>‘_

10.24 2.5.3 (2018-04-18)

10.24.1 Bugfixes

- #983 Warning level too aggressive.
- #1015 Alignment and protection values not saved for named styles.
- #1017 Deleting elements from a legend doesn’t work.
- #1018 Index names repeated for every row in dataframe.
- #1020 Worksheet protection not being stored.
- #1023 Exception raised when reading a tooltip.

10.25 2.5.2 (2018-04-06)

10.25.1 Bugfixes

- #949 High memory use when reading text-heavy files.
- #970 Copying merged cells copies references.
- #978 Cannot set comment size.
- #985 Exception when trying to save workbooks with no views.
- #995 Cannot delete last row or column.
- #1002 Cannot read Drawings containing embedded images.

10.25.2 Minor changes

- Support for dataframes with multiple columns and multiple indices.

10.26 2.5.1 (2018-03-12)

10.26.1 Bugfixes

- #934 Headers and footers not included in write-only mode.
- #960 Deprecation warning raised when using ad-hoc access in read-only mode.
- #964 Not all cells removed when deleting multiple rows.
- #966 Cannot read 3d bar chart correctly.
- #967 Problems reading some charts.
- #968 Worksheets with SHA protection become corrupted after saving.
- #974 Problem when deleting ragged rows or columns.
- #976 GroupTransforms and GroupShapeProperties have incorrect descriptors
- Make sure that headers and footers in chartsheets are included in the file

10.27 2.5.0 (2018-01-24)

10.27.1 Minor changes

- Correct definition for Connection Shapes. Related to # 958

10.28 2.5.0-b2 (2018-01-19)

10.28.1 Bugfixes

- #915 TableStyleInfo has no required attributes
- #925 Cannot read files with 3D drawings
- #926 Incorrect version check in installer
- Cell merging uses transposed parameters
- #928 ExtLst missing keyword for PivotFields
- #932 Inf causes problems for Excel
- #952 Cannot load table styles with custom names

10.28.2 Major Changes

- You can now insert and delete rows and columns in worksheets

10.28.3 Minor Changes

- pip now handles which Python versions can be used.

10.29 2.5.0-b1 (2017-10-19)

10.29.1 Bugfixes

- #812 Explicitly support for multiple cell ranges in conditonal formatting
- #827 Non-contiguous cell ranges in validators get merged
- #837 Empty data validators create invalid Excel files
- #860 Large validation ranges use lots of memory
- #876 Unicode in chart axes not handled correctly in Python 2
- #882 ScatterCharts have defective axes
- #885 Charts with empty numVal elements cannot be read
- #894 Scaling options from existing files ignored
- #895 Charts with PivotSource cannot be read
- #903 Cannot read gradient fills

- #904 Quotes in number formats could be treated as datetimes

10.29.2 Major Changes

worksheet.cell() no longer accepts a *coordinate* parameter. The syntax is now *ws.cell(row, column, value=None)*

10.29.3 Minor Changes

Added CellRange and MultiCellRange types (thanks to Laurent LaPorte for the suggestion) as a utility type for things like data validations, conditional formatting and merged cells.

10.29.4 Deprecations

ws.merged_cell_ranges has been deprecated because MultiCellRange provides sufficient functionality

10.30 2.5.0-a3 (2017-08-14)

10.30.1 Bugfixes

- #848 Reading workbooks with Pie Charts raises an exception
- #857 Pivot Tables without Worksheet Sources raise an exception

10.31 2.5.0-a2 (2017-06-25)

10.31.1 Major Changes

- Read support for charts

10.31.2 Bugfixes

- #833 Cannot access chartsheets by title
- #834 Preserve workbook views
- #841 Incorrect classification of a datetime

10.32 2.5.0-a1 (2017-05-30)

10.32.1 Compatibility

- Dropped support for Python 2.6 and 3.3. openpyxl will not run with Python 2.6

10.32.2 Major Changes

- Read/write support for pivot tables

10.32.3 Deprecations

- Dropped the anchor method from images and additional constructor arguments

10.32.4 Bugfixes

- #779 Fails to recognise Chinese date format‘
- #828 Include hidden cells in charts‘

10.32.5 Pull requests

- 163 Improved GradientFill

10.32.6 Minor changes

- Remove deprecated methods from Cell
- Remove deprecated methods from Worksheet
- Added read/write support for the datetime type for cells

10.33 2.4.11 (2018-01-24)

- #957 https://bitbucket.org/openpyxl/openpyxl/issues/957 Relationship type for tables is borked

10.34 2.4.10 (2018-01-19)

10.34.1 Bugfixes

- #912 https://bitbucket.org/openpyxl/openpyxl/issues/912 Copying objects uses shallow copy
- #921 https://bitbucket.org/openpyxl/openpyxl/issues/921 API documentation not generated automatically
- #927 https://bitbucket.org/openpyxl/openpyxl/issues/927 Exception raised when adding coloured borders together
- #931 https://bitbucket.org/openpyxl/openpyxl/issues/931 Number formats not correctly deduplicated

10.34.2 Pull requests

- 203 https://bitbucket.org/openpyxl/openpyxl/pull-requests/203/ Correction to worksheet protection description
- 210 https://bitbucket.org/openpyxl/openpyxl/pull-requests/210/ Some improvements to the API docs
- 211 https://bitbucket.org/openpyxl/openpyxl/pull-requests/211/ Improved deprecation decorator
- 218 https://bitbucket.org/openpyxl/openpyxl/pull-requests/218/ Fix problems with deepcopy

10.35 2.4.9 (2017-10-19)

10.35.1 Bugfixes

- #809 Incomplete documentation of *copy_worksheet* method
- #811 Scoped definedNames not removed when worksheet is deleted
- #824 Raise an exception if a chart is used in multiple sheets
- #842 Non-ASCII table column headings cause an exception in Python 2
- #846 Conditional formats not supported in write-only mode
- #849 Conditional formats with no sqref cause an exception
- #859 Headers that start with a number conflict with font size
- #902 TableStyleElements don’t always have a condtional format
- #908 Read-only mode sometimes returns too many cells

10.35.2 Pull requests

- #179 Cells kept in a set
- #180 Support for Workbook protection
- #182 Read support for page breaks
- #183 Improve documentation of *copy_worksheet* method
- #198 Fix for #908

10.36 2.4.8 (2017-05-30)

10.36.1 Bugfixes

- AutoFilter.sortState being assignd to the ws.sortState
- #766 Sheetnames with apostrophes need additional escaping
- #729 Cannot open files created by Microsoft Dynamics
- #819 Negative percents not case correctly
- #821 Runtime imports can cause deadlock
- #855 Print area containing only columns leads to corrupt file

10.36.2 Minor changes

- Preserve any table styles

10.37 2.4.7 (2017-04-24)

10.37.1 Bugfixes

- #807 Sample files being included by mistake in sdist

10.38 2.4.6 (2017-04-14)

10.38.1 Bugfixes

- #776 Cannot apply formatting to plot area
- #780 Exception when element attributes are Python keywords

- #781 Exception raised when saving files with styled columns
- #785 Number formats for data labels are incorrect
- #788 Worksheet titles not quoted in defined names
- #800 Font underlines not read correctly

10.39 2.4.5 (2017-03-07)

10.39.1 Bugfixes

- #750 Adding images keeps file handles open
- #772 Exception for column-only ranges
- #773 Cannot copy worksheets with non-ascii titles on Python 2

10.39.2 Pull requests

- 161 Support for non-standard names for Workbook part.
- 162 Documentation correction

10.40 2.4.4 (2017-02-23)

10.40.1 Bugfixes

- #673 Add close method to workbooks
- #762 openpyxl can create files with invalid style indices
- #729 Allow images in write-only mode
- #744 Rounded corners for charts
- #747 Use repr when handling non-convertible objects
- #764 Hashing function is incorrect
- #765 Named styles share underlying array

10.40.2 Minor Changes

- Add roundtrip support for worksheet tables.

10.40.3 Pull requests

- 160 Don’ t init mimetypes more than once.

10.41 2.4.3 (unreleased)

bad release

10.42 2.4.2 (2017-01-31)

10.42.1 Bug fixes

- #727 DeprecationWarning is incorrect
- #734 Exception raised if userName is missing
- #739 Always provide a date1904 attribute
- #740 Hashes should be stored as Base64
- #743 Print titles broken on sheetnames with spaces
- #748 Workbook breaks when active sheet is removed
- #754 Incorrect descriptor for Filter values
- #756 Potential XXE vulerability
- #758 Cannot create files with page breaks and charts
- #759 Problems with worksheets with commas in their titles

10.42.2 Minor Changes

- Add unicode support for sheet name incrementation.

10.43 2.4.1 (2016-11-23)

10.43.1 Bug fixes

- #643 Make checking for duplicate sheet titles case insensitive
- #647 Trouble handling LibreOffice files with named styles
- #687 Directly assigned new named styles always refer to “Normal”

- #690 Cannot parse print titles with multiple sheet names
- #691 Cannot work with macro files created by LibreOffice
- Prevent duplicate differential styles
- #694 Allow sheet titles longer than 31 characters
- #697 Cannot unset hyperlinks
- #699 Exception raised when format objects use cell references
- #703 Copy height and width when copying comments
- #705 Incorrect content type for VBA macros
- #707 IndexError raised in read-only mode when accessing individual cells
- #711 Files with external links become corrupted
- #715 Cannot read files containing macro sheets
- #717 Details from named styles not preserved when reading files
- #722 Remove broken Print Title and Print Area definitions

10.43.2 Minor changes

- Add support for Python 3.6
- Correct documentation for headers and footers

10.43.3 Deprecations

Worksheet methods *get_named_range()* and *get_sqaured_range()*

10.43.4 Bug fixes

10.44 2.4.0 (2016-09-15)

10.44.1 Bug fixes

- #652 Exception raised when epoch is 1904
- #642 Cannot handle unicode in headers and footers in Python 2
- #646 Cannot handle unicode sheetnames in Python 2
- #658 Chart styles, and axis units should not be 0
- #663 Strings in external workbooks not unicode

10.44.2 Major changes

- Add support for builtin styles and include one for Pandas

10.44.3 Minor changes

- Add a *keep_links* option to *load_workbook*. External links contain cached copies of the external workbooks. If these are big it can be advantageous to be able to disable them.
- Provide an example for using cell ranges in DataValidation.
- PR 138 - add copy support to comments.

10.45 2.4.0-b1 (2016-06-08)

10.45.1 Minor changes

- Add an the alias *hide_drop_down* to DataValidation for *showDropDown* because that is how Excel works.

10.45.2 Bug fixes

- #625 Exception raises when inspecting EmptyCells in read-only mode
- #547 Functions for handling OOXML "escaped" ST_XStrings
- #629 Row Dimensions not supported in write-only mode
- #530 Problems when removing worksheets with charts
- #630 Cannot use SheetProtection in write-only mode

10.45.3 Features

- Add write support for worksheet tables

10.46 2.4.0-a1 (2016-04-11)

10.46.1 Minor changes

- Remove deprecated methods from DataValidation
- Remove deprecated methods from PrintPageSetup

- Convert AutoFilter to Serialisable and extend support for filters
- Add support for SortState
- Removed *use_iterators* keyword when loading workbooks. Use *read_only* instead.
- Removed *optimized_write* keyword for new workbooks. Use *write_only* instead.
- Improve print title support
- Add print area support
- New implementation of defined names
- New implementation of page headers and footers
- Add support for Python’s NaN
- Added iter_cols method for worksheets
- ws.rows and ws.columns now always return generators and start at the top of the worksheet
- Add a *values* property for worksheets
- Default column width changed to 8 as per the specification

10.46.2 Deprecations

- Cell anchor method
- Worksheet point_pos method
- Worksheet add_print_title method
- Worksheet HeaderFooter attribute, replaced by individual ones
- Flatten function for cells
- Workbook get_named_range, add_named_range, remove_named_range, get_sheet_names, get_sheet_by_name
- Comment text attribute
- Use of range strings deprecated for ws.iter_rows()
- Use of coordinates deprecated for ws.cell()
- Deprecate .copy() method for StyleProxy objects

10.46.3 Bug fixes

- #152 Hyperlinks lost when reading files
- #171 Add function for copying worksheets

- #386 Cells with inline strings considered empty
- #397 Add support for ranges of rows and columns
- #446 Workbook with definedNames corrupted by openpyxl
- #481 “safe” reserved ranges are not read from workbooks
- #501 Discarding named ranges can lead to corrupt files
- #574 Exception raised when using the class method to parse Relationships
- #579 Crashes when reading defined names with no content
- #597 Cannot read worksheets without coordinates
- #617 Customised named styles not correctly preserved

10.47 2.3.5 (2016-04-11)

10.47.1 Bug fixes

- #618 Comments not written in write-only mode

10.48 2.3.4 (2016-03-16)

10.48.1 Bug fixes

- #594 Content types might be missing when keeping VBA
- #599 Cells with only one cell look empty
- #607 Serialise NaN as “”

10.48.2 Minor changes

- Preserve the order of external references because formualae use numerical indices.
- Typo corrected in cell unit tests (PR 118)

10.49 2.3.3 (2016-01-18)

10.49.1 Bug fixes

- #540 Cannot read merged cells in read-only mode

- #565 Empty styled text blocks cannot be parsed
- #569 Issue warning rather than raise Exception raised for unparsable definedNames
- #575 Cannot open workbooks with embdedded OLE files
- #584 Exception when saving borders with attributes

10.49.2 Minor changes

- PR 103 Documentation about chart scaling and axis limits
- Raise an exception when trying to copy cells from other workbooks.

10.50 2.3.2 (2015-12-07)

10.50.1 Bug fixes

- #554 Cannot add comments to a worksheet when preserving VBA
- #561 Exception when reading phonetic text
- #562 DARKBLUE is the same as RED
- #563 Minimum for row and column indexes not enforced

10.50.2 Minor changes

- PR 97 One VML file per worksheet.
- PR 96 Correct descriptor for CharacterProperties.rtl
- #498 Metadata is not essential to use the package.

10.51 2.3.1 (2015-11-20)

10.51.1 Bug fixes

- #534 Exception when using columns property in read-only mode.
- #536 Incorrectly handle comments from Google Docs files.
- #539 Flexible value types for conditional formatting.
- #542 Missing content types for images.
- #543 Make sure images fit containers on all OSes.

- #544 Gracefully handle missing cell styles.
- #546 ExternalLink duplicated when editing a file with macros.
- #548 Exception with non-ASCII worksheet titles
- #551 Combine multiple LineCharts

10.51.2 Minor changes

- PR 88 Fix page margins in parser.

10.52 2.3.0 (2015-10-20)

10.52.1 Major changes

- Support the creation of chartsheets

10.52.2 Bug fixes

- #532 Problems when cells have no style in read-only mode.

10.52.3 Minor changes

- PR 79 Make PlotArea editable in charts
- Use graphicalProperties as the alias for spPr

10.53 2.3.0-b2 (2015-09-04)

10.53.1 Bug fixes

- #488 Support hashValue attribute for sheetProtection
- #493 Warn that unsupported extensions will be dropped
- #494 Cells with exponentials causes a ValueError
- #497 Scatter charts are broken
- #499 Inconsistent conversion of localised datetimes
- #500 Adding images leads to unreadable files
- #509 Improve handling of sheet names

- #515 Non-ascii titles have bad repr
- #516 Ignore unassigned worksheets

10.53.2 Minor changes

- Worksheets are now iterable by row.
- Assign individual cell styles only if they are explicitly set.

10.54 2.3.0-b1 (2015-06-29)

10.54.1 Major changes

- Shift to using (row, column) indexing for cells. Cells will at some point *lose* coordinates.
- New implementation of conditional formatting. Databars now partially preserved.
- et_xmlfile is now a standalone library.
- Complete rewrite of chart package
- Include a tokenizer for fomulae to be able to adjust cell references in them. PR 63

10.54.2 Minor changes

- Read-only and write-only worksheets renamed.
- Write-only workbooks support charts and images.
- PR76 Prevent comment images from conflicting with VBA

10.54.3 Bug fixes

- #81 Support stacked bar charts
- #88 Charts break hyperlinks
- #97 Pie and combination charts
- #99 Quote worksheet names in chart references
- #150 Support additional chart options
- #172 Support surface charts
- #381 Preserve named styles

- #470 Adding more than 10 worksheets with the same name leads to duplicates sheet names and an invalid file

10.55 2.2.6 (unreleased)

10.55.1 Bug fixes

- #502 Unexpected keyword “mergeCell”
- #503 tostring missing in dump_worksheet
- #506 Non-ASCII formulae cannot be parsed
- #508 Cannot save files with coloured tabs
- Regex for ignoring named ranges is wrong (character class instead of prefix)

10.56 2.2.5 (2015-06-29)

10.56.1 Bug fixes

- #463 Unexpected keyword “mergeCell”
- #484 Unusual dimensions breaks read-only mode
- #485 Move return out of loop

10.57 2.2.4 (2015-06-17)

10.57.1 Bug fixes

- #464 Cannot use images when preserving macros
- #465 ws.cell() returns an empty cell on read-only workbooks
- #467 Cannot edit a file with ActiveX components
- #471 Sheet properties elements must be in order
- #475 Do not redefine class ___slots___ in subclasses
- #477 Write-only support for SheetProtection
- #478 Write-only support for DataValidation
- Improved regex when checking for datetime formats

10.58 2.2.3 (2015-05-26)

10.58.1 Bug fixes

- #451 fitToPage setting ignored
- #458 Trailing spaces lost when saving files.
- #459 setup.py install fails with Python 3
- #462 Vestigial rId conflicts when adding charts, images or comments
- #455 Enable Zip64 extensions for all versions of Python

10.59 2.2.2 (2015-04-28)

10.59.1 Bug fixes

- #447 Uppercase datetime number formats not recognised.
- #453 Borders broken in shared_styles.

10.60 2.2.1 (2015-03-31)

10.60.1 Minor changes

- PR54 Improved precision on times near midnight.
- PR55 Preserve macro buttons

10.60.2 Bug fixes

- #429 Workbook fails to load because header and footers cannot be parsed.
- #433 File-like object with encoding=None
- #434 SyntaxError when writing page breaks.
- #436 Read-only mode duplicates empty rows.
- #437 Cell.offset raises an exception
- #438 Cells with pivotButton and quotePrefix styles cannot be read
- #440 Error when customised versions of builtin formats
- #442 Exception raised when a fill element contains no children

- #444 Styles cannot be copied

10.61 2.2.0 (2015-03-11)

10.61.1 Bug fixes

- #415 Improved exception when passing in invalid in memory files.

10.62 2.2.0-b1 (2015-02-18)

10.62.1 Major changes

- Cell styles deprecated, use formatting objects (fonts, fills, borders, etc.) directly instead
- Charts will no longer try and calculate axes by default
- Support for template file types - PR21
- Moved ancillary functions and classes into utils package - single place of reference
- PR 34 Fully support page setup
- Removed SAX-based XML Generator. Special thanks to Elias Rabel for implementing xmlfile for xml.etree
- Preserve sheet view definitions in existing files (frozen panes, zoom, etc.)

10.62.2 Bug fixes

- #103 Set the zoom of a sheet
- #199 Hide gridlines
- #215 Preserve sheet view setings
- #262 Set the zoom of a sheet
- #392 Worksheet header not read
- #387 Cannot read files without styles.xml
- #410 Exception when preserving whitespace in strings
- #417 Cannot create print titles
- #420 Rename confusing constants
- #422 Preserve color index in a workbook if it differs from the standard

10.62.3 Minor changes

- Use a 2-way cache for column index lookups
- Clean up tests in cells
- PR 40 Support frozen panes and autofilter in write-only mode
- Use ws.calculate_dimension(force=True) in read-only mode for unsized worksheets

10.63 2.1.5 (2015-02-18)

10.63.1 Bug fixes

- #403 Cannot add comments in write-only mode
- #401 Creating cells in an empty row raises an exception
- #408 from_excel adjustment for Julian dates 1 < x < 60
- #409 refersTo is an optional attribute

10.63.2 Minor changes

- Allow cells to be appended to standard worksheets for code compatibility with write-only mode.

10.64 2.1.4 (2014-12-16)

10.64.1 Bug fixes

- #393 IterableWorksheet skips empty cells in rows
- #394 Date format is applied to all columns (while only first column contains dates)
- #395 temporary files not cleaned properly
- #396 Cannot write "=" in Excel file
- #398 Cannot write empty rows in write-only mode with LXML installed

10.64.2 Minor changes

- Add relation namespace to root element for compatibility with iWork
- Serialize comments relation in LXML-backend

10.65 2.1.3 (2014-12-09)

10.65.1 Minor changes

- PR 31 Correct tutorial
- PR 32 See #380
- PR 37 Bind worksheet to ColumnDimension objects

10.65.2 Bug fixes

- #379 ws.append() doesn’t set RowDimension Correctly
- #380 empty cells formatted as datetimes raise exceptions

10.66 2.1.2 (2014-10-23)

10.66.1 Minor changes

- PR 30 Fix regex for positive exponentials
- PR 28 Fix for #328

10.66.2 Bug fixes

- #120, #168 defined names with formulae raise exceptions, #292
- #328 ValueError when reading cells with hyperlinks
- #369 IndexError when reading definedNames
- #372 number_format not consistently applied from styles

10.67 2.1.1 (2014-10-08)

10.67.1 Minor changes

- PR 20 Support different workbook code names
- Allow auto_axis keyword for ScatterCharts

10.67.2 Bug fixes

- #332 Fills lost in ConditionalFormatting
- #360 Support value=" none" in attributes
- #363 Support undocumented value for textRotation
- #364 Preserve integers in read-only mode
- #366 Complete read support for DataValidation
- #367 Iterate over unsized worksheets

10.68 2.1.0 (2014-09-21)

10.68.1 Major changes

- "read_only" and "write_only" new flags for workbooks
- Support for reading and writing worksheet protection
- Support for reading hidden rows
- Cells now manage their styles directly
- ColumnDimension and RowDimension object manage their styles directly
- Use xmlfile for writing worksheets if available - around 3 times faster
- Datavalidation now part of the worksheet package

10.68.2 Minor changes

- Number formats are now just strings
- Strings can be used for RGB and aRGB colours for Fonts, Fills and Borders
- Create all style tags in a single pass
- Performance improvement when appending rows
- Cleaner conversion of Python to Excel values
- PR6 reserve formatting for empty rows
- standard worksheets can append from ranges and generators

10.68.3 Bug fixes

- #153 Cannot read visibility of sheets and rows
- #181 No content type for worksheets
- 241 Cannot read sheets with inline strings
- 322 1-indexing for merged cells
- 339 Correctly handle removal of cell protection
- 341 Cells with formulae do not round-trip
- 347 Read DataValidations
- 353 Support Defined Named Ranges to external workbooks

10.69 2.0.5 (2014-08-08)

10.69.1 Bug fixes

- #348 incorrect casting of boolean strings
- #349 roundtripping cells with formulae

10.70 2.0.4 (2014-06-25)

10.70.1 Minor changes

- Add a sample file illustrating colours

10.70.2 Bug fixes

- #331 DARKYELLOW was incorrect
- Correctly handle extend attribute for fonts

10.71 2.0.3 (2014-05-22)

10.71.1 Minor changes

- Updated docs

10.71.2 Bug fixes

- #319 Cannot load Workbooks with vertAlign styling for fonts

10.72 2.0.2 (2014-05-13)

10.73 2.0.1 (2014-05-13) brown bag

10.74 2.0.0 (2014-05-13) brown bag

10.74.1 Major changes

- This is last release that will support Python 3.2
- Cells are referenced with 1-indexing: A1 == cell(row=1, column=1)
- Use jdcal for more efficient and reliable conversion of datetimes
- Significant speed up when reading files
- Merged immutable styles
- Type inference is disabled by default
- RawCell renamed ReadOnlyCell
- ReadOnlyCell.internal_value and ReadOnlyCell.value now behave the same as Cell
- Provide no size information on unsized worksheets
- Lower memory footprint when reading files

10.74.2 Minor changes

- All tests converted to pytest
- Pyflakes used for static code analysis
- Sample code in the documentation is automatically run
- Support GradientFills
- BaseColWidth set

10.74.3 Pull requests

- #70 Add filterColumn, sortCondition support to AutoFilter
- #80 Reorder worksheets parts
- #82 Update API for conditional formatting
- #87 Add support for writing Protection styles, others
- #89 Better handling of content types when preserving macros

10.74.4 Bug fixes

- #46 ColumnDimension style error
- #86 reader.worksheet.fast_parse sets booleans to integers
- #98 Auto sizing column widths does not work
- #137 Workbooks with chartsheets
- #185 Invalid PageMargins
- #230 Using v in cells creates invalid files
- #243 - IndexError when loading workbook
- #263 - Forded conversion of line breaks
- #267 - Raise exceptions when passed invalid types
- #270 - Cannot open files which use non-standard sheet names or reference Ids
- #269 - Handling unsized worksheets in IterableWorksheet
- #270 - Handling Workbooks with non-standard references
- #275 - Handling auto filters where there are only custom filters
- #277 - Harmonise chart and cell coordinates
- #280- Explicit exception raising for invalid characters
- #286 - Optimized writer can not handle a datetime.time value
- #296 - Cell coordinates not consistent with documentation
- #300 - Missing column width causes load_workbook() exception
- #304 - Handling Workbooks with absolute paths for worksheets (from Sharepoint)

10.75 1.8.6 (2014-05-05)

10.75.1 Minor changes

Fixed typo for import Elementtree

10.75.2 Bugfixes

- #279 Incorrect path for comments files on Windows

10.76 1.8.5 (2014-03-25)

10.76.1 Minor changes

- The ‘=’ string is no longer interpreted as a formula
- When a client writes empty xml tags for cells (e.g. <c r=’A1’></c>), reader will not crash

10.77 1.8.4 (2014-02-25)

10.77.1 Bugfixes

- #260 better handling of undimensioned worksheets
- #268 non-ascii in formualae
- #282 correct implementation of register_namepsace for Python 2.6

10.78 1.8.3 (2014-02-09)

10.78.1 Major changes

Always parse using cElementTree

10.78.2 Minor changes

Slight improvements in memory use when parsing

- #256 - error when trying to read comments with optimised reader
- #260 - unsized worksheets

- #264 - only numeric cells can be dates

10.79 1.8.2 (2014-01-17)

- #247 - iterable worksheets open too many files
- #252 - improved handling of lxml
- #253 - better handling of unique sheetnames

10.80 1.8.1 (2014-01-14)

- #246

10.81 1.8.0 (2014-01-08)

10.81.1 Compatibility

Support for Python 2.5 dropped.

10.81.2 Major changes

- Support conditional formatting
- Support lxml as backend
- Support reading and writing comments
- pytest as testrunner now required
- Improvements in charts: new types, more reliable

10.81.3 Minor changes

- load_workbook now accepts data_only to allow extracting values only from formulae. Default is false.
- Images can now be anchored to cells
- Docs updated
- Provisional benchmarking
- Added convenience methods for accessing worksheets and cells by key

10.82 1.7.0 (2013-10-31)

10.82.1 Major changes

Drops support for Python < 2.5 and last version to support Python 2.5

10.82.2 Compatibility

Tests run on Python 2.5, 2.6, 2.7, 3.2, 3.3

10.82.3 Merged pull requests

- 27 Include more metadata
- 41 Able to read files with chart sheets
- 45 Configurable Worksheet classes
- 3 Correct serialisation of Decimal
- 36 Preserve VBA macros when reading files
- 44 Handle empty oddheader and oddFooter tags
- 43 Fixed issue that the reader never set the active sheet
- 33 Reader set value and type explicitly and TYPE_ERROR checking
- 22 added page breaks, fixed formula serialization
- 39 Fix Python 2.6 compatibility
- 47 Improvements in styling

10.82.4 Known bugfixes

- #109
- #165
- #209
- #112
- #166
- #109
- #223
- #124

- #157

10.82.5 Miscellaneous

Performance improvements in optimised writer

Docs updated

Python 模块索引

索引