openpyxl Documentation

Release 2.4.0

See AUTHORS

November 21, 2015

Contents

1 Introduction **3**
1.1 Support . . . 3
1.2 Sample code: . . . 3

2 User List **5**

3 How to Contribute Code **7**

4 Other ways to help **9**

5 Installation **11**

6 Working with a checkout **13**

7 Usage examples **15**
7.1 Tutorial . . . 15
7.2 Cookbook . . . 19
7.3 Charts . . . 22
7.4 Comments . . . 57
7.5 Read/write large files . . . 58
7.6 Working with styles . . . 59
7.7 Conditional Formatting . . . 63
7.8 Data Validation . . . 66
7.9 Parsing Formulas . . . 67

8 Information for Developers **71**
8.1 Development . . . 71
8.2 Testing on Windows . . . 74

9 API Documentation **77**
9.1 openpyxl package . . . 77

10 Indices and tables **217**

11 Release Notes **219**
11.1 2.4.0 (unreleased) . . . 219
11.2 2.3.2 (unreleased) . . . 219
11.3 2.3.1 (2015-11-20) . . . 219
11.4 2.3.0 (2015-10-20) . . . 220

11.5 2.3.0-b2 (2015-09-04) 220
11.6 2.3.0-b1 (2015-06-29) 221
11.7 2.2.6 (unreleased) 221
11.8 2.2.5 (2015-06-29) 222
11.9 2.2.4 (2015-06-17) 222
11.10 2.2.3 (2015-05-26) 222
11.11 2.2.2 (2015-04-28) 222
11.12 2.2.1 (2015-03-31) 223
11.13 2.2.0 (2015-03-11) 223
11.14 2.2.0-b1 (2015-02-18) 223
11.15 2.1.5 (2015-02-18) 224
11.16 2.1.4 (2014-12-16) 224
11.17 2.1.3 (2014-12-09) 225
11.18 2.1.2 (2014-10-23) 225
11.19 2.1.1 (2014-10-08) 225
11.20 2.1.0 (2014-09-21) 226
11.21 2.0.5 (2014-08-08) 227
11.22 2.0.4 (2014-06-25) 227
11.23 2.0.3 (2014-05-22) 227
11.24 2.0.2 (2014-05-13) 227
11.25 2.0.1 (2014-05-13) brown bag 227
11.26 2.0.0 (2014-05-13) brown bag 227
11.27 1.8.6 (2014-05-05) 229
11.28 1.8.5 (2014-03-25) 229
11.29 1.8.4 (2014-02-25) 229
11.30 1.8.3 (2014-02-09) 229
11.31 1.8.2 (2014-01-17) 230
11.32 1.8.1 (2014-01-14) 230
11.33 1.8.0 (2014-01-08) 230
11.34 1.7.0 (2013-10-31) 231

Python Module Index **233**

Author Eric Gazoni, Charlie Clark

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

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

Generated November 21, 2015

License MIT/Expat

Version 2.4.0

CHAPTER 1

Introduction

Openpyxl is a Python library for reading and writing Excel 2010 xlsx/xlsm/xltx/xltm files.

It was born from lack of existing library to read/write natively from Python the Office Open XML format.

All kudos to the PHPExcel team as openpyxl was initially based on PHPExcel.

1.1 Support

This is an open source project, maintained by volunteers in their spare time. This may well mean that particular features or functions that you would like are missing. But things don't have to stay that way. You can contribute the project development yourself or contract a developer for particular features.

Professional support for openpyxl is available from Clark Consulting & Research and Adimian. Donations to the project to support further development and maintenance are welcome.

Bug reports and feature requests should be submitted using the issue tracker. Please provide a full traceback of any error you see and if possible a sample file. If for reasons of confidentiality you are unable to make a file publicly available then contact of one the developers.

1.2 Sample code:

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

# grab the active worksheet
ws = wb.active

# Data can be assigned directly to cells
ws['A1'] = 42

# Rows can also be appended
ws.append([1, 2, 3])

# Python types will automatically be converted
import datetime
ws['A2'] = datetime.datetime.now()

# Save the file
wb.save("sample.xlsx")
```

CHAPTER 2

User List

Official user list can be found on http://groups.google.com/group/openpyxl-users

How to Contribute Code

Any help will be greatly appreciated, just follow those steps:

1. Please start a new fork (https://bitbucket.org/openpyxl/openpyxl/fork) for each independent feature, don't try to fix all problems at the same time, it's easier for those who will review and merge your changes ;-)

2. Hack hack hack

3. Don't forget to add unit tests for your changes! (YES, even if it's a one-liner, changes without tests will **not** be accepted.) There are plenty of examples in the source if you lack know-how or inspiration.

4. If you added a whole new feature, or just improved something, you can be proud of it, so add yourself to the AUTHORS file :-)

5. Let people know about the shiny thing you just implemented, update the docs!

6. When it's done, just issue a pull request (click on the large "pull request" button on *your* repository) and wait for your code to be reviewed, and, if you followed all theses steps, merged into the main repository.

For further information see Development

CHAPTER 4

Other ways to help

There are several ways to contribute, even if you can't code (or can't code well):

- triaging bugs on the bug tracker: closing bugs that have already been closed, are not relevant, cannot be reproduced, ...
- updating documentation in virtually every area: many large features have been added (mainly about charts and images at the moment) but without any documentation, it's pretty hard to do anything with it
- proposing compatibility fixes for different versions of Python: we support 2.6 to 3.5, so if it does not work on your environment, let us know :-)

Installation

Install openpyxl using pip. It is advisable to do this in a Python virtualenv without system packages:

```
$ pip install openpyxl
```

Note: There is support for the popular lxml library which will be used if it is installed. This is particular useful when creating large files.

Warning: To be able to include images (jpeg, png, bmp,...) into an openpyxl file, you will also need the "pillow" library that can be installed with:

```
$ pip install pillow
```

or browse https://pypi.python.org/pypi/Pillow/, pick the latest version and head to the bottom of the page for Windows binaries.

CHAPTER 6

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 hg+https://bitbucket.org/openpyxl/openpyxl@2.4#egg=openpyxl
```

CHAPTER 7

Usage examples

7.1 Tutorial

7.1.1 Manipulating a workbook in memory

Create a workbook

There is no need to create a file on the filesystem to get started with openpyxl. Just import the Workbook class and start using it

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

A workbook is always created with at least one worksheet. You can get it by using the `openpyxl.workbook.Workbook.active()` property

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

Note: This function uses the *_active_sheet_index* property, set to 0 by default. Unless you modify its value, you will always get the first worksheet by using this method.

You can also create new worksheets by using the `openpyxl.workbook.Workbook.create_sheet()` method

```
>>> ws1 = wb.create_sheet() # insert at the end (default)
# or
>>> ws2 = wb.create_sheet(0) # insert at first position
```

Sheets are given a name automatically when they are created. They are numbered in sequence (Sheet, Sheet1, Sheet2, ...). You can change this name at any time with the *title* property:

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

The background color of the tab holding this title is white by default. You can change this providing an RRGGBB color code to the sheet_properties.tabColor property:

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

Once you gave a worksheet a name, you can get it as a key of the workbook or using the `openpyxl.workbook.Workbook.get_sheet_by_name()` method

```
>>> ws3 = wb["New Title"]
>>> ws4 = wb.get_sheet_by_name("New Title")
>>> ws is ws3 is ws4
True
```

You can review the names of all worksheets of the workbook with the `openpyxl.workbook.Workbook.get_sheet_names()` method

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

You can loop through worksheets

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

Playing with data

Accessing one cell

Now we know how to access a worksheet, we can start modifying cells content.

Cells can be accessed directly as keys of the worksheet

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

This will return the cell at A4 or create one if it does not exist yet. Values can be directly assigned

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

There is also the `openpyxl.worksheet.Worksheet.cell()` method:

```
>>> c = ws.cell('A4')
```

You can also access a cell using row and column notation:

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

Note: When a worksheet is created in memory, it contains no *cells*. They are created when first accessed. This way we don't create objects that would never be accessed, thus reducing the memory footprint.

Warning: Because of this feature, scrolling through cells instead of accessing them directly will create them all in memory, even if you don't assign them a value.
Something like

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

will create 100x100 cells in memory, for nothing.
However, there is a way to clean all those unwanted cells, we'll see that later.

Accessing many cells

Ranges of cells can be accessed using slicing

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

You can also use the `openpyxl.worksheet.Worksheet.iter_rows()` method:

```
>>> tuple(ws.iter_rows('A1:C2'))
((<Cell Sheet1.A1>, <Cell Sheet1.B1>, <Cell Sheet1.C1>),
 (<Cell Sheet1.A2>, <Cell Sheet1.B2>, <Cell Sheet1.C2>))

>>> for row in ws.iter_rows('A1:C2'):
...        for cell in row:
...            print cell
<Cell Sheet1.A1>
<Cell Sheet1.B1>
<Cell Sheet1.C1>
<Cell Sheet1.A2>
<Cell Sheet1.B2>
<Cell Sheet1.C2>
```

If you need to iterate through all the rows or columns of a file, you can instead use the `openpyxl.worksheet.Worksheet.rows()` property:

```
>>> ws = wb.active
>>> ws['C9'] = 'hello world'
>>> 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>))
```

or the `openpyxl.worksheet.Worksheet.columns()` property:

```
>>> 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>))
```

Data storage

Once we have a `openpyxl.cell.Cell`, we can assign it a value:

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

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

You can also enable type and format inference:

```
>>> wb = Workbook(guess_types=True)
>>> c.value = '12%'
>>> print(c.value)
0.12

>>> import datetime
>>> d.value = datetime.datetime.now()
>>> print d.value
datetime.datetime(2010, 9, 10, 22, 25, 18)

>>> c.value = '31.50'
>>> print(c.value)
31.5
```

7.1.2 Saving to a file

The simplest and safest way to save a workbook is by using the `openpyxl.workbook.Workbook.save()` method of the `openpyxl.workbook.Workbook` object:

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

Warning: This operation will overwrite existing files without warning.

Note: Extension is not forced to be xlsx or xlsm, although you might have some trouble opening it directly with another application if you don't use an official extension.

As OOXML files are basically ZIP files, you can also end the filename with .zip and open it with your favourite ZIP archive manager.

You can specify the attribute as_template=True, to save the document as a template

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

or specify the attribute as_template=False (by default), to save the document template (or document) as document.

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

```
>>> wb = load_workbook('document.xlsx')
>>> wb.save('new_document.xlsx', as_template=False)
```

Warning: You should monitor the data attributes and document extensions for saving documents in the document templates and vice versa, otherwise the result table engine can not open the document.

Note: The following will fail:

```
>>> wb = load_workbook('document.xlsx')
>>> # Need to save with the extension *.xlsx
>>> wb.save('new_document.xlsm')
>>> # MS Excel can't open the document
>>>
>>> # or
>>>
>>> # Need specify attribute keep_vba=True
>>> wb = load_workbook('document.xlsm')
>>> wb.save('new_document.xlsm')
>>> # MS Excel can't open the document
>>>
>>> # or
>>>
>>> wb = load_workbook('document.xltm', keep_vba=True)
>>> # If us need template document, then we need specify extension as *.xltm.
>>> # If us need document, then we need specify attribute as_template=False.
>>> wb.save('new_document.xlsm', as_template=True)
>>> # MS Excel can't open the document
```

7.1.3 Loading from a file

The same way as writing, you can import `openpyxl.load_workbook()` to open an existing workbook:

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

This ends the tutorial for now, you can proceed to the Simple usage section

7.2 Cookbook

7.2.1 Simple usage

Write a workbook

```
>>> from openpyxl import Workbook
>>> from openpyxl.compat import range
>>> from openpyxl.cell 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")
>>> for row in range(10, 20):
...     for col in range(27, 54):
...         _ = ws3.cell(column=col, row=row, value="%s" % get_column_letter(col))
>>> print(ws3['AA10'].value)
AA
>>> wb.save(filename = dest_filename)
```

Write a workbook from *.xltx as *.xlsx

```
>>> from openpyxl import load_workbook
>>>
>>>
>>> wb = load_workbook('sample_book.xltx')
>>> ws = wb.active
>>> ws['D2'] = 42
>>>
>>> wb.save('sample_book.xlsx')
>>>
>>> # or you can overwrite the current document template
>>> # wb.save('sample_book.xltx')
```

Write a workbook from *.xltm as *.xlsm

```
>>> from openpyxl import load_workbook
>>>
>>>
>>> wb = load_workbook('sample_book.xltm', keep_vba=True)
>>> ws = wb.active
>>> ws['D2'] = 42
>>>
>>> wb.save('sample_book.xlsm')
>>>
>>> # or you can overwrite the current document template
>>> # wb.save('sample_book.xltm')
```

Read an existing workbook

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

Note: There are several flags that can be used in load_workbook.

- *guess_types* will enable or disable (default) type inference when reading cells.
- *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.

Warning: openpyxl does currently not read all possible items in an Excel file so images and charts will be lost from existing files if they are opened and saved with the same name.

Using number formats

```
>>> 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'
>>> # You can enable type inference on a case-by-case basis
>>> wb.guess_types = True
>>> # set percentage using a string followed by the percent sign
>>> ws['B1'] = '3.14%'
>>> wb.guess_types = False
>>> ws['B1'].value
0.031400000000000004
>>>
>>> ws['B1'].number_format
'0%'
```

Using formulae

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

Warning: NB you must use the English name for a function and function arguments *must* be separated by commas and not other punctuation such as semi-colons.

openpyxl never evaluates formula but it is possible to check the name of a formula:

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

If you're trying to use a formula that isn't known this could be because you're using a formula that was not included in the initial specification. Such formulae must be prefixed with *xlfn.* to work.

Merge / Unmerge cells

```
>>> from openpyxl.workbook import Workbook
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> ws.merge_cells('A1:B1')
>>> ws.unmerge_cells('A1:B1')
>>>
>>> # or
>>> ws.merge_cells(start_row=2,start_column=1,end_row=2,end_column=4)
>>> ws.unmerge_cells(start_row=2,start_column=1,end_row=2,end_column=4)
```

Inserting an image

```
>>> 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')
```

Fold columns (outline)

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

7.3 Charts

7.3.1 Charts

Warning: Openpyxl currently supports chart creation within a worksheet only. Charts in existing workbooks will be lost.

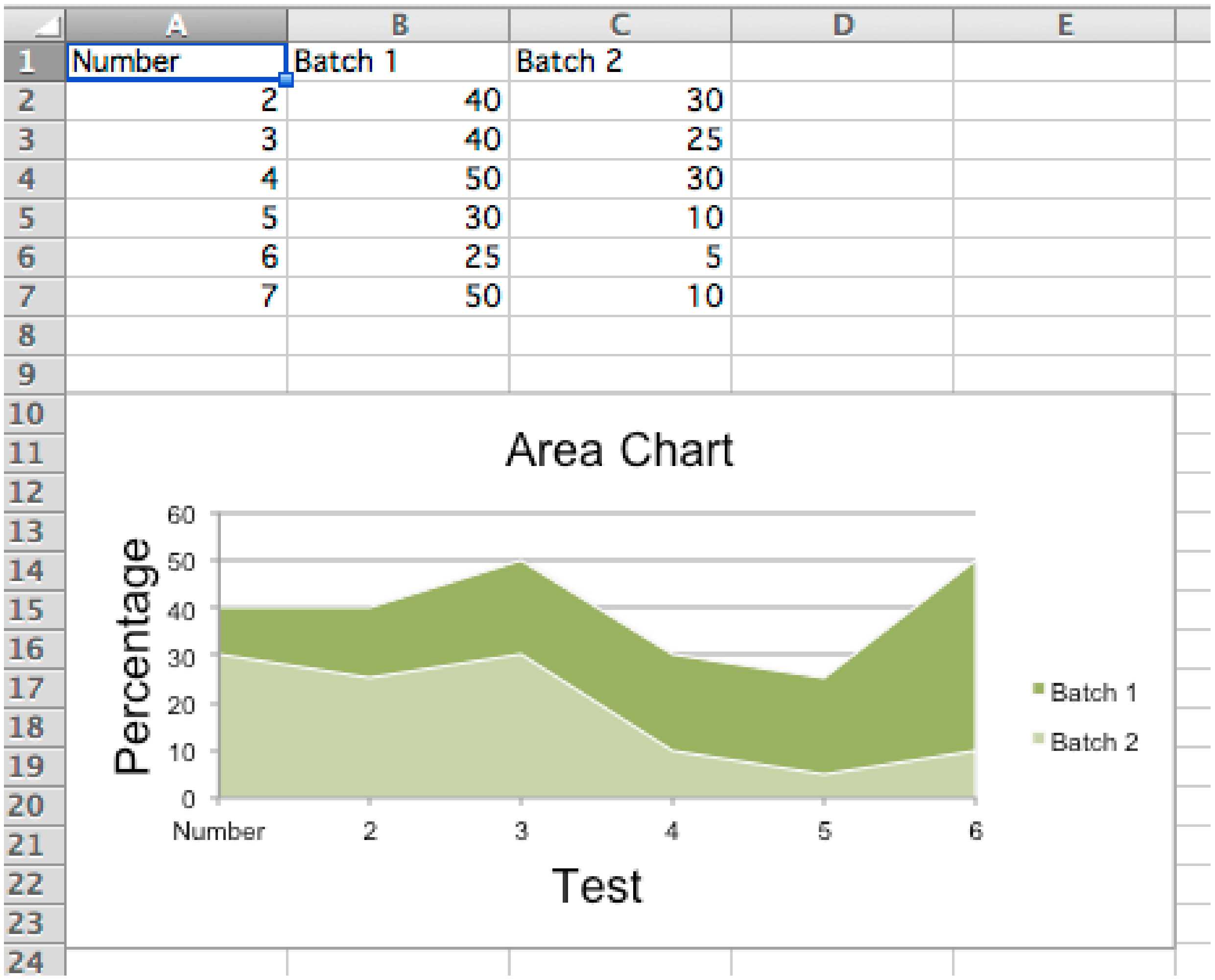
Chart types

The following charts are available:

Area Charts

2D Area Charts Area charts are similar to line charts with the addition that the area underneath the plotted line is filled. Different variants are available by setting the grouping to "standard", "stacked" or "percentStacked"; "standard" is the default.

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

wb = Workbook()
ws = wb.active

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")
```

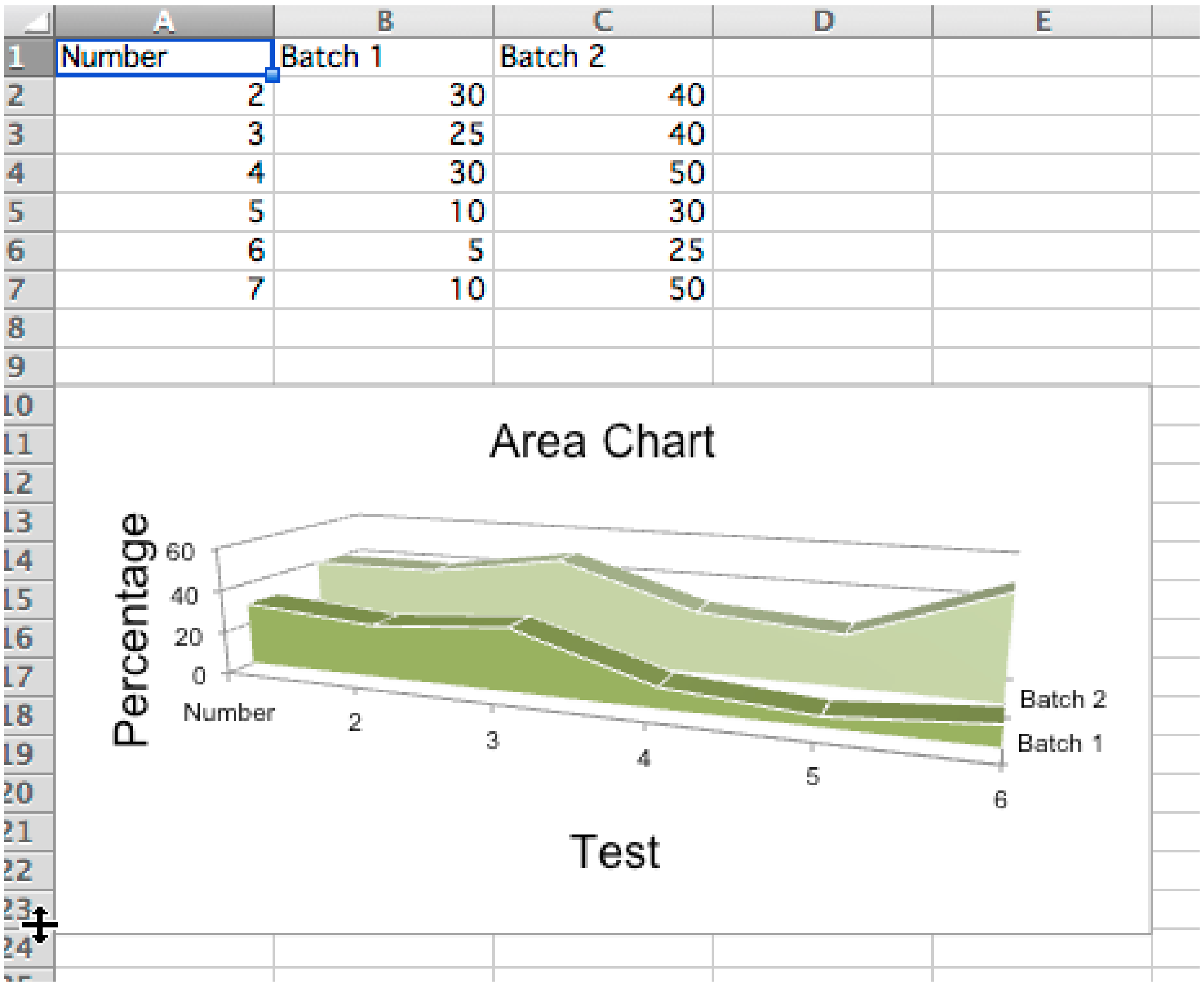


	A	B	C	D	E
1	Number	Batch 1	Batch 2		
2	2	40	30		
3	3	40	25		
4	4	50	30		
5	5	30	10		
6	6	25	5		
7	7	50	10		

3D Area Charts You can also create 3D area charts

```python
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],
    [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")
```

This produces a simple 3D area chart where third axis can be used to replace the legend:



	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		
8					
9					

Bar and Column Charts

In bar charts values are plotted as either horizontal bars or vertical columns.

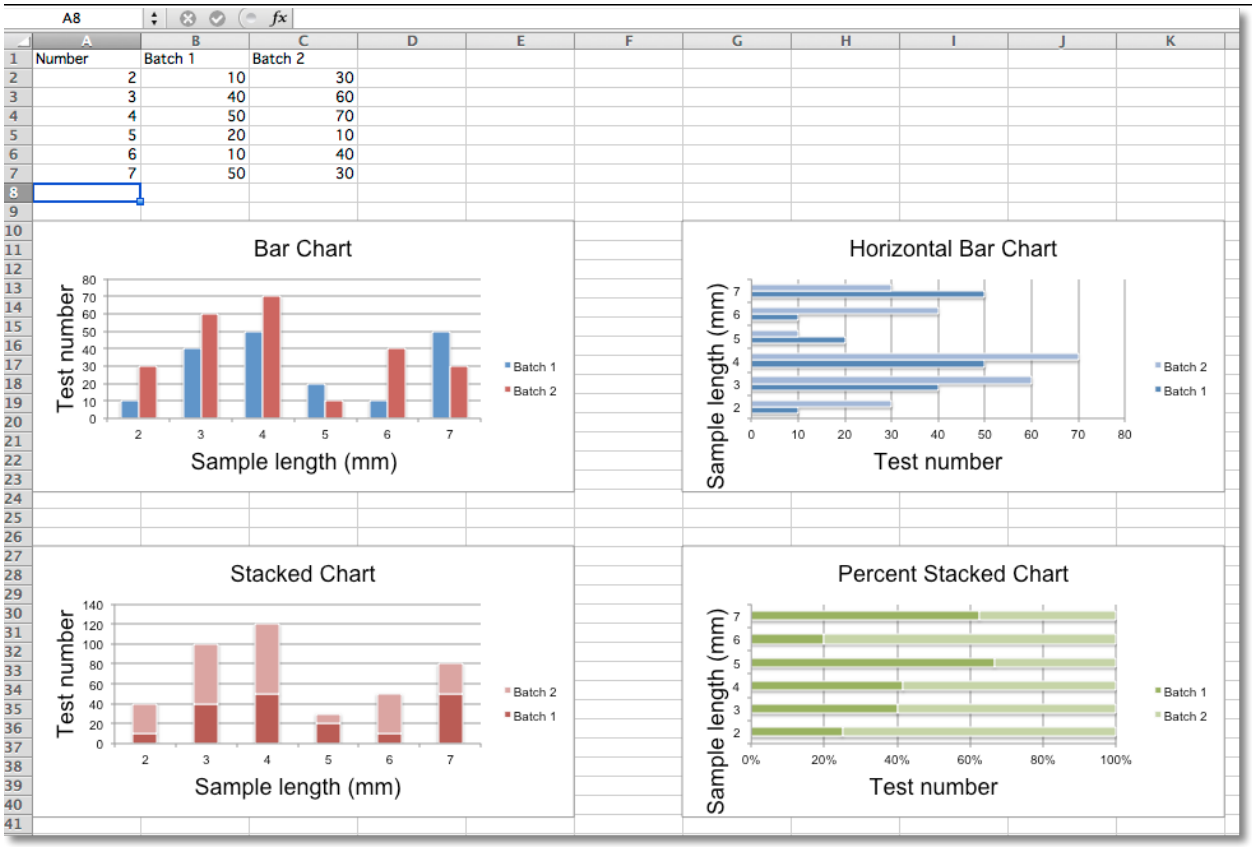
Vertical, Horizontal and Stacked Bar Charts

Note: The following settings affect the different chart types.

Switch between vertical and horizontal bar charts by setting *type* to *col* or *bar* respectively.

When using stacked charts the *overlap* needs to be set to 100.

If bars are horizontal, x and y axes are revesed.



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

```python
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),
]
```

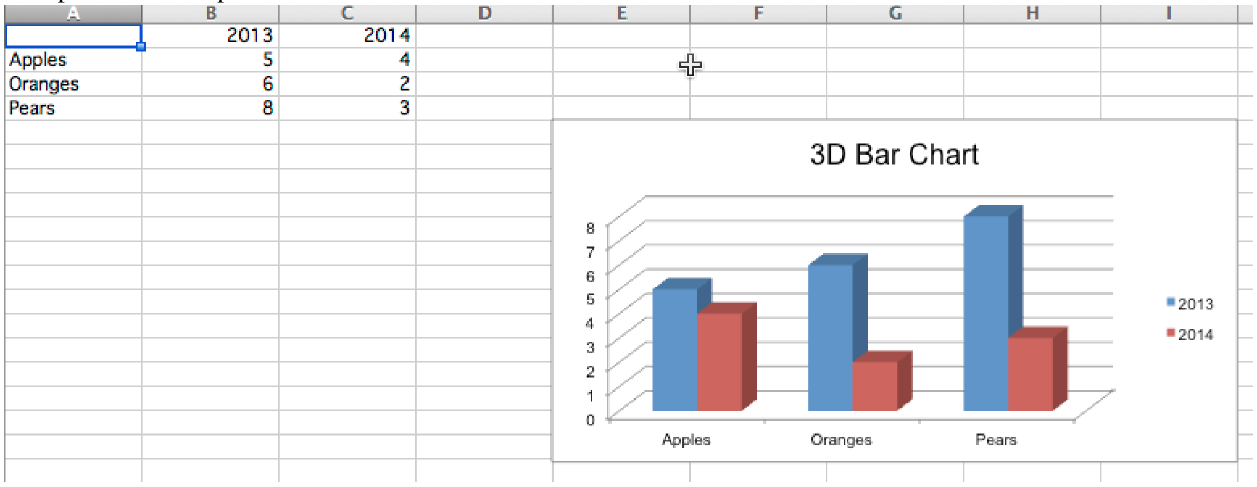
```
for row in rows:
    ws.append(row)


chart1 = BarChart()
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'

ws.add_chart(chart4, "G27")

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

This will produce four charts illustrating the various possibilities.

3D Bar Charts You can also create 3D bar charts

```
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")
```

This produces a simple 3D bar chart



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

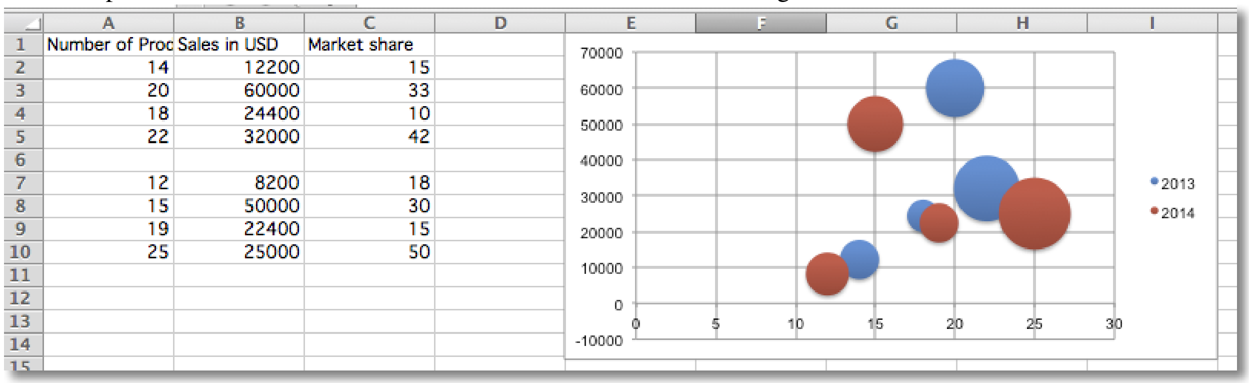
Bubble Charts

Bubble charts are similar to scatter charts but use a third dimension to determine the size of the bubbles. Charts can include multiple series.

```
"""
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)

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")
```

This will produce bubble chart with two series and should look something like this



	A	B	C
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

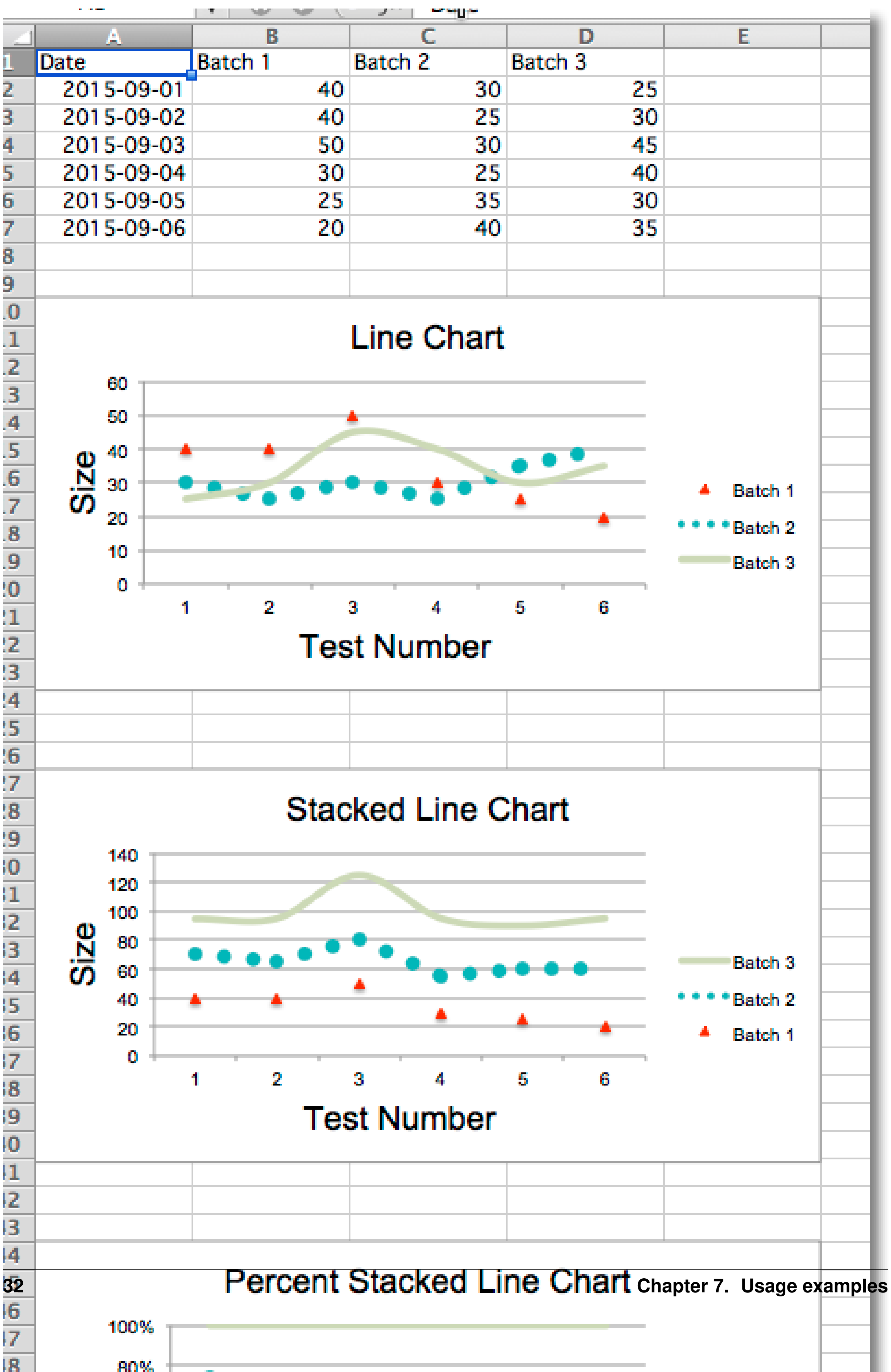
Line Charts

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.

```python
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

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
```

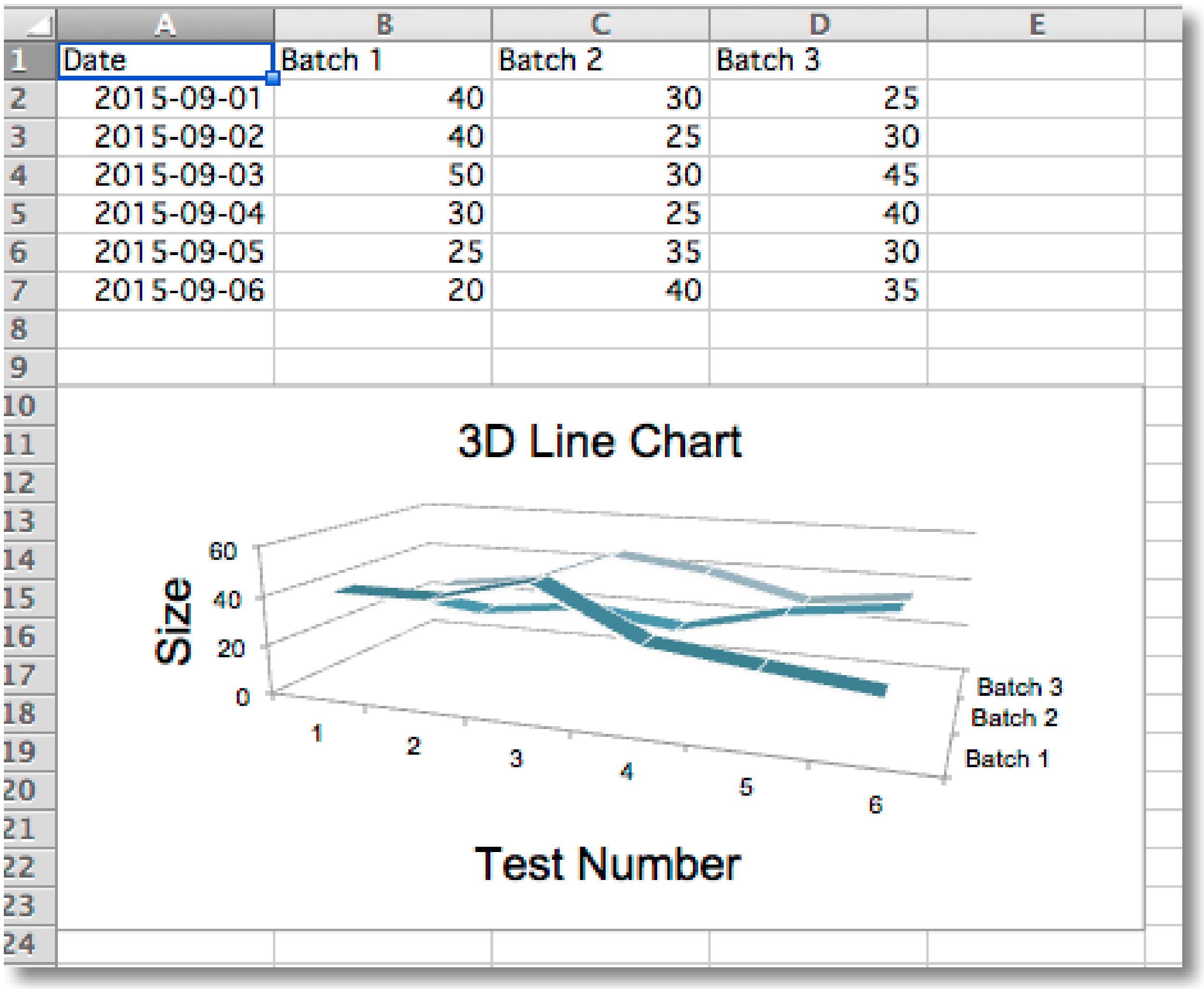
```
stacked = deepcopy(c1)
stacked.grouping = "stacked"
stacked.title = "Stacked Line Chart"
ws.add_chart(stacked, "A27")

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")

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.

```python
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")

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

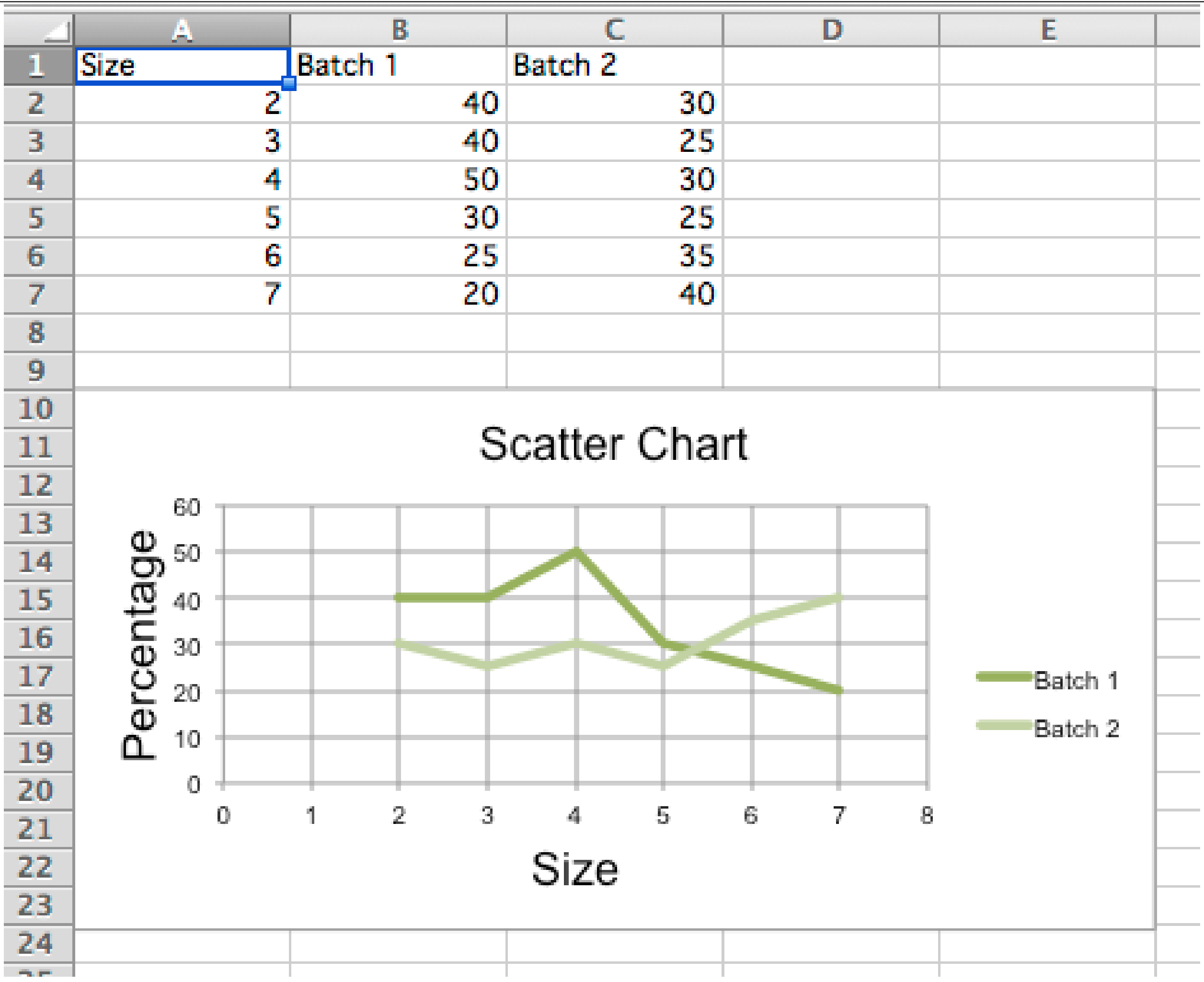


	A	B	C	D	E
1	Date	Batch 1	Batch 2	Batch 3	
2	2015-09-01	40	30	25	
3	2015-09-02	40	25	30	
4	2015-09-03	50	30	45	
5	2015-09-04	30	25	40	
6	2015-09-05	25	35	30	
7	2015-09-06	20	40	35	

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

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		

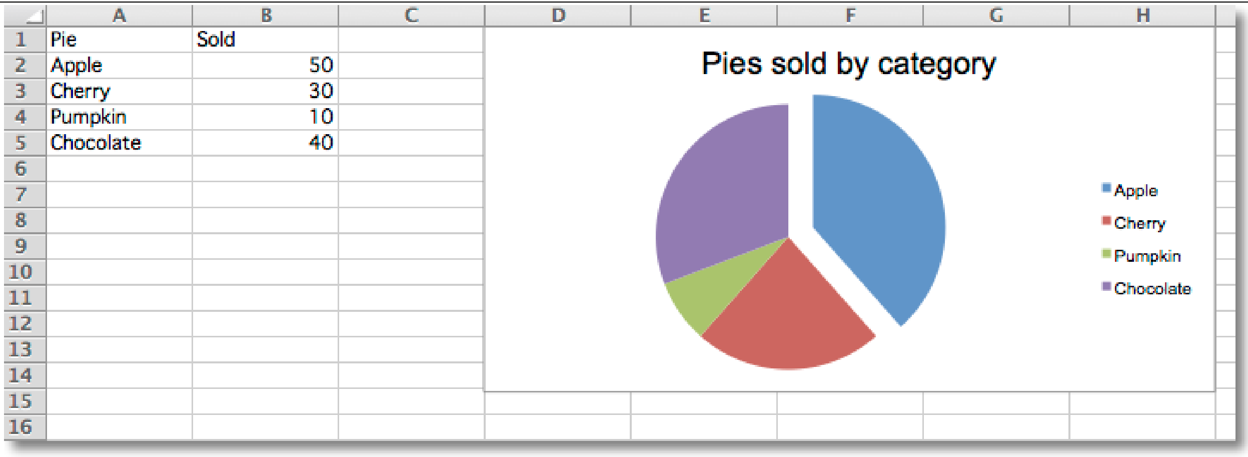
Note: The specification says that there are the following types of scatter charts: ‘line’, ‘lineMarker’, ‘marker’, ‘smooth’, ‘smoothMarker’. However, at least in Microsoft Excel, this is just a shortcut for other settings that otherwise no effect. For consistency with line charts, the style for each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

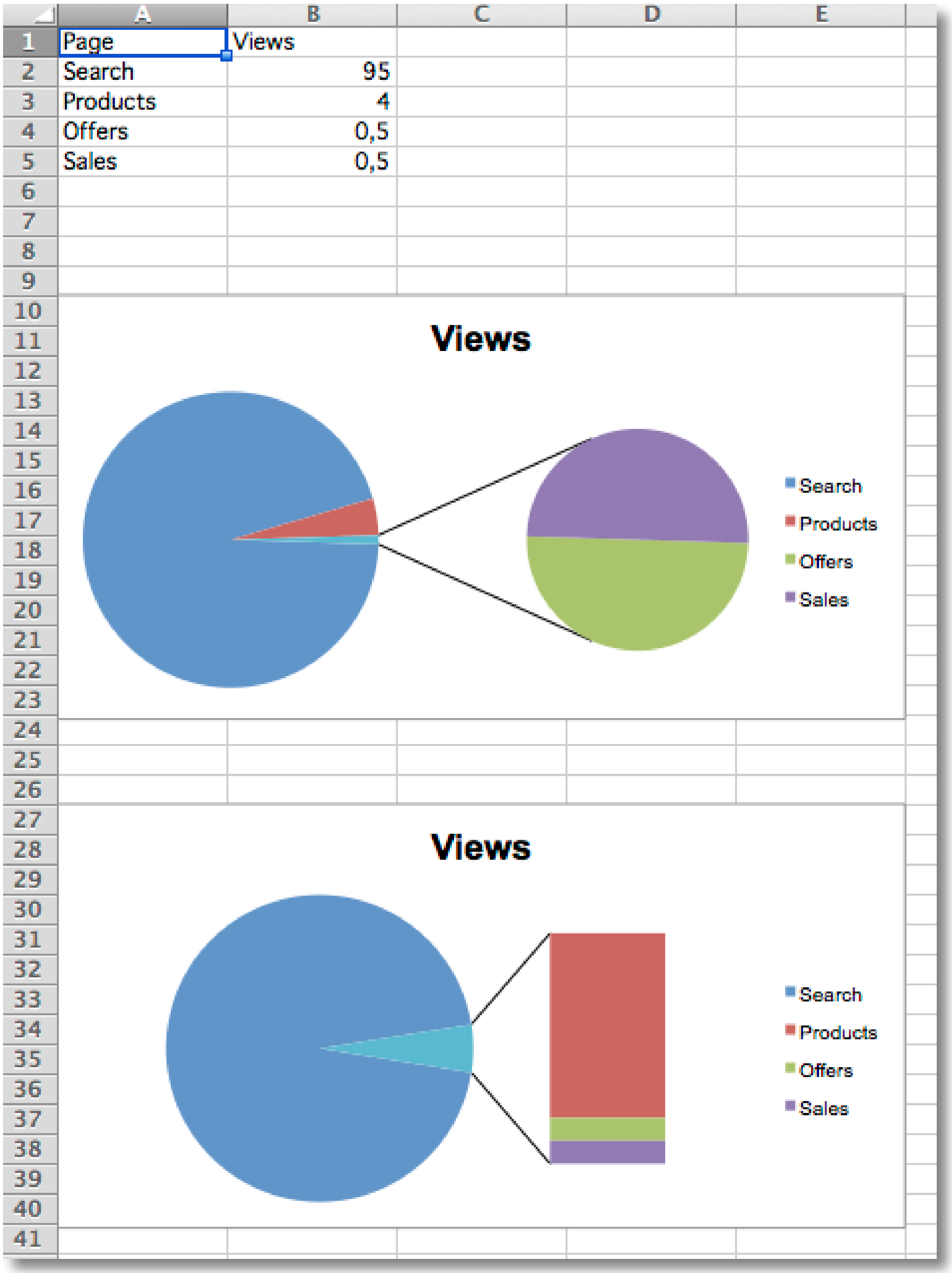
```
from openpyxl import Workbook

from openpyxl.chart import (
    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],
    ['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
1	Pie	Sold
2	Apple	50
3	Cherry	30
4	Pumpkin	10
5	Chocolate	40

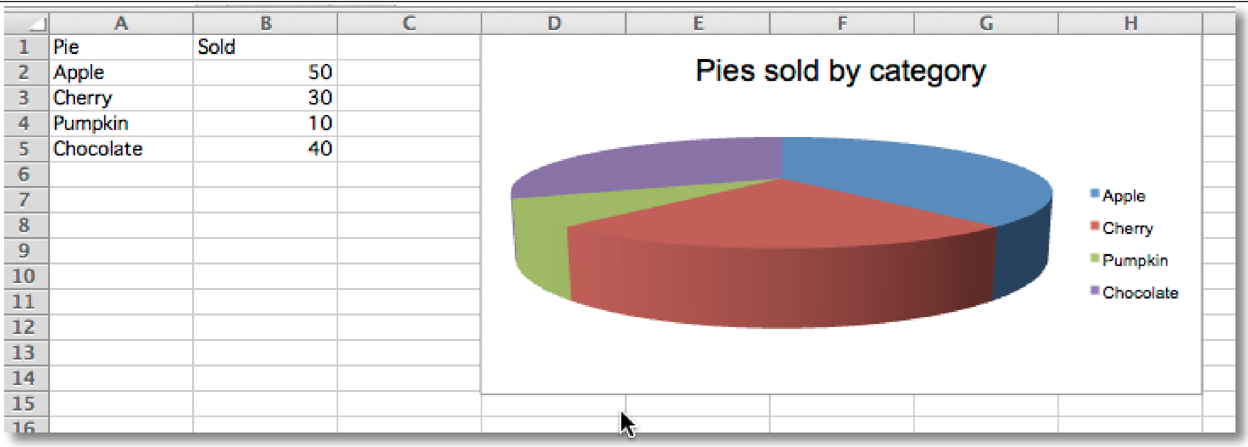
Projected Pie Charts Projected pie charts extract some slices from a pie chart and project them into a second pie or bar chart. This is useful when there are several smaller items in the data series. The chart can be split according percent, val(ue) or pos(ition). If nothing is set then the application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

```
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")
```

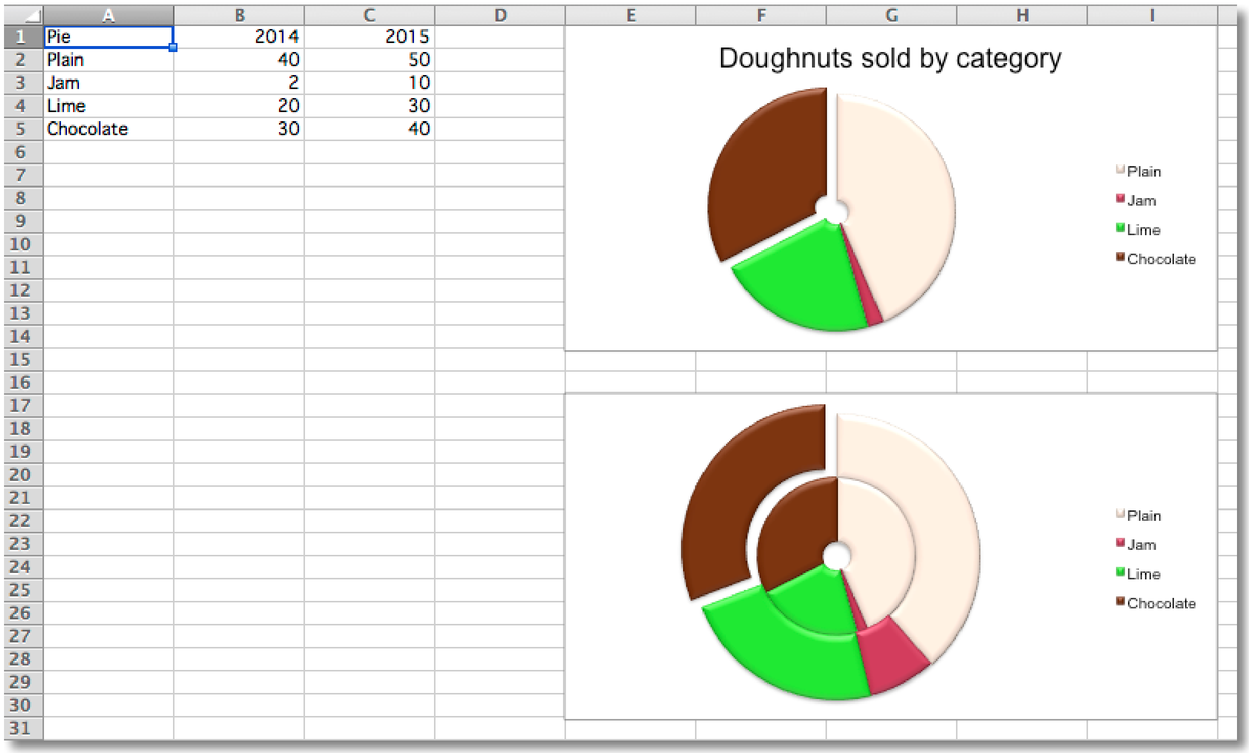


	A	B
1	Pie	Sold
2	Apple	50
3	Cherry	30
4	Pumpkin	10
5	Chocolate	40

Doughnut Charts

Doughnut charts are similar to pie charts except that they use a ring instead of a circle. They can also plot several series of data as concentric rings.

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

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

ws.add_chart(chart2, "E17")

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



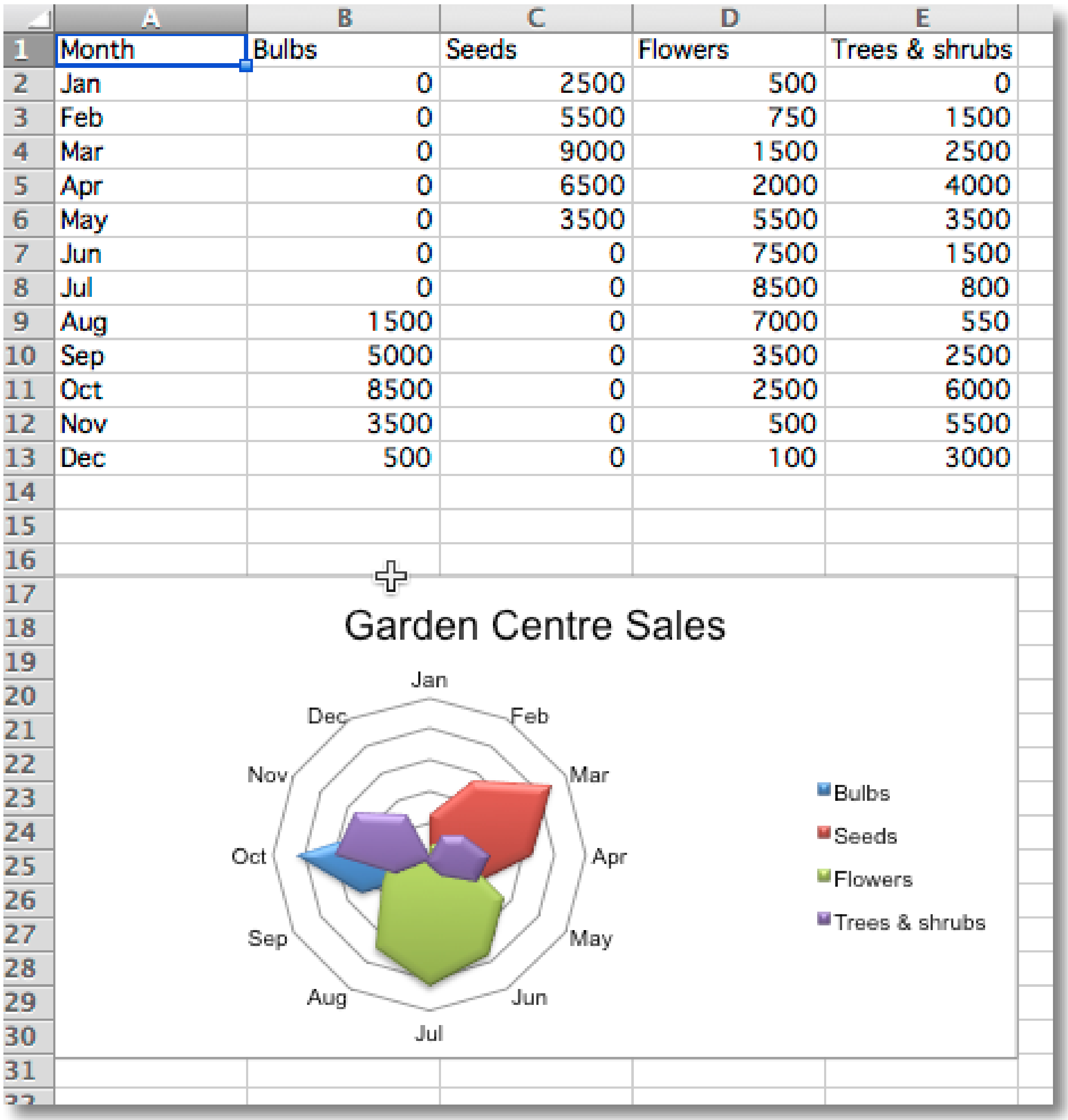
	A	B	C
1	Pie	2014	2015
2	Plain	40	50
3	Jam	2	10
4	Lime	20	30
5	Chocolate	30	40

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 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,
)

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")
```



	A	B	C	D	E
1	Month	Bulbs	Seeds	Flowers	Trees & shrubs
2	Jan	0	2500	500	0
3	Feb	0	5500	750	1500
4	Mar	0	9000	1500	2500
5	Apr	0	6500	2000	4000
6	May	0	3500	5500	3500
7	Jun	0	0	7500	1500
8	Jul	0	0	8500	800
9	Aug	1500	0	7000	550
10	Sep	5000	0	3500	2500
11	Oct	8500	0	2500	6000
12	Nov	3500	0	500	5500
13	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 as a high-low-close chart with the marker for each data point set to XZZ and upDownLines.

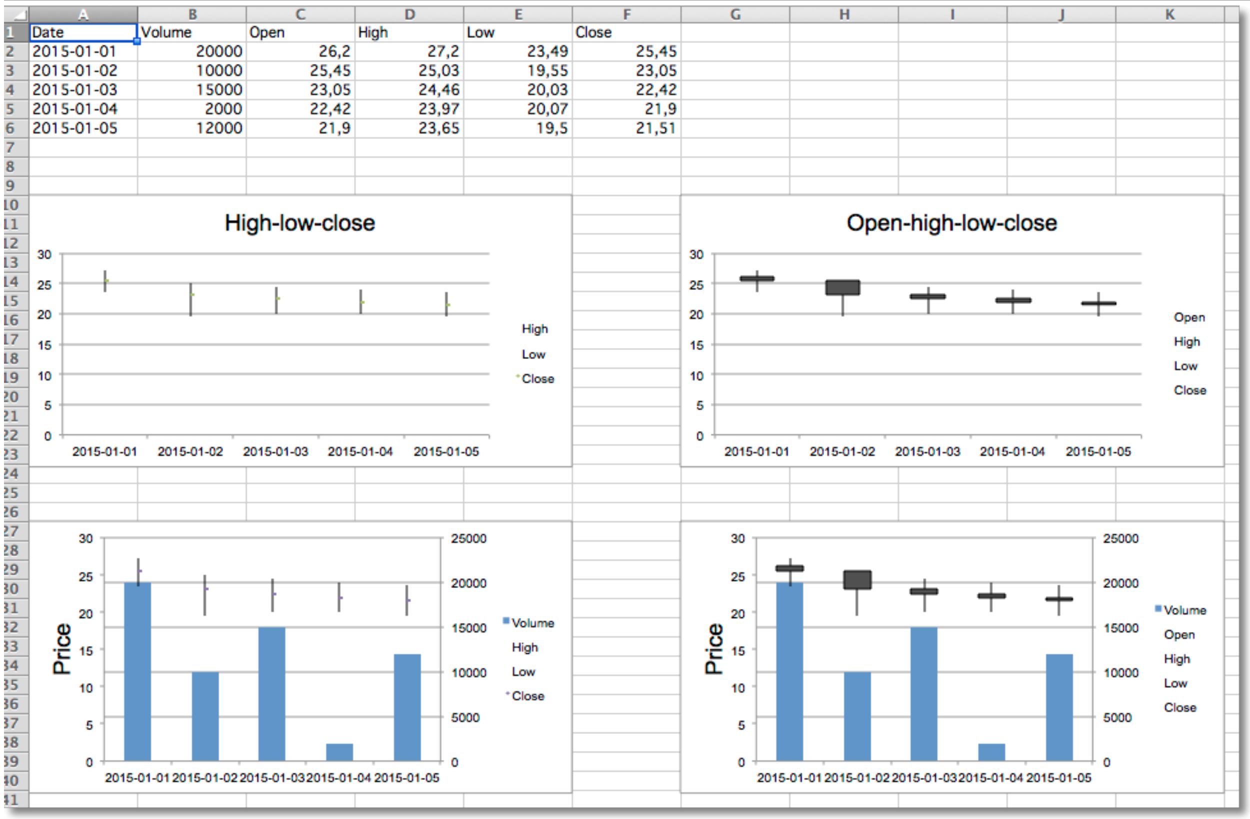
Volume can be added by combining the stock chart with a bar chart for the volume.

```
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)
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()
```

```python
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)

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")
```

Warning: 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

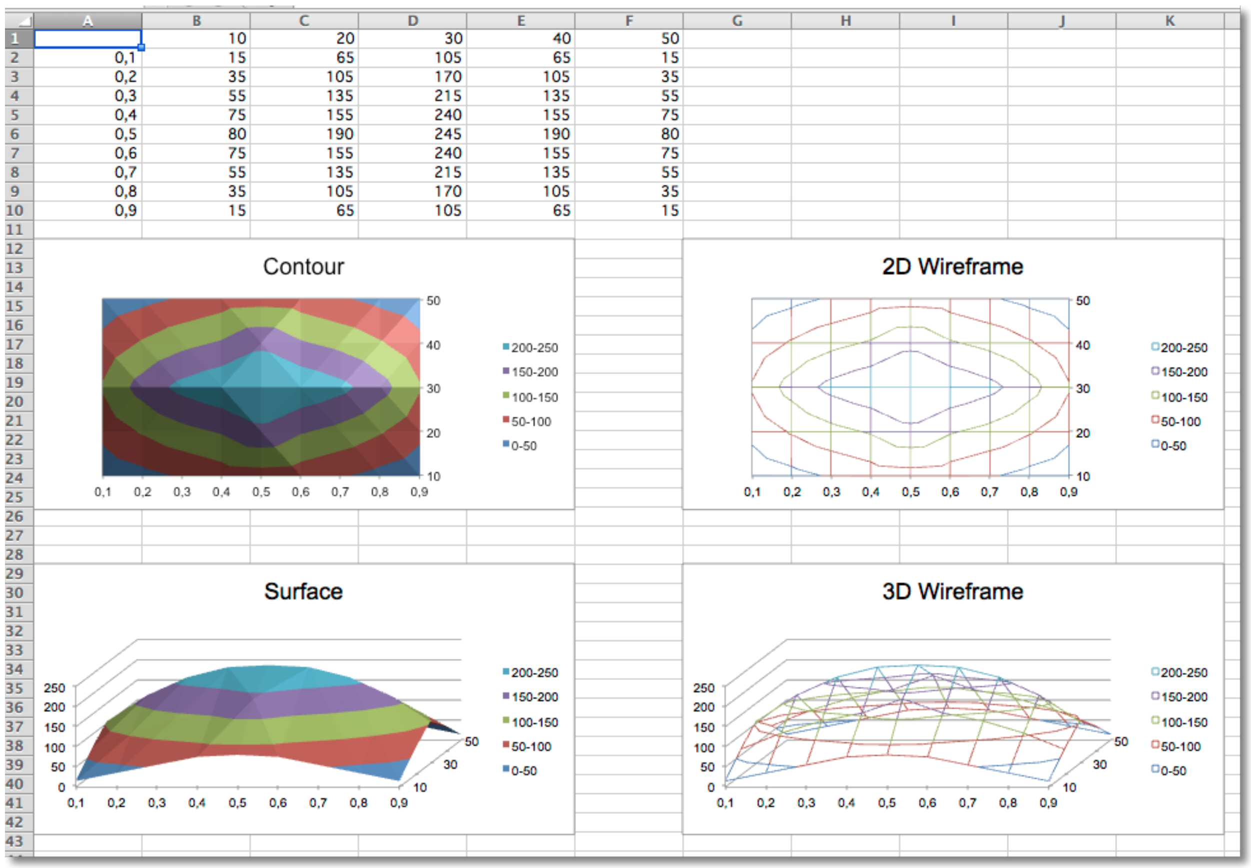
Surface charts

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,],
    [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")

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

Creating a chart

Charts are composed of at least one series of one or more data points. Series themselves are comprised of references to cell ranges.

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

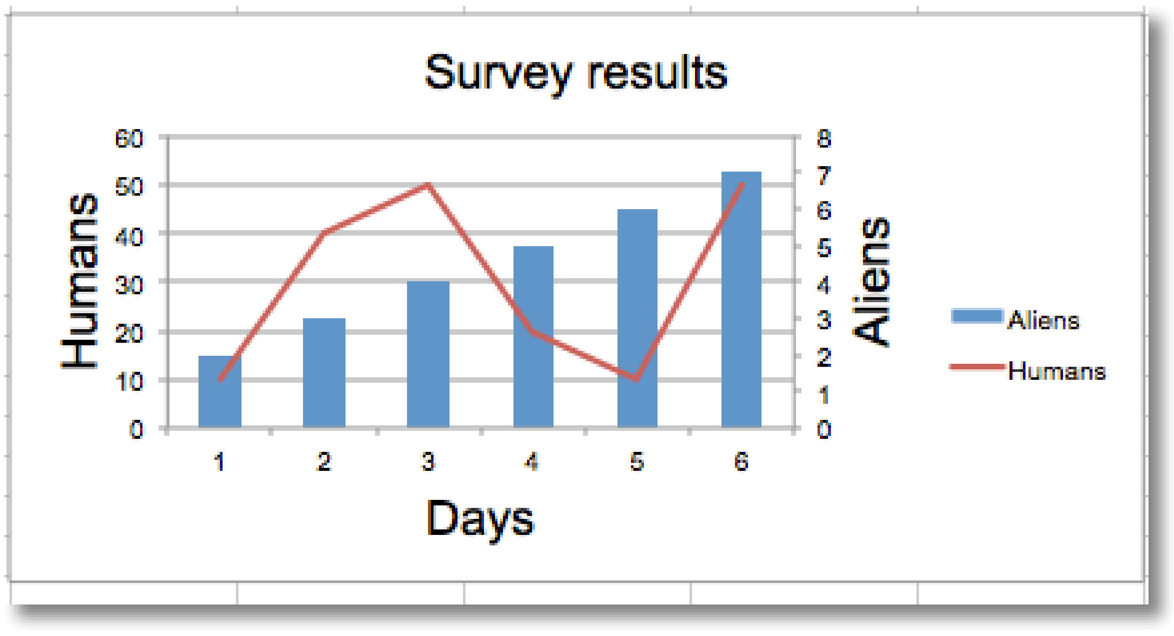
Working with axes

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,
)

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:



Change the chart layout

Changing the layout of plot area and legend

The layout of the chart within the canvas can be set by using the layout property an instance of a layout class.

Chart layout

Size and position The chart can be positioned within its container. `x` and `y` adjust position, `w` and `h` adjust the size . The units are proportions of the container. A chart cannot be positioned outside of its container and the width and height are the dominant constraints: if x + w > 1, then x = 1 - w.

x is the horizontal position from the left y is the vertical position the top h is the height of the chart relative to its container w is the width of the box

Mode In addition to the size and position the mode for the relevant attribute can also be set to either *factor* or *edge*. Factor is the default:

```
layout.xMode = edge
```

Target The layoutTarget can be set to `outer` or `inner`. The default is `outer`:

```
layout.layoutTarget = inner
```

Legend layout The position of the legend can be controlled either by setting its position: `r`, `l`, `t`, `b`, and `tr`, for right, left, top, bottom and top right respectively. The default is `r`.

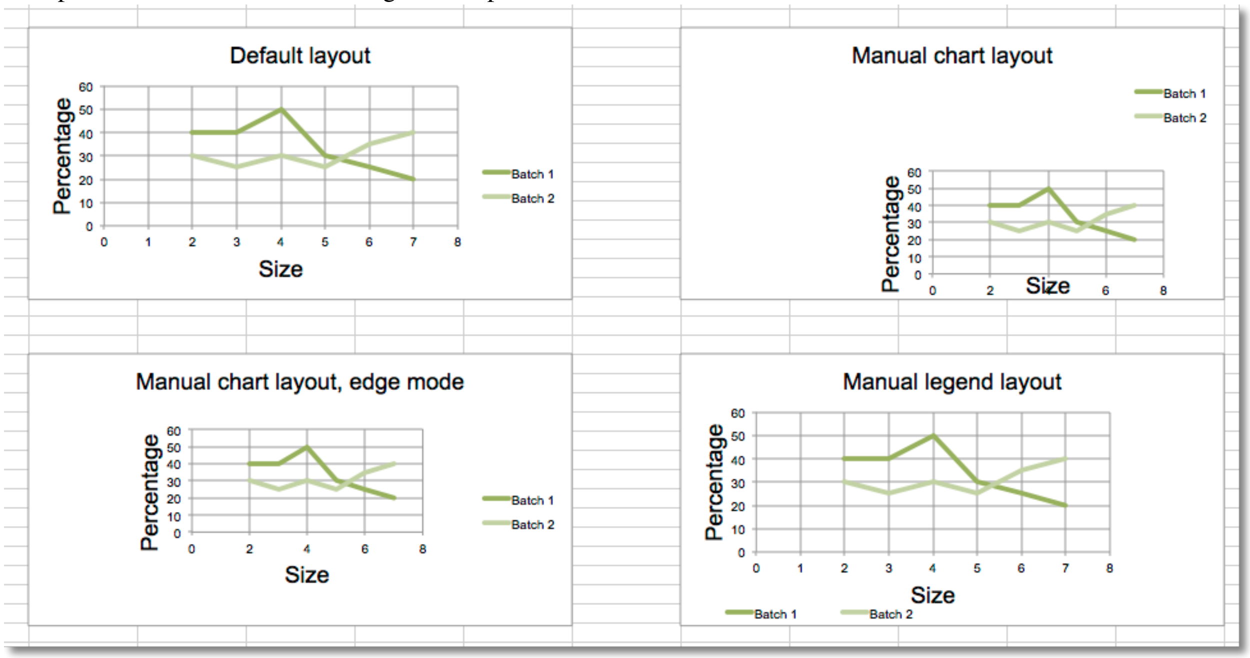
```
legend.position = 'tr'
```

or applying a manual layout:

```
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)
    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
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")
```

This produces four charts illustrating various possibilities:

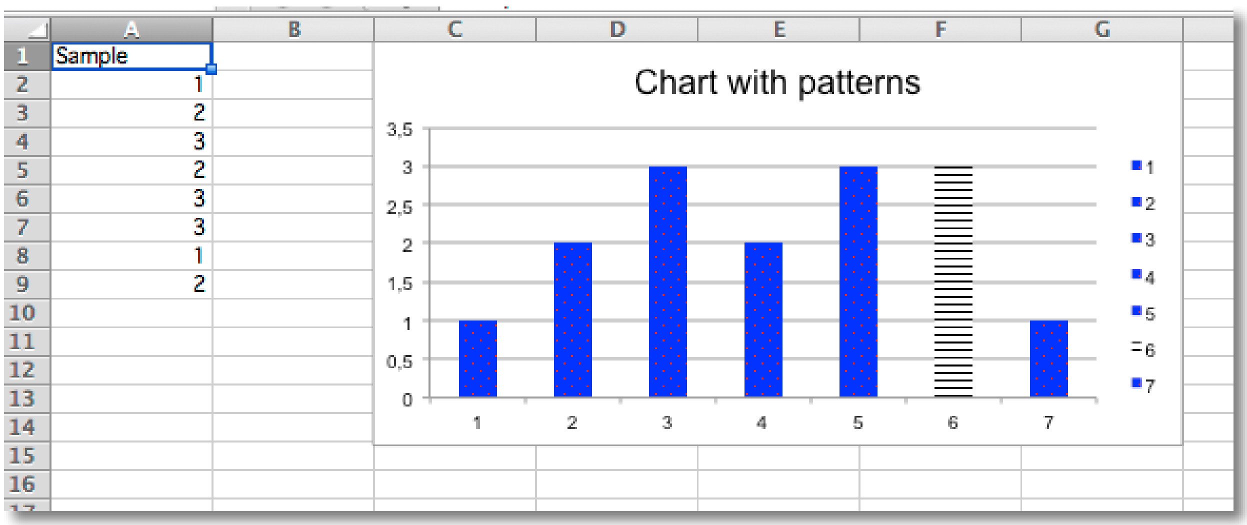


Styling charts

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")
series.dPt.append(pt)

ws.add_chart(c, "C1")

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



Advanced charts

Charts can be combined to create new charts:

Gauge Charts

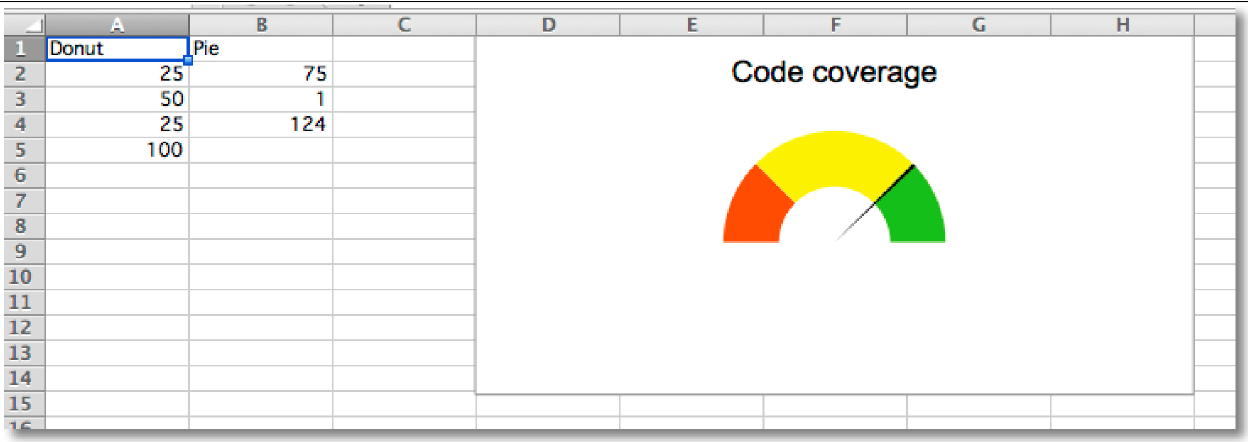
Gauge charts combine a pie chart and a doughnut chart to create a “gauge”. The first chart is a doughnut chart with four slices. The first three slices correspond to the colours of the gauge; the fourth slice, which is half of the doughnut, is made invisible.

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"],
    [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
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
1	Donut	Pie
2	25	75
3	50	1
4	25	124
5	100	

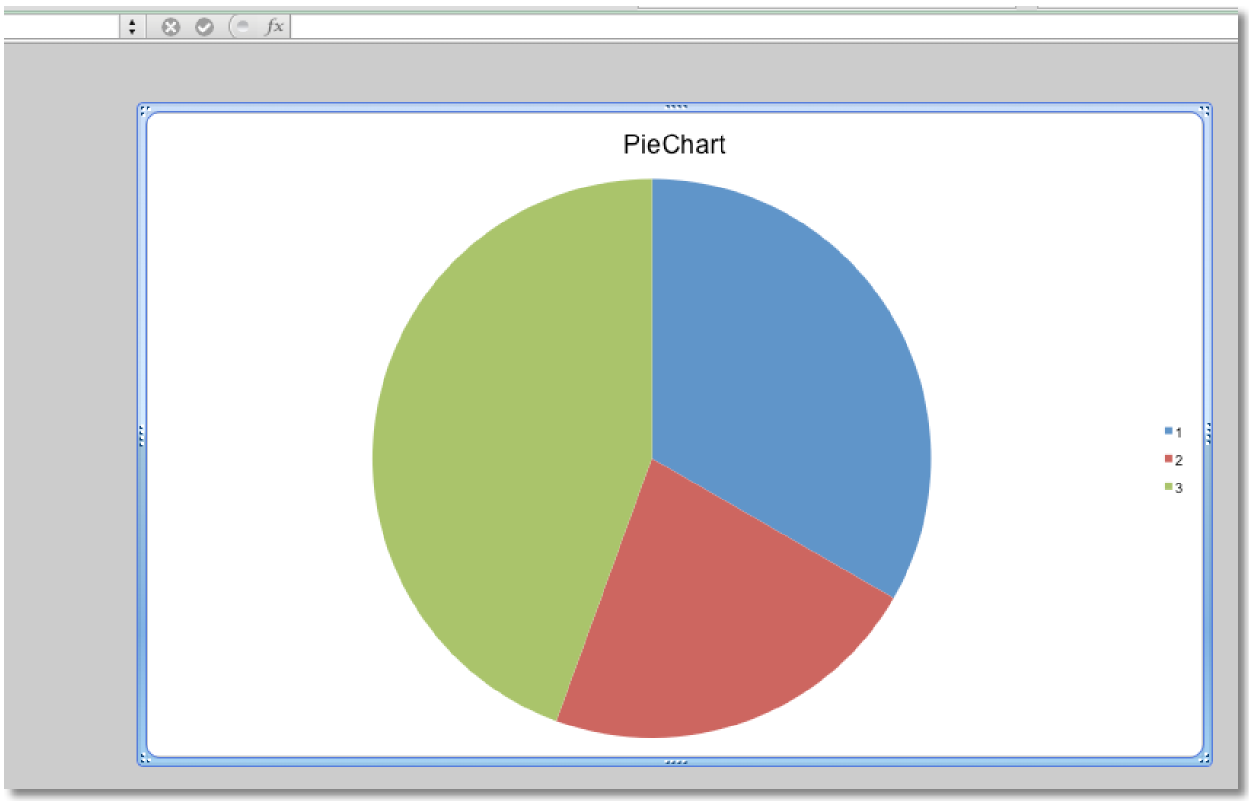
Using chartsheets

Charts can be added to special worksheets called chartsheets:

Chartsheets

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

```python
from openpyxl import Workbook

from openpyxl.chart import PieChart, Reference, Series

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

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



7.4 Comments

7.4.1 Comments

> **Warning:** Openpyxl currently supports the reading and writing of comment text only. Formatting information is lost. Comments are not currently supported if *use_iterators=True* is used.

Adding a comment to a cell

Comments have a text attribute and an author attribute, which must both be set

```
>>> 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'
```

You cannot assign the same Comment object to two different cells. Doing so raises an AttributeError.

```
>>> 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
Traceback (most recent call last):
AttributeError: Comment already assigned to A1 in worksheet Sheet. Cannot
assign a comment to more than one cell
```

Loading and saving comments

Comments present in a workbook when loaded are stored in the comment attribute of their respective cells automatically. Formatting information such as font size, bold and italics are lost, as are the original dimensions and position of the comment's container box.

Comments remaining in a workbook when it is saved are automatically saved to the workbook file.

7.5 Read/write large files

7.5.1 Read-only mode

Sometimes, you will need to open or write extremely large XLSX files, and the common routines in openpyxl won't be able to handle that load. Fortunately, there are two modes that enable you to read and write unlimited amounts of data with (near) constant memory consumption.

Introducing *openpyxl.worksheet.read_only.ReadOnlyWorksheet*:

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

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

Warning:
- *openpyxl.worksheet.read_only.ReadOnlyWorksheet* is read-only

Cells returned are not regular *openpyxl.cell.cell.Cell* but *openpyxl.cell.read_only.ReadOnlyCell*.

7.5.2 Write-only mode

Here again, the regular *openpyxl.worksheet.worksheet.Worksheet* has been replaced by a faster alternative, the *openpyxl.writer.write_only.WriteOnlyWorksheet*. When you want to dump large amounts of data, you might find optimized writer helpful.

```
>>> 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')
```

If you want to have cells with styles or comments then use a `openpyxl.writer.write_only.WriteOnlyCell()`

```
>>> from openpyxl import Workbook
>>> wb = Workbook(write_only=True)
>>> ws = wb.create_sheet()
>>> from openpyxl.writer.write_only import WriteOnlyCell
>>> from openpyxl.comments import Comment
>>> from openpyxl.styles import Style, Font
>>> cell = WriteOnlyCell(ws, value="hello world")
>>> cell.font = Font(name='Courrier', size=36)
>>> cell.comment = Comment(text="A comment", author="Author's Name")
```

This will append one new row with 3 cells, one text cell with custom font and font size, a float and an empty cell that will be discarded anyway.

Warning:

- Those worksheet only have an append() method, it's not possible to access independent cells directly (through cell() or range()). They are write-only.
- It is able to export unlimited amount of data (even more than Excel can handle actually), while keeping memory usage under 10Mb.
- A workbook using the optimized writer can only be saved once. After that, every attempt to save the workbook or append() to an existing worksheet will raise an `openpyxl.utils.exceptions.WorkbookAlreadySaved` exception.

7.6 Working with styles

7.6.1 Working with styles

Introduction

Styles are used to change the look of your data while displayed on screen. They are also used to determine the number format being used for a given cell or range of cells.

Styles can be applied to the following aspects:

- font to set font size, color, underlining, etc.
- fill to set a pattern or color gradient
- border to set borders on a cell
- cell alignment
- protection

The following are the default values

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

Styles are shared between objects and once they have been assigned they cannot be changed. This stops unwanted side-effects such as changing the style for lots of cells when instead of only one.

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

```
>>>
>>> a1.font = Font(color=colors.RED, italic=True) # the change only affects A1
```

Copying styles

Styles can also be copied

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

Basic Font Colors

Colors are usually RGB or aRGB hexvalues. The *colors* module contains some constants

```
>>> from openpyxl.styles import Font
>>> from openpyxl.styles.colors import RED
>>> font = Font(color=RED)
>>> font = Font(color="FFBB00")
```

There is also support for legacy indexed colors as well as themes and tints

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

Applying Styles

Styles are applied directly to cells

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

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")
```

Edit Page Setup

```
>>> 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
```

Edit Print Options

```
>>> from openpyxl.workbook import Workbook
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> ws.print_options.horizontalCentered = True
>>> ws.print_options.verticalCentered = True
```

Header / Footer

Headers and footers use their own formatting language. This is fully supported when writing them but, due to the complexity and the possibility of nesting, only partially when reading them.

```
>>> from openpyxl.workbook import Workbook
>>>
>>> wb = Workbook()
>>> ws = wb.worksheets[0]
>>>
>>> ws.header_footer.center_header.text = 'My Excel Page'
>>> ws.header_footer.center_header.font_size = 14
>>> ws.header_footer.center_header.font_name = "Tahoma,Bold"
>>> ws.header_footer.center_header.font_color = "CC3366"
```

# Or just >>> ws.header_footer.right_footer.text = 'My Right Footer'

Worksheet Additional Properties

These are advanced properties for particular behaviours, the most used ones are the "fitTopage" page setup property and the tabColor that define the background color of the worksheet tab.

Available properties for worksheet: "codeName", "enableFormatConditionsCalculation", "filterMode", "published", "syncHorizontal", "syncRef", "syncVertical", "transitionEvaluation", "transitionEntry", "tabColor". Available fields for page setup properties: "autoPageBreaks", "fitToPage". Available fields for outline properties: "applyStyles", "summaryBelow", "summaryRight", "showOutlineSymbols".

see http://msdn.microsoft.com/en-us/library/documentformat.openxml.spreadsheet.sheetproperties%28v=office.14%29.aspx_ for details.

..note:: By default, outline properties are intitialized so you can directly modify each of their 4 attributes, while page setup properties don't. If you want modify the latter, you should first initialize a PageSetupPr object with the required parameters. Once done, they can be directly modified by the routine later if needed.

```
>>> 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.PageSetupProperties = PageSetupProperties(fitToPage=True, autoPageBreaks=False)
>>> wsprops.outlinePr.summaryBelow = False
>>> wsprops.outlinePr.applyStyles = True
>>> wsprops.PageSetupProperties.autoPageBreaks = True
```

7.7 Conditional Formatting

7.7.1 Conditional Formatting

Excel supports three different types of conditional formatting: builtins, standard and custom. Builtins combine specific rules with predefined styles. Standard conditional formats combine specific rules with custom formatting. In additional it is possible to define custom formulae for applying custom formats using differential styles.

Note: The syntax for the different rules varies so much that it is not possible for openpyxl to know whether a rule makes sense or not.

The basic syntax for creating a formatting rule is:

```
>>> 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='FFEE1111', end_color=
>>> rule = Rule(type='cellIs', dxf=dxf, formula=["10"])
```

Because the signatures for some rules can be quite verbose there are also some convenience factories for creating them.

Builtin formats

The builtins conditional formats are:

- ColorScale
- IconSet
- DataBar

Builtin formats contain a sequence of formatting settings which combine a type with an integer for comparison. Possible types are: *'num'*, *'percent'*, *'max'*, *'min'*, *'formula'*, *'percentile'*.

ColorScale

You can have color scales with 2 or 3 colors. 2 color scales produce a gradient from one color to another; 3 color scales use an additional color for 2 gradients.

The full syntax for creating a ColorScale rule is:

```
>>> from openpyxl.formatting.rule import ColorScale, FormatObject
>>> from openpyxl.styles import Color
>>> first = FormatObject(type='min')
>>> last = FormatObject(type='max')
>>> # colors match the format objects:
>>> colors = [Color('FFAA0000'), Color('FF00AA00')]
>>> 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('FF00AA00'))
>>> 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)
```

There is a convenience function for creating ColorScale rules

```
>>> 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')
```

IconSet

Choose from the following set of icons: *'3Arrows'*, *'3ArrowsGray'*, *'3Flags'*, *'3TrafficLights1'*, *'3TrafficLights2'*, *'3Signs'*, *'3Symbols'*, *'3Symbols2'*, *'4Arrows'*, *'4ArrowsGray'*, *'4RedToBlack'*, *'4Rating'*, *'4TrafficLights'*, *'5Arrows'*, *'5ArrowsGray'*, *'5Rating'*, *'5Quarters'*

The full syntax for creating an IconSet rule is:

```
>>> 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
>>> # assign the icon set to a rule
>>> from openpyxl.formatting.rule import Rule
>>> rule = Rule(type='iconSet', iconSet=iconset)
```

There is a convenience function for creating IconSet rules:

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

DataBar

Currently, openpyxl supports the DataBars as defined in the original specification. Borders and directions were added in a later extension.

The full syntax for creating a DataBar rule is:

```
>>> from openpyxl.formatting.rule import DataBar, FormatObject
>>> first = FormatObject(type='min')
>>> second = FormatObject(type='max')
>>> data_bar = DataBar(cfvo=[first, second], color="FF638EC6", showValue=None, minLength=None, maxLen
>>> # assign the data bar to a rule
>>> from openpyxl.formatting.rule import Rule
>>> rule = Rule(type='dataBar', dataBar=data_bar)
```

There is a convenience function for creating DataBar rules:

```
>>> 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)
```

Standard conditional formats

The standard conditional formats are:

- Average
- Percent
- Unique or duplicate
- Value
- Rank

```
>>> from openpyxl import Workbook
>>> from openpyxl.styles import Color, PatternFill, Font, Border
>>> from openpyxl.formatting.rule import ColorScaleRule, CellIsRule, FormulaRule
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> # Create fill
>>> redFill = PatternFill(start_color='FFEE1111',
...                end_color='FFEE1111',
...                fill_type='solid')
>>>
>>> # Add a two-color scale
>>> # add2ColorScale(range_string, start_type, start_value, start_color, end_type, end_value, end_co
>>> # Takes colors in excel 'FFRRGGBB' style.
>>> ws.conditional_formatting.add('A1:A10',
...             ColorScaleRule(start_type='min', start_color='FFAA0000',
...                           end_type='max', end_color='FF00AA00')
...                           )
>>>
>>> # Add a three-color scale
>>> ws.conditional_formatting.add('B1:B10',
...                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')
...                              )
>>>
>>> # 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))
```

```
>>>
>>> # 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 styl
>>> myFont = Font()
>>> myBorder = Border()
>>> ws.conditional_formatting.add('E1:E10',
...             FormulaRule(formula=['E1=0'], font=myFont, border=myBorder, fill=redFill))
>>>
>>> wb.save("test.xlsx")
```

7.8 Data Validation

7.8.1 Validating cells

You can add data validation to a workbook but currently cannot read existing data validation.

Examples

```
>>> 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.ranges.append('B1:B1048576')
>>>
>>> # Write the sheet out.  If you now open the sheet in Excel, you'll find that
>>> # the cells have data-validation applied.
>>> wb.save("test.xlsx")
```

Other validation examples

Any whole number:

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

Any whole number above 100:

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

Any decimal number:

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

Any decimal number between 0 and 1:

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

Any date:

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

or time:

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

Any string at most 15 characters:

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

Custom rule:

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

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

7.9 Parsing Formulas

7.9.1 Parsing Formulas

openpyxl supports limited parsing of formulas embedded in cells. The *openpyxl.formula* package contains a *Tokenizer* class to break formulas into their consitutuent tokens. Usage is as follows:

```
>>> from openpyxl.formula import Tokenizer
>>> tok = Tokenizer("""=IF($A$1,"then True",MAX(DEFAULT_VAL,'Sheet 2'!B1))""")
>>> tok.parse()
>>> 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
```

As shown above, tokens have three attributes of interest:

- `.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`

CHAPTER 8

Information for Developers

8.1 Development

With the ongoing development of openpyxl, there is occasional information useful to assist developers.

8.1.1 What is suppoprted

The primary aim of openpyxl is to support reading and writing Microsoft Excel 2010 files. Where possible support for files generated by other libraries or programs is available but this is not guaranteed.

8.1.2 Supporting different Python versions

We have a small library of utility functions to support development for Python 2 and 3. This is openpyxl.compat for Python and openpyxl.xml for XML functions.

8.1.3 Coding style

Use PEP-8 except when implementing attributes for roundtripping but always use Python data conventions (boolean, None, etc.) Note exceptions in docstrings.

8.1.4 Getting the source

The source code is hosted on bitbucket.org. You can get it using a Mercurial client and the following URL.

```
$ hg clone https://bitbucket.org/openpyxl/openpyxl
$ hg up 2.4
$ virtualenv openpyxl
$ cd openpyxl
$ source bin/activate
$ pip install -U -r requirements.txt
$ python setup.py develop
```

8.1.5 Testing

Contributions 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.

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.

Please see Testing on Windows for additional information on setting up and testing on Windows.

8.1.6 Contributing

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

8.1.7 Branch naming convention

We use a "major.minor.patch" numbering system, ie. 2.4.0. 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.

8.1.8 Pull Requests

Pull requests should be submitted to the current, unreleased development branch. Eg. if the current release is 2.4.0, pull requests should be made to the 2.4 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
```

8.1.9 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.

8.2 Testing on Windows

Although openpyxl itself is pure Python and should run on any Python, we do use some libraries that require compiling for tests and documentation. The setup for testing on Windows is somewhat different.

8.2.1 Getting started

Once you have installed the versions of Python (2.6, 2.7, 3.3, 3.4) you should setup a development environment for testing so that you do not adversely affect the system install.

8.2.2 Setting up a development environment

First of all you should checkout a copy of the repository. Atlassian provides a nice GUI client SourceTree that allows you to do this with a single-click from the browser.

By default the repository will be installed under your user folder. eg. c:UsersYOURUSERopenpyxl

Switch to the branch you want to work on by double-clicking it. The default branch should never be used for development work.

Creating a virtual environment

You will need to manually install virtualenv. This is best done by first installing pip. open a command line and download the script "get_pip.py" to your preferred Python folder:

```
bitsadmin /transfer pip http://bootstrap.pypa.io/get-pip.py c:\python27\get-pip.py # change the path
```

Install pip (it needs to be at least pip 6.0):

```
python get_pip.py
```

Now you can install virtualenv:

```
Scripts\pip install virtualenv
Scripts\virtualenv c:\Users\YOURUSER\openpyxl
```

8.2.3 lxml

openpyxl needs *lxml* in order to run the tests. Unfortunately, automatic installation of lxml on Windows is tricky as pip defaults to try and compile it. This can be avoided by using pre-compiled versions of the library.

1. In the command line switch to your repository folder:

```
cd c:\Users\YOURUSER\openpyxl
```

2. Activate the virtualenv:

```
Scripts\activate
```

3. Install a development version of openpyxl:

```
python setup.py develop
```

4. Download all the relevant lxml Windows wheels

5. Move all these files to a folder called "downloads" in your openpyxl checkout
6. Install the project requirements:

```
    pip install --download downloads -r requirements.txt
    pip install --no-index --find-links downloads -r requirements.txt
```

To run tests for the virtualenv:

```
py.test -xrf openpyxl # the flag will stop testing at the first error
```

8.2.4 tox

We use *tox* to run the tests on different Python versions and configurations. Using it is as simple as:

```
set PIP_FIND_LINKS=downloads
tox openpyxl
```

CHAPTER 9

API Documentation

9.1 openpyxl package

9.1.1 Subpackages

openpyxl.cell package

Submodules

openpyxl.cell.cell module

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

Bases: `openpyxl.styles.styleable.StyleableObject`

Describes cell associated properties.

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

`ERROR_CODES` = **(‘#NULL!’, ‘#DIV/0!’, ‘#VALUE!’, ‘#REF!’, ‘#NAME?’, ‘#NUM!’, ‘#N/A’)**

`TYPE_BOOL` = **‘b’**

`TYPE_ERROR` = **‘e’**

`TYPE_FORMULA` = **‘f’**

`TYPE_FORMULA_CACHE_STRING` = **‘str’**

`TYPE_INLINE` = **‘inlineStr’**

`TYPE_NULL` = **‘n’**

`TYPE_NUMERIC` = **‘n’**

`TYPE_STRING` = **‘s’**

`VALID_TYPES` = **(‘s’, ‘f’, ‘n’, ‘b’, ‘n’, ‘inlineStr’, ‘e’, ‘str’)**

`anchor`

returns the expected position of a cell in pixels from the top-left of the sheet. For example, A1 anchor should be (0,0).

Return type tuple(int, int)

`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

column

comment
: Returns the comment associated with this cell

    Return type `openpyxl.comments.Comment`

coordinate

data_type

encoding

guess_types

hyperlink
: Return the hyperlink target or an empty string

internal_value
: Always returns the value for excel.

is_date
: Whether the value is formatted as a date

    Return type bool

offset(*row=0*, *column=0*)
: Returns a cell location relative to this cell.

    Parameters

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

    Return type `openpyxl.cell.Cell`

parent

row

set_explicit_value(*value=None*, *data_type='s'*)
: Coerce values according to their explicit type

value
: Get or set the value held in the cell. ':rtype: depends on the value (string, float, int or ' '`datetime.datetime`)'

openpyxl.cell.interface module

class `openpyxl.cell.interface.`**AbstractCell**(*value=None*)
: Bases: `abc.ABC`

base_date

comment

coordinate

encoding

guess_types

internal_value

is_date

number_format

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

style

value

openpyxl.cell.read_only module

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

Bases: object

alignment

base_date

border

column

coordinate

data_type

fill

font

internal_value

is_date

number_format

parent

protection

row

shared_strings

style

style_array

value

openpyxl.cell.text module

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*)

Bases: *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 {'minor', 'major'}

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 {'single', 'doubleAccounting', 'double', 'singleAccounting'}

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

class `openpyxl.cell.text.`**`PhoneticProperties`**(*fontId=None*, *type=None*, *alignment=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

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

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

class `openpyxl.cell.text.`**`PhoneticText`**(*sb=None*, *eb=None*, *t=None*)
Bases: *`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 Values must be of type <class 'str'>

class `openpyxl.cell.text.`**`RichText`** (*rPr=None*, *t=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

t
Values must be of type <class 'str'>

tagname = 'RElt'

class `openpyxl.cell.text.`**`Text`** (*t=None*, *r=()*, *rPh=()*, *phoneticPr=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

content
Text stripped of all formatting

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

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*)
Bases: `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 {'stacked', 'standard', '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*)
Bases: *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 {'stacked', 'standard', '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*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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=1*, *majorTimeUnit=None*, *minorUnit=None*, *minorTimeUnit=None*, *extLst=None*, ***kw*)
Bases: 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 {'b', 'r', 'l', 't'}

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

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

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

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 {'in', 'out', 'cross'}

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

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 {'in', 'out', 'cross'}

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

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*)

Bases: `openpyxl.descriptors.serialisable.Serialisable`

`layout`
Values must be of type <class 'openpyxl.chart.layout.Layout'>

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

`tagname` = 'dispUnitsLbl'

`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*)

Bases: `openpyxl.descriptors.serialisable.Serialisable`

`builtInUnit`
Value must be one of {'tenMillions', 'thousands', 'hundredMillions', 'trillions', 'hundreds', 'billions', 'hundredThousands', 'tenThousands', 'millions'}

`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*)

Bases: `openpyxl.chart.axis._BaseAxis`

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

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

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

`crossBetween`
Value must be one of {'midCat', 'between'}

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

`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'>

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

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

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 {'in', 'out', 'cross'}

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*)
Bases: *`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 {'maxMin', 'minMax'}

tagname = 'scaling'

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

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

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

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

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

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 {'in', 'out', 'cross'}

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

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

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*)
Bases: 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 {'b', 'r', 'l', 't'}

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

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

`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 {'ctr', 'r', 'l'}

`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 {'in', 'out', 'cross'}

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

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

`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*, *axId=None*, *extLst=None*, ***kw*)
Bases: `openpyxl.chart.bar_chart._BarChartBase`

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

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 {'stacked', 'standard', 'clustered', '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*, *axId=None*, *extLst=None*, ***kw*)
Bases: `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 {'col', 'bar'}

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 {'stacked', 'standard', 'clustered', '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 {'coneToMax', 'pyramid', 'pyramidToMax', 'cylinder', 'box', 'cone'}

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*, *axId=None*, *extLst=None*)

Bases: `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'>

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 {'area', 'w'}

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=None*, *view3D=None*, *floor=None*, *sideWall=None*, *backWall=None*, *plotArea=None*, *legend=None*, *plotVisOnly=None*, *dispBlanksAs='gap'*, *showDLblsOverMax=None*, *extLst=None*)
Bases: *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', 'span', 'gap'}

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

plotArea
Values must be of type <class 'openpyxl.chart.chartspace.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*)

Bases: *`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'>

`lang`
Values must be of type <class 'str'>

`pivotSource`
Values must be of type <class 'openpyxl.chart.chartspace.PivotSource'>

`printSettings`
Values must be of type <class 'openpyxl.chart.chartspace.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 'int'>

`tagname` **= 'chartSpace'**

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

`userShapes`
Values must be of type <class 'openpyxl.chart.chartspace.RelId'>

class `openpyxl.chart.chartspace.`**`DataTable`**(*showHorzBorder=None*, *showVertBorder=None*, *showOutline=None*, *showKeys=None*, *spPr=None*, *txPr=None*, *extLst=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

`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.chartspace.**ExternalData**(*autoUpdate=None*, *id=None*)
Bases: *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.**PivotFormat**(*idx=0*, *spPr=None*, *txPr=None*, *marker=None*, *dLbl=None*, *extLst=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

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

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.chartspace.**PivotFormatList**(*pivotFmt=()*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

tagname = 'pivotFmts'

class openpyxl.chart.chartspace.**PivotSource**(*name=None*, *fmtId=None*, *extLst=None*)
Bases: *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'

class openpyxl.chart.chartspace.**PlotArea**(*layout=None, dTable=None, spPr=None, areaChart=None, area3DChart=None, lineChart=None, line3DChart=None, stockChart=None, radarChart=None, scatterChart=None, pieChart=None, pie3DChart=None, doughnutChart=None, barChart=None, bar3DChart=None, ofPieChart=None, surfaceChart=None, surface3DChart=None, bubbleChart=None, valAx=(), catAx=(), serAx=(), dateAx=(), extLst=None*)

Bases: *openpyxl.descriptors.serialisable.Serialisable*

area3DChart
Values must be of type <class 'openpyxl.chart.area_chart.AreaChart3D'>

areaChart
Values must be of type <class 'openpyxl.chart.area_chart.AreaChart'>

bar3DChart
Values must be of type <class 'openpyxl.chart.bar_chart.BarChart3D'>

barChart
Values must be of type <class 'openpyxl.chart.bar_chart.BarChart'>

bubbleChart
Values must be of type <class 'openpyxl.chart.bubble_chart.BubbleChart'>

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

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

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

doughnutChart
Values must be of type <class 'openpyxl.chart.pie_chart.DoughnutChart'>

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

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

line3DChart
Values must be of type <class 'openpyxl.chart.line_chart.LineChart3D'>

lineChart
Values must be of type <class 'openpyxl.chart.line_chart.LineChart'>

ofPieChart
Values must be of type <class 'openpyxl.chart.pie_chart.ProjectedPieChart'>

pie3DChart
Values must be of type <class 'openpyxl.chart.pie_chart.PieChart3D'>

pieChart
Values must be of type <class 'openpyxl.chart.pie_chart.PieChart'>

radarChart
Values must be of type <class 'openpyxl.chart.radar_chart.RadarChart'>

scatterChart
Values must be of type <class 'openpyxl.chart.scatter_chart.ScatterChart'>

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

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

stockChart
Values must be of type <class 'openpyxl.chart.stock_chart.StockChart'>

surface3DChart
Values must be of type <class 'openpyxl.chart.surface_chart.SurfaceChart3D'>

surfaceChart
Values must be of type <class 'openpyxl.chart.surface_chart.SurfaceChart'>

tagname = 'plotArea'

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

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

class openpyxl.chart.chartspace.**PrintSettings**(*headerFooter=None*, *pageMargins=None*, *pageSetup=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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'>

tagname = 'printSettings'

class openpyxl.chart.chartspace.**Protection**(*chartObject=None*, *data=None*, *formatting=None*, *selection=None*, *userInterface=None*)
Bases: *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'>

class openpyxl.chart.chartspace.**RelId**
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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*)
Bases: *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'>

strLit
Values must be of type <class 'openpyxl.chart.data_source.StrData'>

strRef
Values must be of type <class 'openpyxl.chart.data_source.StrRef'>

class openpyxl.chart.data_source.**NumData** (*formatCode=None*, *ptCount=None*, *pt=()*, *extLst=None*)
Bases: *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*)
Bases: *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*)
Bases: *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*)
Bases: *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'>

class openpyxl.chart.data_source.**NumVal**(*idx=None*, *formatCode=None*, *v=None*)
Bases: *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 'float'>

class openpyxl.chart.data_source.**StrData**(*ptCount=None*, *pt=None*, *extLst=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

pt
Values must be of type <class 'openpyxl.chart.data_source.StrVal'>

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

tagname = 'strData'

class openpyxl.chart.data_source.**StrRef**(*f=None*, *strCache=None*, *extLst=None*)
Bases: *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*)
Bases: *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*)
Bases: *openpyxl.descriptors.nested.NestedMinMax*

allow_none = True

max = 500

min = 0

class openpyxl.chart.descriptors.**NestedOverlap**(***kw*)
Bases: *openpyxl.descriptors.nested.NestedMinMax*

allow_none = True

max = 100

min = -100

class openpyxl.chart.descriptors.**NumberFormatDescriptor**(**args*, ***kw*)
Bases: *openpyxl.descriptors.base.Typed*

Allow direct assignment of format code

allow_none = True

expected_type
alias of NumFmt

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*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

errBarType
Value must be one of {'plus', 'both', 'minus'}

errDir
Value must be one of {'y', 'x'}

errValType
Value must be one of {'percentage', 'cust', 'fixedVal', 'stdDev', 'stdErr'}

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

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'>

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

tagname = 'errBars'

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

openpyxl.chart.label module

class openpyxl.chart.label.**DataLabel**(*idx=0*, ***kw*)
Bases: openpyxl.chart.label._DataLabelBase

dLblPos
Value must be one of {'bestFit', 'inEnd', 'r', 'ctr', 'l', 'b', 'inBase', 'outEnd', '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=()*, ***kw*)
Bases: 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 {'bestFit', 'inEnd', 'r', 'ctr', 'l', 'b', 'inBase', 'outEnd', 't'}

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*)
Bases: *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=None*, *hMode=None*, *x=None*, *y=None*, *w=None*, *h=None*, *extLst=None*)
Bases: *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 {'edge', 'factor'}

layoutTarget
Value must be one of {'inner', 'outer'}

tagname = 'manualLayout'

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

wMode
Value must be one of {'edge', 'factor'}

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

xMode
Value must be one of {'edge', 'factor'}

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

yMode
Value must be one of {'edge', 'factor'}

openpyxl.chart.legend module

class `openpyxl.chart.legend.`**`Legend`**(*legendPos='r'*, *legendEntry=None*, *layout=None*, *overlay=None*, *spPr=None*, *txPr=None*, *extLst=None*)
Bases: `openpyxl.descriptors.serialisable.Serialisable`

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

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

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

legendPos
Value must be one of {'b', 'r', 'l', 'tr', 't'}

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

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

tagname = 'legend'

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

class `openpyxl.chart.legend.`**`LegendEntry`**(*idx=0*, *delete=False*, *txPr=None*, *extLst=None*)
Bases: `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*, *axId=None*, *extLst=None*, ***kw*)
Bases: `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 {'stacked', 'standard', '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*, *axId=None*, ***kw*)
Bases: `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 {'stacked', 'standard', '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*)
Bases: `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'>

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*)
Bases: `openpyxl.descriptors.serialisable.Serialisable`

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

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 {'star', 'square', 'diamond', 'auto', 'triangle', 'picture', 'plus', 'circle', 'dash', 'dot', 'x'}

tagname = 'marker'

openpyxl.chart.picture module

class openpyxl.chart.picture.**PictureOptions**(*applyToFront=None*, *applyToSides=None*, *applyToEnd=None*, *pictureFormat=None*, *pictureStackUnit=None*)
Bases: *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 {'stretch', 'stack', 'stackScale'}

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

tagname = 'pictureOptions'

openpyxl.chart.pie_chart module

class openpyxl.chart.pie_chart.**CustomSplit**(*secondPiePt=()*)
Bases: *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*)
Bases: 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*)
Bases: 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*)
Bases: 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*)
Bases: 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'>

ofPieType
Value must be one of {'bar', 'pie'}

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', 'pos', 'cust', 'percent', 'val'}

tagname = 'ofPieChart'

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

openpyxl.chart.radar_chart module

class `openpyxl.chart.radar_chart.`**`RadarChart`**(*radarStyle='standard', varyColors=None, ser=(), dLbls=None, axId=None, extLst=None*)
Bases: `openpyxl.chart._chart.ChartBase`

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

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

radarStyle
Value must be one of {'filled', 'standard', 'marker'}

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

tagname = 'radarChart'

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.reference module

class `openpyxl.chart.reference.`**`DummyWorksheet`**(*title*)
Bases: `object`

class `openpyxl.chart.reference.`**`Reference`**(*worksheet=None, min_col=None, min_row=None, max_col=None, max_row=None, range_string=None*)
Bases: *`openpyxl.descriptors.Strict`*

Normalise cell range references

cells
Return a flattened list of all cells (by column)

cols
Return all cells in range by row

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 cells in range by column

sheetname

openpyxl.chart.scatter_chart module

class openpyxl.chart.scatter_chart.**ScatterChart**(*scatterStyle=None*, *varyColors=None*, *ser=()*, *dLbls=None*, *axId=None*, *extLst=None*)
Bases: openpyxl.chart._chart.ChartBase

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

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

scatterStyle
Value must be one of {'lineMarker', 'smooth', 'smoothMarker', 'line', 'marker'}

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*)

Bases: *`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

`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'>

`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'>

`pictureOptions`
Values must be of type <class 'openpyxl.chart.picture.PictureOptions'>

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

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

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

`tagname` = **'ser'**

`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'>

class `openpyxl.chart.series.`**`SeriesLabel`**(*strRef=None*, *v=None*)
Bases: `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'>

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*)
Bases: `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*)
Bases: *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 {'gray', 'blackWhite', 'hidden', 'auto', 'grayWhite', 'ltGray', 'white', 'clr', 'invGray', 'blackGray', 'black'}

custGeom
Values must be of type <class 'openpyxl.drawing.shapes.CustomGeometry2D'>

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

gradFill
Values must be of type <class 'openpyxl.drawing.fill.GradientFillProperties'>

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.shapes.PresetGeometry2D'>

scene3d
Values must be of type <class 'openpyxl.drawing.shapes.Scene3D'>

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

sp3d
Values must be of type <class 'openpyxl.drawing.shapes.Shape3D'>

tagname = 'spPr'

xfrm
Values must be of type <class 'openpyxl.drawing.shapes.Transform2D'>

openpyxl.chart.stock_chart module

class `openpyxl.chart.stock_chart.`**StockChart**(*ser=()*, *dLbls=None*, *dropLines=None*, *hiLowLines=None*, *upDownBars=None*, *axId=None*, *extLst=None*)
Bases: `openpyxl.chart._chart.ChartBase`

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'>

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*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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=()*)
Bases: *`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*)
  Bases: *`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`**(*axId=None*, ***kw*)
  Bases: `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*)
  Bases: *`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

  `tagname` **= 'rich'**

class `openpyxl.chart.text.`**`Text`**(*strRef=None*, *rich=None*)

Bases: `openpyxl.descriptors.serialisable.Serialisable`

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

`strRef`
Values must be of type <class 'openpyxl.chart.data_source.StrRef'>

openpyxl.chart.title module

class `openpyxl.chart.title.`**`Title`**(*tx=None*, *layout=None*, *overlay=None*, *spPr=None*, *txPr=None*, *extLst=None*)

Bases: `openpyxl.descriptors.serialisable.Serialisable`

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

`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'**

`tx`
Values must be of type <class 'openpyxl.chart.text.Text'>

`txPr`
Values must be of type <class 'openpyxl.drawing.text.RichTextProperties'>

class `openpyxl.chart.title.`**`TitleDescriptor`**(**args*, ***kw*)

Bases: `openpyxl.descriptors.base.Typed`

`allow_none` = **True**

`expected_type`
alias of `Title`

`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*)

Bases: `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'>

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 {'linear', 'power', 'poly', 'movingAvg', 'exp', 'log'}

class `openpyxl.chart.trendline.`**`TrendlineLabel`**(*layout=None*, *tx=None*, *numFmt=None*, *spPr=None*, *txPr=None*, *extLst=None*)
Bases: `openpyxl.descriptors.serialisable.Serialisable`

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

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'

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*)
Bases: `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

Subpackages

openpyxl.chartsheet.tests package

Submodules

openpyxl.chartsheet.tests.test_chartsheet module

`openpyxl.chartsheet.tests.test_chartsheet.`**`Chartsheet`**`()`

class `openpyxl.chartsheet.tests.test_chartsheet.`**`DummyWorkbook`**
Bases: `object`

class `openpyxl.chartsheet.tests.test_chartsheet.`**`TestChartsheet`**
Bases: `object`

`test_ctor`(*Chartsheet*)

`test_read`(*Chartsheet*)

`test_write`(*Chartsheet*)

`test_write_charts`(*Chartsheet*)

openpyxl.chartsheet.tests.test_custom module

`openpyxl.chartsheet.tests.test_custom.`**`CustomChartsheetView`**`()`

`openpyxl.chartsheet.tests.test_custom.`**`CustomChartsheetViews`**`()`

class `openpyxl.chartsheet.tests.test_custom.`**`TestCustomChartsheetView`**
Bases: `object`

`test_read`(*CustomChartsheetView*)

`test_write`(*CustomChartsheetView*)

class `openpyxl.chartsheet.tests.test_custom.`**`TestCustomChartsheetViews`**
Bases: `object`

`test_read`(*CustomChartsheetViews*)

`test_write`(*CustomChartsheetViews*)

openpyxl.chartsheet.tests.test_properties module

`openpyxl.chartsheet.tests.test_properties.`**`ChartsheetProperties`**`()`

class `openpyxl.chartsheet.tests.test_properties.`**`TestChartsheetPr`**
Bases: `object`

`test_read`(*ChartsheetProperties*)

`test_write`(*ChartsheetProperties*)

openpyxl.chartsheet.tests.test_protection module

openpyxl.chartsheet.tests.test_protection.**ChartsheetProtection**()

class openpyxl.chartsheet.tests.test_protection.**TestChartsheetProtection**

Bases: object

test_read(*ChartsheetProtection*)

test_write(*ChartsheetProtection*)

openpyxl.chartsheet.tests.test_publish module

class openpyxl.chartsheet.tests.test_publish.**TestWebPublishItems**

Bases: object

test_read(*WebPublishItems*)

test_write(*WebPublishItems*)

class openpyxl.chartsheet.tests.test_publish.**TestWebPulishItem**

Bases: object

test_read(*WebPublishItem*)

test_write(*WebPublishItem*)

openpyxl.chartsheet.tests.test_publish.**WebPublishItem**()

openpyxl.chartsheet.tests.test_publish.**WebPublishItems**()

openpyxl.chartsheet.tests.test_relation module

openpyxl.chartsheet.tests.test_relation.**DrawingHF**()

openpyxl.chartsheet.tests.test_relation.**SheetBackgroundPicture**()

class openpyxl.chartsheet.tests.test_relation.**TestDrawingHF**

Bases: object

test_read(*DrawingHF*)

test_write(*DrawingHF*)

class openpyxl.chartsheet.tests.test_relation.**TestSheetBackgroundPicture**

Bases: object

test_read(*SheetBackgroundPicture*)

test_write(*SheetBackgroundPicture*)

openpyxl.chartsheet.tests.test_views module

openpyxl.chartsheet.tests.test_views.**ChartsheetView**()

openpyxl.chartsheet.tests.test_views.**ChartsheetViewList**()

class openpyxl.chartsheet.tests.test_views.**TestChartsheetView**

Bases: object

test_read(*ChartsheetView*)

test_write(*ChartsheetView*)

class openpyxl.chartsheet.tests.test_views.**TestChartsheetViewList**

Bases: object

test_read(*ChartsheetViewList*)

test_write(*ChartsheetViewList*)

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'*)

Bases: `openpyxl.workbook.child._WorkbookChild`, `openpyxl.descriptors.serialisable.Serialisa`

`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'>

`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 {'visible', 'veryHidden', '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*)

Bases: *`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 {'visible', 'veryHidden', 'hidden'}

`tagname` = 'customSheetView'

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

class `openpyxl.chartsheet.custom.`**`CustomChartsheetViews`**(*customSheetView=None*)

Bases: *`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*)

Bases: *`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*)

Bases: *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

hash_password(*password*)

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*)
Bases: *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 {'query', 'sheet', 'pivotTable', 'label', 'printArea', 'autoFilter', 'range', 'chart'}

tagname = 'webPublishItem'

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

class openpyxl.chartsheet.publish.**WebPublishItems**(*count=None*, *webPublishItem=None*)
Bases: *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*)

Bases: *openpyxl.descriptors.serialisable.Serialisable*

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'>

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'>

class openpyxl.chartsheet.relation.**SheetBackgroundPicture**(*id*)
Bases: *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*)
Bases: *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*)
Bases: *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=()*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

tagname = 'authors'

openpyxl.comments.comments module

class `openpyxl.comments.comments.`**`Comment`**(*text*, *author*)

Bases: `object`

`parent`

`text`

Any comment text stripped of all formatting.

openpyxl.comments.properties module

class `openpyxl.comments.properties.`**`Comment`**(*ref=''*, *authorId=0*, *guid=None*, *shapeId=0*, *text=None*, *commentPr=None*, *author=None*)

Bases: *`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.properties.Properties'>

`content`

Remove all inline formatting and stuff

`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.properties.`**`CommentSheet`**(*authors=None*, *commentList=None*, *extLst=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`authors`

Values must be of type <class 'openpyxl.comments.author.AuthorList'>

`commentList`

Wrap a sequence in an containing object

`extLst`

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

`tagname` **= 'comments'**

`to_tree`()

class `openpyxl.comments.properties.`**`ObjectAnchor`**(*moveWithCells=None*, *sizeWithCells=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`moveWithCells`

Values must be of type <class 'bool'>

`sizeWithCells`

Values must be of type <class 'bool'>

class `openpyxl.comments.properties.`**`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*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`altText`
Values must be of type <class 'str'>

`anchor`
Values must be of type <class 'openpyxl.comments.properties.ObjectAnchor'>

`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 {'right', 'justify', 'center', 'distributed', 'left'}

`textVAlign`
Value must be one of {'justify', 'center', 'top', 'bottom', 'distributed'}

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

openpyxl.comments.reader module

`openpyxl.comments.reader.`**`get_comments_file`**(*worksheet_path*, *archive*, *valid_files*)
Returns the XML filename in the archive which contains the comments for the spreadsheet with codename sheet_codename.

`openpyxl.comments.reader.`**`read_comments`**(*ws*, *xml_source*)
Given a worksheet and the XML of its comments file, assigns comments to cells

openpyxl.comments.writer module

class `openpyxl.comments.writer.`**`CommentWriter`**`(`*sheet*`)`

Bases: `object`

`write_comments``()`

Create list of comments and authors

`write_comments_vml``()`

openpyxl.descriptors package

class `openpyxl.descriptors.`**`MetaSerialisable`**

Bases: `type`

class `openpyxl.descriptors.`**`MetaStrict`**

Bases: `type`

class `openpyxl.descriptors.`**`Strict`**

Bases: `object`

Submodules

openpyxl.descriptors.base module

class `openpyxl.descriptors.base.`**`ASCII`**`(`**args*, ***kw*`)`

Bases: *`openpyxl.descriptors.base.Typed`*

`expected_type`

alias of `bytes`

class `openpyxl.descriptors.base.`**`Alias`**`(`*alias*`)`

Bases: *`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*`)`

Bases: *`openpyxl.descriptors.base.Convertible`*

`expected_type`

alias of `bool`

class `openpyxl.descriptors.base.`**`Convertible`**`(`**args*, ***kw*`)`

Bases: *`openpyxl.descriptors.base.Typed`*

Values must be convertible to a particular type

class `openpyxl.descriptors.base.`**`DateTime`**`(`**args*, ***kw*`)`

Bases: *`openpyxl.descriptors.base.Typed`*

`expected_type`

alias of `datetime`

class `openpyxl.descriptors.base.`**`Default`**`(`*name=None*, ***kw*`)`

Bases: *`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*`)`

Bases: `object`

class `openpyxl.descriptors.base.`**`Float`**`(`**args*, ***kw*`)`
Bases: *`openpyxl.descriptors.base.Convertible`*

`expected_type`
alias of `float`

class `openpyxl.descriptors.base.`**`Integer`**`(`**args*, ***kw*`)`
Bases: *`openpyxl.descriptors.base.Convertible`*

`expected_type`
alias of `int`

class `openpyxl.descriptors.base.`**`Length`**`(`*name=None*, ***kw*`)`
Bases: *`openpyxl.descriptors.base.Descriptor`*

class `openpyxl.descriptors.base.`**`MatchPattern`**`(`*name=None*, ***kw*`)`
Bases: *`openpyxl.descriptors.base.Descriptor`*

Values must match a regex pattern

`allow_none` **= False**

class `openpyxl.descriptors.base.`**`Max`**`(`***kw*`)`
Bases: *`openpyxl.descriptors.base.Convertible`*

Values must be less than a *max* value

`allow_none` **= False**

`expected_type`
alias of `float`

class `openpyxl.descriptors.base.`**`Min`**`(`***kw*`)`
Bases: *`openpyxl.descriptors.base.Convertible`*

Values must be greater than a *min* value

`allow_none` **= False**

`expected_type`
alias of `float`

class `openpyxl.descriptors.base.`**`MinMax`**`(`***kw*`)`
Bases: *`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*`)`
Bases: *`openpyxl.descriptors.base.Set`*

'none' will be treated as None

class `openpyxl.descriptors.base.`**`Set`**`(`*name=None*, ***kw*`)`
Bases: *`openpyxl.descriptors.base.Descriptor`*

Value can only be from a set of know values

class `openpyxl.descriptors.base.`**`String`**`(`**args*, ***kw*`)`
Bases: *`openpyxl.descriptors.base.Typed`*

`expected_type`
alias of `str`

class `openpyxl.descriptors.base.`**`Tuple`**`(`**args*, ***kw*`)`
Bases: *`openpyxl.descriptors.base.Typed`*

expected_type
alias of `tuple`

class `openpyxl.descriptors.base.`**`Typed`**(**args*, ***kw*)
Bases: *`openpyxl.descriptors.base.Descriptor`*

Values must of a particular type

allow_none = False

expected_type
alias of `NoneType`

nested = False

openpyxl.descriptors.excel module

class `openpyxl.descriptors.excel.`**`Base64Binary`**(*name=None*, ***kw*)
Bases: *`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.`**`Extension`**(*uri=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

class `openpyxl.descriptors.excel.`**`ExtensionList`**(*ext=()*)
Bases: *`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*)
Bases: *`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*)
Bases: *`openpyxl.descriptors.base.MatchPattern`*

pattern = '[0-9a-fA-F]+$'

class `openpyxl.descriptors.excel.`**`Percentage`**(*name=None*, ***kw*)
Bases: *`openpyxl.descriptors.base.MatchPattern`*

pattern = '((100)|([0-9][0-9]?))(\\.[0-9][0-9]?)?%'

class `openpyxl.descriptors.excel.`**`Relation`**(**args*, ***kw*)
Bases: *`openpyxl.descriptors.base.String`*

allow_none = True

namespace = 'http://schemas.openxmlformats.org/officeDocument/2006/relationships'

class `openpyxl.descriptors.excel.`**`TextPoint`**(***kw*)
Bases: *`openpyxl.descriptors.base.MinMax`*

Size in hundredths of points. In theory other units of measurement can be used but these are unbounded

expected_type
alias of `int`

max = 400000

min = -400000

class openpyxl.descriptors.excel.**UniversalMeasure**(*name=None*, ***kw*)
Bases: *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

class openpyxl.descriptors.nested.**EmptyTag**(**args*, ***kw*)
Bases: *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*)
Bases: *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*)
Bases: *openpyxl.descriptors.nested.NestedValue*, *openpyxl.descriptors.base.Bool*

from_tree(*node*)

class openpyxl.descriptors.nested.**NestedFloat**(**args*, ***kw*)
Bases: *openpyxl.descriptors.nested.NestedValue*, *openpyxl.descriptors.base.Float*

class openpyxl.descriptors.nested.**NestedInteger**(**args*, ***kw*)
Bases: *openpyxl.descriptors.nested.NestedValue*, *openpyxl.descriptors.base.Integer*

class openpyxl.descriptors.nested.**NestedMinMax**(***kw*)
Bases: *openpyxl.descriptors.nested.Nested*, *openpyxl.descriptors.base.MinMax*

class openpyxl.descriptors.nested.**NestedNoneSet**(*name=None*, ***kw*)
Bases: *openpyxl.descriptors.nested.Nested*, *openpyxl.descriptors.base.NoneSet*

class openpyxl.descriptors.nested.**NestedSet**(*name=None*, ***kw*)
Bases: *openpyxl.descriptors.nested.Nested*, *openpyxl.descriptors.base.Set*

class openpyxl.descriptors.nested.**NestedString**(**args*, ***kw*)
Bases: *openpyxl.descriptors.nested.NestedValue*, *openpyxl.descriptors.base.String*

class openpyxl.descriptors.nested.**NestedText**(**args*, ***kw*)
Bases: *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*)
Bases: *`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.`**`NestedSequence`**(*name=None*, ***kw*)
Bases: *`openpyxl.descriptors.sequence.Sequence`*

Wrap a sequence in an containing object

`count` = **False**

`from_tree`(*node*)

`to_tree`(*tagname*, *obj*, *namespace=None*)

class `openpyxl.descriptors.sequence.`**`Sequence`**(*name=None*, ***kw*)
Bases: *`openpyxl.descriptors.base.Descriptor`*

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

`expected_type`
alias of `NoneType`

`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*)
Bases: *`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*)

openpyxl.descriptors.serialisable module

class `openpyxl.descriptors.serialisable.`**`Serialisable`**
Bases: `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.drawing package

Submodules

openpyxl.drawing.colors module

class `openpyxl.drawing.colors.`**`ColorChoice`**(*scrgbClr=None*, *srgbClr=None*, *hslClr=None*, *sysClr=None*, *schemeClr=None*, *prstClr=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`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 {'lightSlateGray', 'oldLace', 'ltPink', 'lightSteelBlue', 'peachPuff', 'steelBlue', 'dkViolet', 'greenYellow', 'darkGrey', 'moccasin', 'snow', 'yellow', 'firebrick', 'medTurquoise', 'gray', 'lightSkyBlue', 'darkOrchid', 'medSeaGreen', 'salmon', 'mistyRose', 'black', 'aquamarine', 'dkOrange', 'mintCream', 'red', 'magenta', 'ltSalmon', 'indianRed', 'dkGoldenrod', 'lightSeaGreen', 'paleVioletRed', 'royalBlue', 'darkSlateBlue', 'pink', 'crimson', 'darkGoldenrod', 'darkTurquoise', 'dimGray', 'tomato', 'dkOliveGreen', 'springGreen', 'dkKhaki', 'mediumSlateBlue', 'lightBlue', 'lavenderBlush', 'darkViolet', 'lightCyan', 'bisque', 'lightSlateGrey', 'oliveDrab', 'peru', 'darkBlue', 'wheat', 'blanchedAlmond', 'maroon', 'midnightBlue', 'darkGray', 'grey', 'antiqueWhite', 'darkOrange', 'dkGreen', 'goldenrod', 'orchid', 'navy', 'ltGray', 'ltSkyBlue', 'ltSteelBlue', 'medSlateBlue', 'navajoWhite', 'violet', 'gold', 'dkSlateGrey', 'dkTurquoise', 'paleGoldenrod', 'dkGray', 'medPurple', 'mediumPurple', 'darkGreen', 'darkSeaGreen', 'saddleBrown', 'dkRed', 'skyBlue', 'teal', 'ghostWhite', 'mediumVioletRed', 'ltSlateGray', 'cornsilk', 'seaGreen', 'silver', 'honeydew', 'ltGreen', 'dkSeaGreen', 'deepPink', 'medAquamarine', 'dkMagenta', 'lightCoral', 'medBlue', 'medOrchid', 'darkSlateGray', 'aqua', 'beige', 'ltSeaGreen', 'lemonChiffon', 'orange', 'whiteSmoke', 'blue', 'lightGoldenrodYellow', 'cyan', 'dkCyan', 'indigo', 'chocolate', 'lightSalmon', 'coral', 'darkSalmon', 'dkGrey', 'sienna', 'dkSalmon', 'papayaWhip', 'darkCyan', 'thistle', 'khaki', 'lightPink', 'dimGrey', 'ltGrey', 'cornflowerBlue', 'ltSlateGrey', 'purple', 'orangeRed', 'ivory', 'dkOrchid', 'floralWhite', 'linen', 'rosyBrown', 'gainsboro', 'olive', 'hotPink', 'lightGreen', 'dkBlue', 'dodgerBlue', 'darkRed', 'blueViolet', 'darkSlateGrey', 'ltGoldenrodYellow', 'mediumOrchid', 'burlyWood', 'ltYellow', 'lawnGreen', 'azure', 'limeGreen', 'lightYellow', 'dkSlateBlue', 'ltBlue', 'slateBlue', 'mediumAquamarine', 'tan', 'green', 'slateGrey', 'lightGray', 'medVioletRed', 'dkSlateGray', 'lavender', 'darkKhaki', 'cadetBlue', 'mediumSeaGreen', 'darkOliveGreen', 'paleGreen', 'ltCoral', 'mediumBlue', 'sandyBrown', 'paleTurquoise', 'mediumSpringGreen', 'brown', 'fuchsia', 'deepSkyBlue', 'plum', 'seaShell', 'ltCyan', 'forestGreen', 'slateGray', 'lightGrey', 'chartreuse', 'aliceBlue', 'lime', 'mediumTurquoise', 'darkMagenta', 'medSpringGreen', 'yellowGreen', 'powderBlue', 'turquoise', 'white'}

`schemeClr`
Value must be one of {'accent3', 'phClr', 'accent5', 'hlink', 'dk1', 'accent6', 'bg1', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'bg2', 'folHlink', 'tx1', 'tx2'}

`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*)

Bases: *`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
alias of `ColorChoice`

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*)

Bases: `openpyxl.descriptors.serialisable.Serialisable`

accent1
Value must be one of {'accent3', 'accent5', 'hlink', 'dk1', 'accent6', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'folHlink'}

accent2
Value must be one of {'accent3', 'accent5', 'hlink', 'dk1', 'accent6', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'folHlink'}

accent3
Value must be one of {'accent3', 'accent5', 'hlink', 'dk1', 'accent6', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'folHlink'}

accent4
Value must be one of {'accent3', 'accent5', 'hlink', 'dk1', 'accent6', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'folHlink'}

accent5
Value must be one of {'accent3', 'accent5', 'hlink', 'dk1', 'accent6', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'folHlink'}

accent6
Value must be one of {'accent3', 'accent5', 'hlink', 'dk1', 'accent6', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'folHlink'}

bg1
Value must be one of {'accent3', 'accent5', 'hlink', 'dk1', 'accent6', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'folHlink'}

bg2
Value must be one of {'accent3', 'accent5', 'hlink', 'dk1', 'accent6', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'folHlink'}

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

folHlink
Value must be one of {'accent3', 'accent5', 'hlink', 'dk1', 'accent6', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'folHlink'}

hlink
Value must be one of {'accent3', 'accent5', 'hlink', 'dk1', 'accent6', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'folHlink'}

tagname = 'clrMapOvr'

tx1
Value must be one of {'accent3', 'accent5', 'hlink', 'dk1', 'accent6', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'folHlink'}

tx2
Value must be one of {'accent3', 'accent5', 'hlink', 'dk1', 'accent6', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'folHlink'}

class openpyxl.drawing.colors.**HSLColor**(*hue=None*, *sat=None*, *lum=None*)
Bases: *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*)
Bases: *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.**SystemColor**(*val='bg1'*, *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*)
Bases: *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 'openpyxl.styles.colors.RGB'>

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

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

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

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 {'accent3', 'phClr', 'accent5', 'hlink', 'dk1', 'accent6', 'bg1', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'bg2', 'folHlink', 'tx1', 'tx2'}

class `openpyxl.drawing.colors.`**`Transform`**
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

openpyxl.drawing.drawing module

class `openpyxl.drawing.drawing.`**`Drawing`**
Bases: `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

height

set_dimension(*w=0*, *h=0*)

width

openpyxl.drawing.effect module

class `openpyxl.drawing.effect.`**`AlphaBiLevelEffect`**(*thresh=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

class `openpyxl.drawing.effect.`**`AlphaCeilingEffect`**
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

class `openpyxl.drawing.effect.`**`AlphaFloorEffect`**
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

class `openpyxl.drawing.effect.`**`AlphaInverseEffect`**
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

class `openpyxl.drawing.effect.`**`AlphaModulateEffect`**(*cont=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

cont
Values must be of type <class 'openpyxl.drawing.effect.EffectContainer'>

class `openpyxl.drawing.effect.`**`AlphaModulateFixedEffect`**(*amt=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

class `openpyxl.drawing.effect.`**`AlphaReplaceEffect`**(*a=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`a`
Values must be of type <class 'int'>

class `openpyxl.drawing.effect.`**`BiLevelEffect`**(*thresh=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`thresh`
Values must be of type <class 'int'>

class `openpyxl.drawing.effect.`**`BlurEffect`**(*rad=None*, *grow=None*)
Bases: *`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`**
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

class `openpyxl.drawing.effect.`**`ColorChangeEffect`**(*useA=None*, *clrFrom=None*, *clrTo=None*)
Bases: *`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`**
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

class `openpyxl.drawing.effect.`**`DuotoneEffect`**
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

class `openpyxl.drawing.effect.`**`EffectContainer`**(*type=None*, *name=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`name`
Values must be of type <class 'str'>

`type`
Value must be one of {'sib', 'tree'}

class `openpyxl.drawing.effect.`**`EffectList`**(*blur=None*, *fillOverlay=None*, *glow=None*, *innerShdw=None*, *outerShdw=None*, *prstShdw=None*, *reflection=None*, *softEdge=None*)
Bases: *`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.OuterShadowEffect'>

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*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

blend
Value must be one of {'over', 'mult', 'screen', 'darken', 'lighten'}

class `openpyxl.drawing.effect.`**`GlowEffect`**(*rad=None*, ***kw*)
Bases: *`openpyxl.drawing.colors.ColorChoice`*

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

prstClr
Value must be one of {'lightSlateGray', 'oldLace', 'ltPink', 'lightSteelBlue', 'peachPuff', 'steelBlue', 'dkViolet', 'greenYellow', 'darkGrey', 'moccasin', 'snow', 'yellow', 'firebrick', 'medTurquoise', 'gray', 'lightSkyBlue', 'darkOrchid', 'medSeaGreen', 'salmon', 'mistyRose', 'black', 'aquamarine', 'dkOrange', 'mintCream', 'red', 'magenta', 'ltSalmon', 'indianRed', 'dkGoldenrod', 'lightSeaGreen', 'paleVioletRed', 'royalBlue', 'darkSlateBlue', 'pink', 'crimson', 'darkGoldenrod', 'darkTurquoise', 'dimGray', 'tomato', 'dkOliveGreen', 'springGreen', 'dkKhaki', 'mediumSlateBlue', 'lightBlue', 'lavenderBlush', 'darkViolet', 'lightCyan', 'bisque', 'lightSlateGrey', 'oliveDrab', 'peru', 'darkBlue', 'wheat', 'blanchedAlmond', 'maroon', 'midnightBlue', 'darkGray', 'grey', 'antiqueWhite', 'darkOrange', 'dkGreen', 'goldenrod', 'orchid', 'navy', 'ltGray', 'ltSkyBlue', 'ltSteelBlue', 'medSlateBlue', 'navajoWhite', 'violet', 'gold', 'dkSlateGrey', 'dkTurquoise', 'paleGoldenrod', 'dkGray', 'medPurple', 'mediumPurple', 'darkGreen', 'darkSeaGreen', 'saddleBrown', 'dkRed', 'skyBlue', 'teal', 'ghostWhite', 'mediumVioletRed', 'ltSlateGray', 'cornsilk', 'seaGreen', 'silver', 'honeydew', 'ltGreen', 'dkSeaGreen', 'deepPink', 'medAquamarine', 'dkMagenta', 'lightCoral', 'medBlue', 'medOrchid', 'darkSlateGray', 'aqua', 'beige', 'ltSeaGreen', 'lemonChiffon', 'orange', 'whiteSmoke', 'blue', 'lightGoldenrodYellow', 'cyan', 'dkCyan', 'indigo', 'chocolate', 'lightSalmon', 'coral', 'darkSalmon', 'dkGrey', 'sienna', 'dkSalmon', 'papayaWhip', 'darkCyan', 'thistle', 'khaki', 'lightPink', 'dimGrey', 'ltGrey', 'cornflowerBlue', 'ltSlateGrey', 'purple', 'orangeRed', 'ivory', 'dkOrchid', 'floralWhite', 'linen', 'rosyBrown', 'gainsboro', 'olive', 'hotPink', 'lightGreen', 'dkBlue', 'dodgerBlue', 'darkRed', 'blueViolet', 'darkSlateGrey', 'ltGoldenrodYellow', 'mediumOrchid', 'burlyWood', 'ltYellow', 'lawnGreen', 'azure', 'limeGreen', 'lightYellow', 'dkSlateBlue', 'ltBlue', 'slateBlue', 'mediumAquamarine', 'tan', 'green', 'slateGrey', 'lightGray', 'medVioletRed', 'dkSlateGray', 'lavender', 'darkKhaki', 'cadetBlue', 'mediumSeaGreen', 'darkOliveGreen', 'paleGreen', 'ltCoral', 'mediumBlue', 'sandyBrown', 'paleTurquoise', 'mediumSpringGreen', 'brown', 'fuchsia', 'deepSkyBlue', 'plum', 'seaShell', 'ltCyan', 'forestGreen', 'slateGray', 'lightGrey', 'chartreuse', 'aliceBlue', 'lime', 'mediumTurquoise', 'darkMagenta', 'medSpringGreen', 'yellowGreen', 'powderBlue', 'turquoise', 'white'}

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

schemeClr
Value must be one of {'accent3', 'phClr', 'accent5', 'hlink', 'dk1', 'accent6', 'bg1', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'bg2', 'folHlink', 'tx1', 'tx2'}

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**
Bases: *openpyxl.descriptors.serialisable.Serialisable*

class openpyxl.drawing.effect.**HSLEffect**(*hue=None*, *sat=None*, *lum=None*)
Bases: *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*)
Bases: *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 {'lightSlateGray', 'oldLace', 'ltPink', 'lightSteelBlue', 'peachPuff', 'steelBlue', 'dkViolet', 'greenYellow', 'darkGrey', 'moccasin', 'snow', 'yellow', 'firebrick', 'medTurquoise', 'gray', 'lightSkyBlue', 'darkOrchid', 'medSeaGreen', 'salmon', 'mistyRose', 'black', 'aquamarine', 'dkOrange', 'mintCream', 'red', 'magenta', 'ltSalmon', 'indianRed', 'dkGoldenrod', 'lightSeaGreen', 'paleVioletRed', 'royalBlue', 'darkSlateBlue', 'pink', 'crimson', 'darkGoldenrod', 'darkTurquoise', 'dimGray', 'tomato', 'dkOliveGreen', 'springGreen', 'dkKhaki', 'mediumSlateBlue', 'lightBlue', 'lavenderBlush', 'darkViolet', 'lightCyan', 'bisque', 'lightSlateGrey', 'oliveDrab', 'peru', 'darkBlue', 'wheat', 'blanchedAlmond', 'maroon', 'midnightBlue', 'darkGray', 'grey', 'antiqueWhite', 'darkOrange', 'dkGreen', 'goldenrod', 'orchid', 'navy', 'ltGray', 'ltSkyBlue', 'ltSteelBlue', 'medSlateBlue', 'navajoWhite', 'violet', 'gold', 'dkSlateGrey', 'dkTurquoise', 'paleGoldenrod', 'dkGray', 'medPurple', 'mediumPurple', 'darkGreen', 'darkSeaGreen', 'saddleBrown', 'dkRed', 'skyBlue', 'teal', 'ghostWhite', 'mediumVioletRed', 'ltSlateGray', 'cornsilk', 'seaGreen', 'silver', 'honeydew', 'ltGreen', 'dkSeaGreen', 'deepPink', 'medAquamarine', 'dkMagenta', 'lightCoral', 'medBlue', 'medOrchid', 'darkSlateGray', 'aqua', 'beige', 'ltSeaGreen', 'lemonChiffon', 'orange', 'whiteSmoke', 'blue', 'lightGoldenrodYellow', 'cyan', 'dkCyan', 'indigo', 'chocolate', 'lightSalmon', 'coral', 'darkSalmon', 'dkGrey', 'sienna', 'dkSalmon',

'papayaWhip', 'darkCyan', 'thistle', 'khaki', 'lightPink', 'dimGrey', 'ltGrey', 'cornflowerBlue', 'ltSlateGrey', 'purple', 'orangeRed', 'ivory', 'dkOrchid', 'floralWhite', 'linen', 'rosyBrown', 'gainsboro', 'olive', 'hotPink', 'lightGreen', 'dkBlue', 'dodgerBlue', 'darkRed', 'blueViolet', 'darkSlateGrey', 'ltGoldenrodYellow', 'mediumOrchid', 'burlyWood', 'ltYellow', 'lawnGreen', 'azure', 'limeGreen', 'lightYellow', 'dkSlateBlue', 'ltBlue', 'slateBlue', 'mediumAquamarine', 'tan', 'green', 'slateGrey', 'lightGray', 'medVioletRed', 'dkSlateGray', 'lavender', 'darkKhaki', 'cadetBlue', 'mediumSeaGreen', 'darkOliveGreen', 'paleGreen', 'ltCoral', 'mediumBlue', 'sandyBrown', 'paleTurquoise', 'mediumSpringGreen', 'brown', 'fuchsia', 'deepSkyBlue', 'plum', 'seaShell', 'ltCyan', 'forestGreen', 'slateGray', 'lightGrey', 'chartreuse', 'aliceBlue', 'lime', 'mediumTurquoise', 'darkMagenta', 'medSpringGreen', 'yellowGreen', 'powderBlue', 'turquoise', 'white'}

schemeClr
Value must be one of {'accent3', 'phClr', 'accent5', 'hlink', 'dk1', 'accent6', 'bg1', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'bg2', 'folHlink', 'tx1', 'tx2'}

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=None*, *contrast=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

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

class openpyxl.drawing.effect.**OuterShadowEffect**(*blurRad=None*, *dist=None*, *dir=None*, *sx=None*, *sy=None*, *kx=None*, *ky=None*, *algn=None*, *rotWithShape=None*, ***kw*)
Bases: *openpyxl.drawing.colors.ColorChoice*

algn
Value must be one of {'br', 'r', 'ctr', 'l', 'b', 'tl', 'bl', 'tr', '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 {'lightSlateGray', 'oldLace', 'ltPink', 'lightSteelBlue', 'peachPuff', 'steelBlue', 'dkViolet', 'greenYellow', 'darkGrey', 'moccasin', 'snow', 'yellow', 'firebrick', 'medTurquoise',

'gray', 'lightSkyBlue', 'darkOrchid', 'medSeaGreen', 'salmon', 'mistyRose', 'black', 'aquamarine', 'dkOrange', 'mintCream', 'red', 'magenta', 'ltSalmon', 'indianRed', 'dkGoldenrod', 'lightSeaGreen', 'paleVioletRed', 'royalBlue', 'darkSlateBlue', 'pink', 'crimson', 'darkGoldenrod', 'darkTurquoise', 'dimGray', 'tomato', 'dkOliveGreen', 'springGreen', 'dkKhaki', 'mediumSlateBlue', 'lightBlue', 'lavenderBlush', 'darkViolet', 'lightCyan', 'bisque', 'lightSlateGrey', 'oliveDrab', 'peru', 'darkBlue', 'wheat', 'blanchedAlmond', 'maroon', 'midnightBlue', 'darkGray', 'grey', 'antiqueWhite', 'darkOrange', 'dkGreen', 'goldenrod', 'orchid', 'navy', 'ltGray', 'ltSkyBlue', 'ltSteelBlue', 'medSlateBlue', 'navajoWhite', 'violet', 'gold', 'dkSlateGrey', 'dkTurquoise', 'paleGoldenrod', 'dkGray', 'medPurple', 'mediumPurple', 'darkGreen', 'darkSeaGreen', 'saddleBrown', 'dkRed', 'skyBlue', 'teal', 'ghostWhite', 'mediumVioletRed', 'ltSlateGray', 'cornsilk', 'seaGreen', 'silver', 'honeydew', 'ltGreen', 'dkSeaGreen', 'deepPink', 'medAquamarine', 'dkMagenta', 'lightCoral', 'medBlue', 'medOrchid', 'darkSlateGray', 'aqua', 'beige', 'ltSeaGreen', 'lemonChiffon', 'orange', 'whiteSmoke', 'blue', 'lightGoldenrodYellow', 'cyan', 'dkCyan', 'indigo', 'chocolate', 'lightSalmon', 'coral', 'darkSalmon', 'dkGrey', 'sienna', 'dkSalmon', 'papayaWhip', 'darkCyan', 'thistle', 'khaki', 'lightPink', 'dimGrey', 'ltGrey', 'cornflowerBlue', 'ltSlateGrey', 'purple', 'orangeRed', 'ivory', 'dkOrchid', 'floralWhite', 'linen', 'rosyBrown', 'gainsboro', 'olive', 'hotPink', 'lightGreen', 'dkBlue', 'dodgerBlue', 'darkRed', 'blueViolet', 'darkSlateGrey', 'ltGoldenrodYellow', 'mediumOrchid', 'burlyWood', 'ltYellow', 'lawnGreen', 'azure', 'limeGreen', 'lightYellow', 'dkSlateBlue', 'ltBlue', 'slateBlue', 'mediumAquamarine', 'tan', 'green', 'slateGrey', 'lightGray', 'medVioletRed', 'dkSlateGray', 'lavender', 'darkKhaki', 'cadetBlue', 'mediumSeaGreen', 'darkOliveGreen', 'paleGreen', 'ltCoral', 'mediumBlue', 'sandyBrown', 'paleTurquoise', 'mediumSpringGreen', 'brown', 'fuchsia', 'deepSkyBlue', 'plum', 'seaShell', 'ltCyan', 'forestGreen', 'slateGray', 'lightGrey', 'chartreuse', 'aliceBlue', 'lime', 'mediumTurquoise', 'darkMagenta', 'medSpringGreen', 'yellowGreen', 'powderBlue', 'turquoise', 'white'}

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

schemeClr
Value must be one of {'accent3', 'phClr', 'accent5', 'hlink', 'dk1', 'accent6', 'bg1', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'bg2', 'folHlink', 'tx1', 'tx2'}

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'>

class `openpyxl.drawing.effect.`**`PresetShadowEffect`**(*prst=None*, *dist=None*, *dir=None*, ***kw*)

Bases: `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 {'shdw7', 'shdw19', 'shdw1', 'shdw4', 'shdw20', 'shdw10', 'shdw2', 'shdw15', 'shdw11', 'shdw14', 'shdw13', 'shdw12', 'shdw17', 'shdw8', 'shdw3', 'shdw5', 'shdw6', 'shdw9', 'shdw16', 'shdw18'}

prstClr
Value must be one of {'lightSlateGray', 'oldLace', 'ltPink', 'lightSteelBlue', 'peachPuff', 'steelBlue', 'dkViolet', 'greenYellow', 'darkGrey', 'moccasin', 'snow', 'yellow', 'firebrick', 'medTurquoise', 'gray', 'lightSkyBlue', 'darkOrchid', 'medSeaGreen', 'salmon', 'mistyRose', 'black', 'aquamarine', 'dkOrange', 'mintCream', 'red', 'magenta', 'ltSalmon', 'indianRed', 'dkGoldenrod', 'lightSeaGreen', 'paleVioletRed', 'royalBlue', 'darkSlateBlue', 'pink', 'crimson', 'darkGoldenrod', 'darkTurquoise', 'dimGray', 'tomato', 'dkOliveGreen', 'springGreen', 'dkKhaki', 'mediumSlateBlue', 'lightBlue', 'lavenderBlush', 'darkViolet', 'lightCyan', 'bisque', 'lightSlateGrey', 'oliveDrab', 'peru', 'darkBlue', 'wheat', 'blanchedAlmond', 'maroon', 'midnightBlue', 'darkGray', 'grey', 'antiqueWhite', 'darkOrange', 'dkGreen', 'goldenrod', 'orchid', 'navy', 'ltGray', 'ltSkyBlue', 'ltSteelBlue', 'medSlateBlue', 'navajoWhite', 'violet', 'gold', 'dkSlateGrey', 'dkTurquoise', 'paleGoldenrod', 'dkGray', 'medPurple', 'mediumPurple', 'darkGreen', 'darkSeaGreen', 'saddleBrown', 'dkRed', 'skyBlue', 'teal', 'ghostWhite', 'mediumVioletRed', 'ltSlateGray', 'cornsilk', 'seaGreen', 'silver', 'honeydew', 'ltGreen', 'dkSeaGreen', 'deepPink', 'medAquamarine', 'dkMagenta', 'lightCoral', 'medBlue', 'medOrchid', 'darkSlateGray', 'aqua', 'beige', 'ltSeaGreen', 'lemonChiffon', 'orange', 'whiteSmoke', 'blue', 'lightGoldenrodYellow', 'cyan', 'dkCyan', 'indigo', 'chocolate', 'lightSalmon', 'coral', 'darkSalmon', 'dkGrey', 'sienna', 'dkSalmon', 'papayaWhip', 'darkCyan', 'thistle', 'khaki', 'lightPink', 'dimGrey', 'ltGrey', 'cornflowerBlue', 'ltSlateGrey', 'purple', 'orangeRed', 'ivory', 'dkOrchid', 'floralWhite', 'linen', 'rosyBrown', 'gainsboro', 'olive', 'hotPink', 'lightGreen', 'dkBlue', 'dodgerBlue', 'darkRed', 'blueViolet', 'darkSlateGrey', 'ltGoldenrodYellow', 'mediumOrchid', 'burlyWood', 'ltYellow', 'lawnGreen', 'azure', 'limeGreen', 'lightYellow', 'dkSlateBlue', 'ltBlue', 'slateBlue', 'mediumAquamarine', 'tan', 'green', 'slateGrey', 'lightGray', 'medVioletRed', 'dkSlateGray', 'lavender', 'darkKhaki', 'cadetBlue', 'mediumSeaGreen', 'darkOliveGreen', 'paleGreen', 'ltCoral', 'mediumBlue', 'sandyBrown', 'paleTurquoise', 'mediumSpringGreen', 'brown', 'fuchsia', 'deepSkyBlue', 'plum', 'seaShell', 'ltCyan', 'forestGreen', 'slateGray', 'lightGrey', 'chartreuse', 'aliceBlue', 'lime', 'mediumTurquoise', 'darkMagenta', 'medSpringGreen', 'yellowGreen', 'powderBlue', 'turquoise', 'white'}

schemeClr
Value must be one of {'accent3', 'phClr', 'accent5', 'hlink', 'dk1', 'accent6', 'bg1', 'lt2', 'accent2', 'accent1', 'dk2', 'lt1', 'accent4', 'bg2', 'folHlink', 'tx1', 'tx2'}

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*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

algn
Value must be one of {'br', 'r', 'ctr', 'l', 'b', 'tl', 'bl', 'tr', '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*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

class openpyxl.drawing.effect.**TintEffect**(*hue=None*, *amt=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

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

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*)

Bases: *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 {'print', 'screen', 'hqprint', 'email'}

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>*, *srcRect=None*)
Bases: *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=None*, *lin=None*, *path=None*, *tileRect=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

flip
Value must be one of {‘y’, ‘xy’, ‘x’}

gsLst
Values must be of type <class ‘openpyxl.drawing.fill.GradientStopList’>

lin
Values must be of type <class ‘openpyxl.drawing.fill.LinearShadeProperties’>

path
Values must be of type <class ‘openpyxl.drawing.fill.PathShadeProperties’>

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

tagname = ‘gradFill’

tileRect
Values must be of type <class ‘openpyxl.drawing.fill.RelativeRect’>

class openpyxl.drawing.fill.**GradientStop**(*pos=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

tagname = ‘gradStop’

class openpyxl.drawing.fill.**GradientStopList**(*gs=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

tagname = ‘gradStopLst’

class openpyxl.drawing.fill.**LinearShadeProperties**(*ang=None*, *scaled=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

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

class openpyxl.drawing.fill.**PathShadeProperties**(*path=None*, *fillToRect=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

fillToRect
Values must be of type <class ‘openpyxl.drawing.fill.RelativeRect’>

path
Value must be one of {‘shape’, ‘rect’, ‘circle’}

class openpyxl.drawing.fill.**PatternFillProperties**(*prst=None*, *fgClr=None*, *bg-Clr=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

bgClr
Values must be of type <class ‘openpyxl.drawing.colors.ColorChoice’>

fgClr
Values must be of type <class ‘openpyxl.drawing.colors.ColorChoice’>

namespace = ‘http://schemas.openxmlformats.org/drawingml/2006/main’

prst
Value must be one of {'ltHorz', 'trellis', 'lgCheck', 'pct60', 'diagBrick', 'dkVert', 'pct80', 'narVert', 'dotDmnd', 'weave', 'dashVert', 'pct50', 'plaid', 'dashUpDiag', 'pct10', 'narHorz', 'dkDnDiag', 'pct20', 'wdUpDiag', 'solidDmnd', 'openDmnd', 'pct40', 'lgConfetti', 'pct5', 'dkHorz', 'pct70', 'smGrid', 'dashHorz', 'wdDnDiag', 'smCheck', 'pct90', 'wave', 'divot', 'horz', 'pct30', 'cross', 'zigZag', 'dashDnDiag', 'dotGrid', 'sphere', 'pct75', 'ltUpDiag', 'ltVert', 'dnDiag', 'dkUpDiag', 'horzBrick', 'ltDnDiag', 'smConfetti', 'shingle', 'upDiag', 'lgGrid', 'diagCross', 'vert', 'pct25'}

tagname = 'pattFill'

class openpyxl.drawing.fill.**RelativeRect**(*l=None*, *t=None*, *r=None*, *b=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

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

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

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

t
Values must be of type <class 'float'>

tagname = 'rect'

class openpyxl.drawing.fill.**StretchInfoProperties**(*fillRect=<openpyxl.drawing.fill.RelativeRect object>*)
Bases: *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*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

algn
Value must be one of {'br', 'r', 'ctr', 'l', 'b', 'tl', 'bl', 'tr', '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.graphic module

class `openpyxl.drawing.graphic.`**`ChartRelation`**(*id*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

`namespace` **= 'http://schemas.openxmlformats.org/drawingml/2006/chart'**

`tagname` **= 'chart'**

class `openpyxl.drawing.graphic.`**`Connection`**(*id=None*, *idx=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`id`
Values must be of type <class 'int'>

`idx`
Values must be of type <class 'int'>

class `openpyxl.drawing.graphic.`**`Connector`**(*macro=None*, *fPublished=None*, *nvCxnSpPr=None*, *spPr=None*, *style=None*)

Bases: *`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.graphic.ConnectorNonVisual'>

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

`style`
Values must be of type <class 'openpyxl.drawing.shapes.ShapeStyle'>

class `openpyxl.drawing.graphic.`**`ConnectorLocking`**(*extLst=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

class `openpyxl.drawing.graphic.`**`ConnectorNonVisual`**(*cNvPr=None*, *cNvCxnSpPr=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`cNvCxnSpPr`
Values must be of type <class 'openpyxl.drawing.graphic.NonVisualConnectorProperties'>

`cNvPr`
Values must be of type <class 'openpyxl.drawing.graphic.NonVisualDrawingProps'>

class `openpyxl.drawing.graphic.`**`GraphicData`**(*uri='http://schemas.openxmlformats.org/drawingml/2006/chart'*, *chart=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`chart`
Values must be of type <class 'openpyxl.drawing.graphic.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*)
Bases: *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.shapes.Transform2D'>

class openpyxl.drawing.graphic.**GraphicFrameLocking**(*noGrp=None*, *noDrilldown=None*, *noSelect=None*, *noChangeAspect=None*, *noMove=None*, *noResize=None*, *extLst=None*)
Bases: *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*)
Bases: *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.`**`GroupLocking`**(*noGrp=None*, *noUngrp=None*, *noSelect=None*, *noRot=None*, *noChangeAspect=None*, *noMove=None*, *noResize=None*, *extLst=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

`noChangeAspect`
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'>

class `openpyxl.drawing.graphic.`**`GroupShape`**(*nvGrpSpPr=None*, *grpSpPr=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`grpSpPr`
Values must be of type <class 'openpyxl.drawing.graphic.GroupShapeProperties'>

`nvGrpSpPr`
Values must be of type <class 'openpyxl.drawing.graphic.NonVisualGroupShape'>

class `openpyxl.drawing.graphic.`**`GroupShapeProperties`**(*bwMode=None*, *xfrm=None*, *scene3d=None*, *extLst=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`bwMode`
Value must be one of {'gray', 'blackWhite', 'hidden', 'auto', 'grayWhite', 'ltGray', 'white', 'clr', 'invGray', 'blackGray', 'black'}

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

`scene3d`
Values must be of type <class 'openpyxl.drawing.shapes.Scene3D'>

`xfrm`
Values must be of type <class 'openpyxl.drawing.graphic.GroupTransform2D'>

class `openpyxl.drawing.graphic.`**`GroupTransform2D`**(*rot=None*, *flipH=None*, *flipV=None*, *off=None*, *ext=None*, *chOff=None*, *chExt=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`chExt`
Values must be of type <class 'openpyxl.drawing.shapes.PositiveSize2D'>

chOff
Values must be of type <class 'openpyxl.drawing.shapes.Point2D'>

ext
Values must be of type <class 'openpyxl.drawing.shapes.PositiveSize2D'>

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

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

off
Values must be of type <class 'openpyxl.drawing.shapes.Point2D'>

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

class `openpyxl.drawing.graphic.`**`NonVisualConnectorProperties`**(*cxnSpLocks=None, stCxn=None, endCxn=None, extLst=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

cxnSpLocks
Values must be of type <class 'openpyxl.drawing.graphic.ConnectorLocking'>

endCxn
Values must be of type <class 'openpyxl.drawing.graphic.Connection'>

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

stCxn
Values must be of type <class 'openpyxl.drawing.graphic.Connection'>

class `openpyxl.drawing.graphic.`**`NonVisualDrawingProps`**(*id=None, name=None, descr=None, hidden=None, title=None, hlinkClick=None, hlinkHover=None, extLst=None*)

Bases: *`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.graphic.`**`NonVisualGraphicFrame`**(*cNvPr=None*, *cNvGraphicFramePr=None*)
Bases: `openpyxl.descriptors.serialisable.Serialisable`

cNvGraphicFramePr
Values must be of type <class 'openpyxl.drawing.graphic.NonVisualGraphicFrameProperties'>

cNvPr
Values must be of type <class 'openpyxl.drawing.graphic.NonVisualDrawingProps'>

tagname = 'nvGraphicFramePr'

class `openpyxl.drawing.graphic.`**`NonVisualGraphicFrameProperties`**(*graphicFrameLocks=None*, *extLst=None*)
Bases: `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'

class `openpyxl.drawing.graphic.`**`NonVisualGroupDrawingShapeProps`**(*grpSpLocks=None*, *extLst=None*)
Bases: `openpyxl.descriptors.serialisable.Serialisable`

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

grpSpLocks
Values must be of type <class 'openpyxl.drawing.graphic.GroupLocking'>

class `openpyxl.drawing.graphic.`**`NonVisualGroupShape`**(*cNvPr=None*, *cNvGrpSpPr=None*)
Bases: `openpyxl.descriptors.serialisable.Serialisable`

cNvGrpSpPr
Values must be of type <class 'openpyxl.drawing.graphic.NonVisualGroupDrawingShapeProps'>

cNvPr
Values must be of type <class 'openpyxl.drawing.graphic.NonVisualDrawingProps'>

class `openpyxl.drawing.graphic.`**`NonVisualPictureProperties`**(*preferRelativeResize=None*, *picLocks=None*, *extLst=None*)
Bases: `openpyxl.descriptors.serialisable.Serialisable`

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

picLocks
Values must be of type <class 'openpyxl.drawing.graphic.PictureLocking'>

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

tagname = 'cNvPicPr'

class `openpyxl.drawing.graphic.`**`PictureFrame`**(*macro=None*, *fPublished=None*, *nvPicPr=None*, *blipFill=None*, *spPr=None*, *style=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`blipFill`
Values must be of type <class 'openpyxl.drawing.fill.BlipFillProperties'>

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

`macro`
Values must be of type <class 'str'>

`nvPicPr`
Values must be of type <class 'openpyxl.drawing.graphic.PictureNonVisual'>

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

`style`
Values must be of type <class 'openpyxl.drawing.shapes.ShapeStyle'>

`tagname` = 'pic'

class `openpyxl.drawing.graphic.`**`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*)

Bases: *`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.graphic.**PictureNonVisual**(*cNvPr=None*, *cNvPicPr=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

cNvPicPr
Values must be of type <class 'openpyxl.drawing.graphic.NonVisualPictureProperties'>

cNvPr
Values must be of type <class 'openpyxl.drawing.graphic.NonVisualDrawingProps'>

tagname = 'nvPicPr'

openpyxl.drawing.image module

class openpyxl.drawing.image.**Image**(*img*, *coordinates=((0, 0), (1, 1))*, *size=(None, None)*, *nochangeaspect=True*, *nochangearrowheads=True*)
Bases: object

Raw Image class

anchor(*cell*, *anchortype='absolute'*)
anchors the image to the given cell optional parameter anchortype supports 'absolute' or 'oneCell'

openpyxl.drawing.image.**bounding_box**(*bw*, *bh*, *w*, *h*)
Returns a tuple (new_width, new_height) which has the property that it fits within box_width and box_height and has (close to) the same aspect ratio as the original size

openpyxl.drawing.line module

class openpyxl.drawing.line.**DashStop**(*d=0*, *sp=0*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

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

tagname = 'ds'

class openpyxl.drawing.line.**DashStopList**(*ds=None*)
Bases: *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*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

len
Value must be one of {'lg', 'med', 'sm'}

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

tagname = 'end'

type
Value must be one of {'stealth', 'arrow', 'none', 'diamond', 'triangle', 'oval'}

w
Value must be one of {'lg', 'med', 'sm'}

class `openpyxl.drawing.line.`**`LineJoinMiterProperties`**(*lim=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

tagname = 'miter'

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, mitre=None, headEnd=None, tailEnd=None, extLst=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

algn
Value must be one of {'in', 'ctr'}

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

cap
Value must be one of {'sq', 'flat', 'rnd'}

cmpd
Value must be one of {'thickThin', 'tri', 'thinThick', 'sng', 'dbl'}

custDash
Values must be of type <class 'openpyxl.drawing.line.DashStop'>

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 'openpyxl.drawing.line.LineJoinMiterProperties'>

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 {'solid', 'sysDashDot', 'lgDash', 'lgDashDot', 'lgDashDotDot', 'sysDashDotDot', 'sysDash', 'dashDot', 'dash', 'sysDot', 'dot'}

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'>

openpyxl.drawing.shape module

class openpyxl.drawing.shape.**Shape**(*chart*, *coordinates=((0, 0), (1, 1))*, *text=None*, *scheme='accent1'*)
Bases: object

a drawing inside a chart coordiantes are specified by the user in the axis units

FONT_HEIGHT = 8

FONT_WIDTH = 7

MARGIN_BOTTOM = 28

MARGIN_LEFT = 20

RECT = 'rect'
"line" "lineInv" "triangle" "rtTriangle" "diamond" "parallelogram" "trapezoid" "nonIsoscelesTrapezoid" "pentagon" "hexagon" "heptagon" "octagon" "decagon" "dodecagon" "star4" "star5" "star6" "star7" "star8" "star10" "star12" "star16" "star24" "star32" "roundRect" "round1Rect" "round2SameRect" "round2DiagRect" "snipRoundRect" "snip1Rect" "snip2SameRect" "snip2DiagRect" "plaque" "ellipse" "teardrop" "homePlate" "chevron" "pieWedge" "pie" "blockArc" "donut" "noSmoking" "rightArrow" "leftArrow" "upArrow" "downArrow" "stripedRightArrow" "notchedRightArrow" "bentUpArrow" "leftRightArrow" "upDownArrow" "leftUpArrow" "leftRightUpArrow" "quadArrow" "leftArrowCallout" "rightArrowCallout" "upArrowCallout" "downArrowCallout" "leftRightArrowCallout" "upDownArrowCallout" "quadArrowCallout" "bentArrow" "uturnArrow" "circularArrow" "leftCircularArrow" "leftRightCircularArrow" "curvedRightArrow" "curvedLeftArrow" "curvedUpArrow" "curvedDownArrow" "swooshArrow" "cube" "can" "lightningBolt" "heart" "sun" "moon" "smileyFace" "irregularSeal1" "irregularSeal2" "foldedCorner" "bevel" "frame" "halfFrame" "corner" "diagStripe" "chord" "arc" "leftBracket" "rightBracket" "leftBrace" "rightBrace" "bracketPair" "bracePair" "straightConnector1" "bentConnector2" "bentConnector3" "bentConnector4" "bentConnector5" "curvedConnector2" "curvedConnector3" "curvedConnector4" "curvedConnector5" "callout1" "callout2" "callout3" "accentCallout1" "accentCallout2" "accentCallout3" "borderCallout1" "borderCallout2" "borderCallout3" "accentBorderCallout1" "accentBorderCallout2" "accentBorderCallout3" "wedgeRectCallout" "wedgeRoundRectCallout" "wedgeEllipseCallout" "cloudCallout" "cloud" "ribbon" "ribbon2" "ellipseRibbon" "ellipseRibbon2" "leftRightRibbon" "verticalScroll" "horizontalScroll" "wave" "doubleWave" "plus" "flowChartProcess" "flowChartDecision" "flowChartInputOutput" "flowChartPredefinedProcess" "flowChartInternalStorage" "flowChartDocument" "flowChartMultidocument" "flowChartTerminator" "flowChartPreparation" "flowChartManualInput" "flowChartManualOperation" "flowChartConnector" "flowChartPunchedCard" "flowChartPunchedTape" "flowChartSummingJunction" "flowChartOr" "flowChartCollate" "flowChartSort" "flowChartExtract" "flowChartMerge" "flowChartOfflineStorage" "flowChartOnlineStorage" "flowChartMagneticTape" "flowChartMagneticDisk" "flowChartMagneticDrum" "flowChartDisplay" "flowChartDelay" "flowChartAlternateProcess" "flowChartOffpageConnector" "actionButtonBlank" "actionButtonHome" "actionButtonHelp" "actionButtonInformation" "actionButtonForwardNext" "actionButtonBackPrevious" "actionButtonEnd" "actionButtonBeginning" "actionButtonReturn" "actionButtonDocument" "actionButtonSound" "actionButtonMovie" "gear6" "gear9"

“funnel” “mathPlus” “mathMinus” “mathMultiply” “mathDivide” “mathEqual” “mathNotEqual” “cornerTabs” “squareTabs” “plaqueTabs” “chartX” “chartStar” “chartPlus”

ROUND_RECT = ‘roundRect’

border_color

border_width

color

coordinates

Return coordindates in axis units

text_color

class openpyxl.drawing.shape.**ShapeWriter**(*shapes*)

Bases: object

one file per shape

write(*shape_id*)

openpyxl.drawing.shapes module

class openpyxl.drawing.shapes.**AdjPoint2D**(*x=None*, *y=None*)

Bases: *openpyxl.descriptors.serialisable.Serialisable*

x

Values must be of type <class ‘int’>

y

Values must be of type <class ‘int’>

class openpyxl.drawing.shapes.**AdjustHandleList**

Bases: *openpyxl.descriptors.serialisable.Serialisable*

class openpyxl.drawing.shapes.**Backdrop**(*anchor=None*, *norm=None*, *up=None*, *extLst=None*)

Bases: *openpyxl.descriptors.serialisable.Serialisable*

anchor

Values must be of type <class ‘openpyxl.drawing.shapes.Point3D’>

extLst

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

norm

Values must be of type <class ‘openpyxl.drawing.shapes.Vector3D’>

up

Values must be of type <class ‘openpyxl.drawing.shapes.Vector3D’>

class openpyxl.drawing.shapes.**Bevel**(*w=None*, *h=None*, *prst=None*)

Bases: *openpyxl.descriptors.serialisable.Serialisable*

h

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

prst

Values must be of type <openpyxl.descriptors.base.Set object at 0x7fd32b6e5898>

w

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

class openpyxl.drawing.shapes.**Camera**(*prst=None*, *fov=None*, *zoom=None*, *rot=None*)

Bases: *openpyxl.descriptors.serialisable.Serialisable*

fov
Values must be of type <class 'openpyxl.descriptors.base.Integer'>

prst
Values must be of type <openpyxl.descriptors.base.Set object at 0x7fd32b6e5080>

rot
Values must be of type <class 'openpyxl.drawing.shapes.SphereCoords'>

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

class openpyxl.drawing.shapes.**ConnectionSite**(*ang=None*, *pos=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

pos
Values must be of type <class 'openpyxl.drawing.shapes.AdjPoint2D'>

class openpyxl.drawing.shapes.**ConnectionSiteList**(*cxn=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

cxn
Values must be of type <class 'openpyxl.drawing.shapes.ConnectionSite'>

class openpyxl.drawing.shapes.**CustomGeometry2D**(*avLst=None*, *gdLst=None*, *ahLst=None*, *cxnLst=None*, *rect=None*, *pathLst=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

ahLst
Values must be of type <class 'openpyxl.drawing.shapes.AdjustHandleList'>

avLst
Values must be of type <class 'openpyxl.drawing.shapes.GeomGuideList'>

cxnLst
Values must be of type <class 'openpyxl.drawing.shapes.ConnectionSiteList'>

gdLst
Values must be of type <class 'openpyxl.drawing.shapes.GeomGuideList'>

pathLst
Values must be of type <class 'openpyxl.drawing.shapes.Path2DList'>

rect
Values must be of type <class 'openpyxl.drawing.shapes.GeomRect'>

class openpyxl.drawing.shapes.**FontReference**(*idx=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

class openpyxl.drawing.shapes.**GeomGuide**(*name=None*, *fmla=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

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

class `openpyxl.drawing.shapes.`**`GeomGuideList`**(*gd=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`gd`
Values must be of type <class 'openpyxl.drawing.shapes.GeomGuide'>

class `openpyxl.drawing.shapes.`**`GeomRect`**(*l=None*, *t=None*, *r=None*, *b=None*)
Bases: *`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.shapes.`**`LightRig`**(*rig=None*, *dir=None*, *rot=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`dir`
Values must be of type <openpyxl.descriptors.base.Set object at 0x7fd32b6e52b0>

`rig`
Values must be of type <openpyxl.descriptors.base.Set object at 0x7fd32b6e51d0>

`rot`
Values must be of type <class 'openpyxl.drawing.shapes.SphereCoords'>

class `openpyxl.drawing.shapes.`**`Path2D`**(*w=None*, *h=None*, *fill=None*, *stroke=None*, *extrusionOk=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

`fill`
Value must be one of {'darkenLess', 'norm', 'lightenLess', 'lighten', '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.shapes.`**`Path2DList`**(*path=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`path`
Values must be of type <class 'openpyxl.drawing.shapes.Path2D'>

class `openpyxl.drawing.shapes.`**`Point2D`**(*x=None*, *y=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`x`
Values must be of type <class 'int'>

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

class openpyxl.drawing.shapes.**Point3D**(*x=None*, *y=None*, *z=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

x
Values must be of type <class 'openpyxl.descriptors.base.Integer'>

y
Values must be of type <class 'openpyxl.descriptors.base.Integer'>

z
Values must be of type <class 'openpyxl.descriptors.base.Integer'>

class openpyxl.drawing.shapes.**PositiveSize2D**(*cx=None*, *cy=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

Dimensions in EMUs

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

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

class openpyxl.drawing.shapes.**PresetGeometry2D**(*prst=None*, *avLst=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

avLst
Values must be of type <class 'openpyxl.drawing.shapes.GeomGuideList'>

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

prst
Value must be one of {'flowChartOnlineStorage', 'gear6', 'bentUpArrow', 'bentConnector3', 'homePlate', 'callout2', 'flowChartManualOperation', 'borderCallout3', 'actionButtonBackPrevious', 'star8', 'round2SameRect', 'accentCallout3', 'leftUpArrow', 'decagon', 'corner', 'flowChartInputOutput', 'mathNotEqual', 'diamond', 'borderCallout1', 'actionButtonMovie', 'horizontalScroll', 'flowChartManualInput', 'leftArrowCallout', 'callout3', 'star5', 'accentCallout1', 'star10', 'flowChartMagneticDrum', 'actionButtonReturn', 'actionButtonSound', 'flowChartMultidocument', 'actionButtonHome', 'actionButtonEnd', 'mathEqual', 'stripedRightArrow', 'callout1', 'mathMultiply', 'flowChartInternalStorage', 'flowChartPunchedCard', 'ribbon2', 'actionButtonBlank', 'quadArrowCallout', 'bracePair', 'flowChartOfflineStorage', 'blockArc', 'curvedUpArrow', 'foldedCorner', 'heptagon', 'uturnArrow', 'hexagon', 'roundRect', 'flowChartPreparation', 'trapezoid', 'flowChartMagneticTape', 'donut', 'lightningBolt', 'star4', 'ellipseRibbon', 'irregularSeal1', 'octagon', 'triangle', 'doubleWave', 'noSmoking', 'mathPlus', 'wedgeRectCallout', 'accentBorderCallout2', 'upArrowCallout', 'pentagon', 'plaque', 'ellipse', 'borderCallout2', 'cornerTabs', 'pie', 'quadArrow', 'flowChartDocument', 'notchedRightArrow', 'teardrop', 'snip2DiagRect', 'star6', 'actionButtonBeginning', 'leftRightArrow', 'curvedRightArrow', 'accentCallout2', 'leftRightCircularArrow', 'leftRightUpArrow', 'round2DiagRect', 'moon', 'actionButtonDocument', 'parallelogram', 'cloudCallout', 'flowChartExtract', 'curvedConnector3', 'flowChartMagneticDisk', 'lineInv', 'irregularSeal2', 'curvedConnector4', 'line', 'flowChartSort', 'leftRightRibbon', 'diagStripe', 'rtTriangle', 'rect', 'star12', 'star16', 'flowChartOffpageConnector', 'chord', 'halfFrame', 'wedgeRoundRectCallout', 'squareTabs', 'rightArrowCallout', 'gear9', 'upDownArrowCallout', 'ellipseRibbon2', 'snip2SameRect', 'flowChartPunchedTape', 'actionButtonInformation', 'flowChartProcess', 'accentBorderCallout3', 'flowChartCollate', 'upDownArrow', 'rightArrow', 'circularArrow', 'flowChartMerge', 'bevel', 'wave', 'flowChartAlternateProcess', 'smileyFace', 'flowChartConnector', 'flowChartPredefinedProcess', 'curvedConnector5', 'bentArrow', 'curvedConnector2', 'can', 'flowChartDisplay', 'mathMinus', 'nonIsoscelesTrapezoid', 'mathDivide', 'arc', 'bentConnector4', 'snip1Rect',

'downArrow', 'star7', 'rightBrace', 'accentBorderCallout1', 'rightBracket', 'flowChartDelay', 'leftBracket', 'chartX', 'bentConnector5', 'actionButtonForwardNext', 'cube', 'curvedLeftArrow', 'sun', 'leftArrow', 'straightConnector1', 'leftCircularArrow', 'frame', 'chartPlus', 'dodecagon', 'flowChartDecision', 'actionButtonHelp', 'snipRoundRect', 'star24', 'flowChartOr', 'funnel', 'curvedDownArrow', 'leftRightArrowCallout', 'swooshArrow', 'pieWedge', 'leftBrace', 'plaqueTabs', 'round1Rect', 'heart', 'plus', 'chevron', 'flowChartTerminator', 'chartStar', 'downArrowCallout', 'bracketPair', 'upArrow', 'verticalScroll', 'flowChartSummingJunction', 'star32', 'wedgeEllipseCallout', 'bentConnector2', 'cloud', 'ribbon'}

class `openpyxl.drawing.shapes.`**`Scene3D`**(*camera=None*, *lightRig=None*, *backdrop=None*, *extLst=None*)

Bases: `openpyxl.descriptors.serialisable.Serialisable`

`backdrop`
Values must be of type <class 'openpyxl.drawing.shapes.Backdrop'>

`camera`
Values must be of type <class 'openpyxl.drawing.shapes.Camera'>

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

`lightRig`
Values must be of type <class 'openpyxl.drawing.shapes.LightRig'>

class `openpyxl.drawing.shapes.`**`Shape3D`**(*z=None*, *extrusionH=None*, *contourW=None*, *prstMaterial=None*, *bevelT=None*, *bevelB=None*, *extrusionClr=None*, *contourClr=None*, *extLst=None*)

Bases: `openpyxl.descriptors.serialisable.Serialisable`

`bevelB`
Values must be of type <class 'openpyxl.drawing.shapes.Bevel'>

`bevelT`
Values must be of type <class 'openpyxl.drawing.shapes.Bevel'>

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

`contourW`
Values must be of type 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 Values must be of type <class 'int'>

`prstMaterial`
Values must be of type <openpyxl.descriptors.base.Set object at 0x7fd32b6e5a58>

`z`
Values must be of type <class 'openpyxl.descriptors.base.Integer'>

class `openpyxl.drawing.shapes.`**`ShapeStyle`**(*lnRef=None*, *fillRef=None*, *effectRef=None*, *fontRef=None*)

Bases: `openpyxl.descriptors.serialisable.Serialisable`

`effectRef`
Values must be of type <class 'openpyxl.drawing.shapes.StyleMatrixReference'>

fillRef
Values must be of type <class 'openpyxl.drawing.shapes.StyleMatrixReference'>

fontRef
Values must be of type <class 'openpyxl.drawing.shapes.FontReference'>

lnRef
Values must be of type <class 'openpyxl.drawing.shapes.StyleMatrixReference'>

class `openpyxl.drawing.shapes.`**`SphereCoords`**(*lat=None*, *lon=None*, *rev=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

lat
Values must be of type <class 'openpyxl.descriptors.base.Integer'>

lon
Values must be of type <class 'openpyxl.descriptors.base.Integer'>

rev
Values must be of type <class 'openpyxl.descriptors.base.Integer'>

class `openpyxl.drawing.shapes.`**`StyleMatrixReference`**(*idx=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

class `openpyxl.drawing.shapes.`**`Transform2D`**(*rot=None*, *flipH=None*, *flipV=None*, *off=None*, *ext=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

ext
Values must be of type <class 'openpyxl.drawing.shapes.PositiveSize2D'>

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

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

off
Values must be of type <class 'openpyxl.drawing.shapes.Point2D'>

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

tagname = 'xfrm'

class `openpyxl.drawing.shapes.`**`Vector3D`**(*dx=None*, *dy=None*, *dz=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

dx
Values must be of type <class 'openpyxl.descriptors.base.Integer'>

dy
Values must be of type <class 'openpyxl.descriptors.base.Integer'>

dz
Values must be of type <class 'openpyxl.descriptors.base.Integer'>

openpyxl.drawing.spreadsheet_drawing module

class `openpyxl.drawing.spreadsheet_drawing.`**`AbsoluteAnchor`**(*pos=None*, *ext=None*, ***kw*)

Bases: `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.graphic.Connector'>

`ext`
Values must be of type <class 'openpyxl.drawing.shapes.PositiveSize2D'>

`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.graphic.PictureFrame'>

`pos`
Values must be of type <class 'openpyxl.drawing.shapes.Point2D'>

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

`tagname` **= 'absoluteAnchor'**

class `openpyxl.drawing.spreadsheet_drawing.`**`AnchorClientData`**(*fLocksWithSheet=None*, *fPrintsWithSheet=None*)

Bases: *`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*)

Bases: *`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*)

Bases: `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.graphic.Connector'>

ext
Values must be of type <class 'openpyxl.drawing.shapes.PositiveSize2D'>

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.graphic.PictureFrame'>

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

tagname = 'oneCellAnchor'

class `openpyxl.drawing.spreadsheet_drawing.`**`SpreadsheetDrawing`**(*twoCellAnchor=()*, *oneCellAnchor=()*, *absoluteAnchor=()*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

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

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*)
Bases: `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.graphic.Connector'>

editAs
Value must be one of {'oneCell', 'twoCell', 'absolute'}

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.graphic.PictureFrame'>

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

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*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

type
Value must be one of {'arabicDbPlain', 'hindiAlpha1Period', 'ea1JpnKorPeriod', 'ea1ChsPlain', 'circleNumWdWhitePlain', 'thaiAlphaParenR', 'circleNumWdBlackPlain', 'ea1ChtPlain', 'thaiAlphaParenBoth', 'alphaUcParenR', 'hebrew2Minus', 'romanUcPeriod', 'arabicDbPeriod', 'hindiAlphaPeriod', 'arabicPlain', 'circleNumDbPlain', 'ea1JpnChsDbPeriod', 'thaiNumParenBoth', 'arabic2Minus', 'thaiAlphaPeriod', 'romanLcPeriod', 'arabicParenR', 'alphaLcParenR', 'romanUcParenR', 'ea1ChsPeriod', 'arabicParenBoth', 'alphaLcPeriod', 'romanLcParenR', 'alphaUcParenBoth', 'ea1ChtPeriod', 'thaiNumParenR', 'romanLcParenBoth', 'arabic1Minus', 'alphaLcParenBoth', 'romanUcParenBoth', 'alphaUcPeriod', 'arabicPeriod', 'thaiNumPeriod', 'hindiNumPeriod', 'hindiNumParenR', 'ea1JpnKorPlain'}

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*)
Bases: *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 {'all', 'small'}

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 {'noStrike', 'dblStrike', 'sngStrike'}

sym
Values must be of type <class 'openpyxl.drawing.text.Font'>

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

tagname = 'defRPr'

u
Value must be one of {'wavy', 'wavyHeavy', 'words', 'wavyDbl', 'dotDotDashHeavy', 'dottedHeavy', 'heavy', 'dashLong', 'dashLongHeavy', 'dotDashHeavy', 'dashHeavy', 'dotted', 'dotDotDash', 'dash', 'sng', 'dotDash', 'dbl'}

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*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

name
Values must be of type <class 'openpyxl.descriptors.base.String'>

class openpyxl.drawing.text.**Font**(*typeface=None*, *panose=None*, *pitchFamily=None*, *charset=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

charset
Values must be of type <class 'openpyxl.descriptors.base.MinMax'>

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

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

pitchFamily
Values must be of type <class 'openpyxl.descriptors.base.MinMax'>

tagname = 'latin'

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

class openpyxl.drawing.text.**GeomGuide**(*name=None*, *fmla=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

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

class openpyxl.drawing.text.**GeomGuideList**(*gd=None*)
Bases: *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*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

action
Values must be of type <class 'openpyxl.descriptors.base.String'>

endSnd
Values must be of type <class 'openpyxl.descriptors.base.Bool'>

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

highlightClick
Values must be of type <class 'openpyxl.descriptors.base.Bool'>

history
Values must be of type <class 'openpyxl.descriptors.base.Bool'>

invalidUrl
Values must be of type <class 'openpyxl.descriptors.base.String'>

snd
Values must be of type <class 'openpyxl.drawing.text.EmbeddedWAVAudioFile'>

tgtFrame
Values must be of type <class 'openpyxl.descriptors.base.String'>

tooltip
Values must be of type <class 'openpyxl.descriptors.base.String'>

class openpyxl.drawing.text.**LineBreak**(*rPr=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

rPr
Values must be of type <class 'openpyxl.drawing.text.CharacterProperties'>

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*)

Bases: `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*)

Bases: `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'>

`r`
Values must be of type <class 'openpyxl.drawing.text.RegularTextRun'>

`tagname` = 'p'

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*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`algn`
Value must be one of {‘dist’, ‘r’, ‘ctr’, ‘l’, ‘just’, ‘thaiDist’, ‘justLow’}

`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 ‘openpyxl.descriptors.base.Integer’>

`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 {'b', 'ctr', 'auto', 'base', '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*)
Bases: *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 0x7fd32e0830f0>

class `openpyxl.drawing.text.`**RegularTextRun**(*rPr=None*, *t=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

namespace = 'http://schemas.openxmlformats.org/drawingml/2006/main'

rPr
Values must be of type <class 'openpyxl.drawing.text.CharacterProperties'>

t
Values must be of type <class 'str'>

tagname = 'r'

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*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`anchor`
Value must be one of {'b', 'ctr', 'just', 'dist', '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 {'clip', 'overflow'}

`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.shapes.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 {'mongolianVert', 'eaVert', 'wordArtVertRtl', 'horz', 'vert270', 'vert', 'wordArtVert'}

vertOverflow
Value must be one of {'clip', 'ellipsis', 'overflow'}

wrap
Value must be one of {'none', 'square'}

class openpyxl.drawing.text.**Spacing**(*spcPct=None*, *spcPts=None*)
Bases: *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*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

algn
Values must be of type <openpyxl.descriptors.base.Set object at 0x7fd32e076e80>

pos
Values must be of type <class 'openpyxl.descriptors.base.Integer'>

class openpyxl.drawing.text.**TabStopList**(*tab=None*)
Bases: *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*)
Bases: *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 'openpyxl.descriptors.base.String'>

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

class openpyxl.drawing.text.**TextNormalAutofit**(*fontScale=None*, *lnSpcReduction=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

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

openpyxl.formatting package

Submodules

openpyxl.formatting.formatting module

class openpyxl.formatting.formatting.**ConditionalFormatting**
Bases: object

Conditional formatting rules.

add(*range_string*, *cfRule*)
Add a rule such as ColorScaleRule, FormulaRule or CellIsRule

The priority will be added automatically.

setDxfStyles(*wb*)

update(*cfRules*)

openpyxl.formatting.formatting.**unpack_rules**(*cfRules*)

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*)
Bases: *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*)

Bases: `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*)

Bases: `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', 'num', 'max', '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*)

Bases: `openpyxl.formatting.rule.RuleType`

`iconSet`
Value must be one of {'3TrafficLights1', '5Quarters', '4RedToBlack', '3Symbols2', '4Arrows', '3Symbols', '3TrafficLights2', '4Rating', '5Rating', '3Arrows', '5Arrows', '3ArrowsGray', '3Flags', '4TrafficLights', '5ArrowsGray', '4ArrowsGray', '3Signs'}

`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*)

Bases: `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 {'notEqual', 'notContains', 'between', 'beginsWith', 'endsWith', 'greaterThan', 'lessThanOrEqual', 'notBetween', 'containsText', 'greaterThanOrEqual', 'lessThan', 'equal'}

`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 {'yesterday', 'last7Days', 'tomorrow', 'nextMonth', 'lastWeek', 'nextWeek', 'lastMonth', 'thisMonth', 'thisWeek', 'today'}

type
Value must be one of {'aboveAverage', 'dataBar', 'top10', 'cellIs', 'expression', 'notContainsBlanks', 'colorScale', 'uniqueValues', 'beginsWith', 'endsWith', 'iconSet', 'containsErrors', 'notContainsErrors', 'containsBlanks', 'containsText', 'timePeriod', 'duplicateValues', 'notContainsText'}

class openpyxl.formatting.rule.**RuleType**
Bases: *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*)
Bases: *openpyxl.descriptors.base.Float*

Expected type depends upon type attribue of parent :-(

openpyxl.packaging package

Stuff related to Office OpenXML packaging: relationships, archive, content types.

Submodules

openpyxl.packaging.manifest module

class openpyxl.packaging.manifest.**FileExtension**(*Extension*, *ContentType*)
Bases: *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=()*)
Bases: *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

extensions

filenames

tagname = **'Types'**

to_tree()
Custom serialisation method to allow setting a default namespace

class openpyxl.packaging.manifest.**Override**(*PartName*, *ContentType*)
Bases: *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.**write_content_types**(*workbook*, *as_template=False*, *exts=None*)

openpyxl.packaging.relationship module

class openpyxl.packaging.relationship.**Relationship**(*type=None*, *target=None*, *targetMode=None*, *id=None*, *Id=None*, *Type=None*, *Target=None*)
Bases: *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'>

tagname = 'Relationship'

class openpyxl.packaging.relationship.**RelationshipList**(*Relationship=()*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

append(*value*)

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.reader package

Submodules

openpyxl.reader.excel module

openpyxl.reader.excel.**load_workbook**(*filename*, *read_only=False*, *keep_vba=False*, *data_only=False*, *guess_types=False*)
Open the given filename and return the workbook

Parameters

- **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)
- **guess_types** (*bool*) – guess cell content type and do not read it from the file
- **data_only** (*bool*) – controls whether cells with formulae have either the formula (default) or the value stored the last time Excel read the sheet

Return type openpyxl.workbook.Workbook

Note: When using lazy load, all worksheets will be openpyxl.worksheet.iter_worksheet.IterableWorksheet and the returned workbook will be read-only.

---

openpyxl.reader.excel.**repair_central_directory**(*zipFile*, *is_file_instance*)
trims trailing data from the central directory code taken from http://stackoverflow.com/a/7457686/570216, courtesy of Uri Cohen

openpyxl.reader.strings module

openpyxl.reader.strings.**read_string_table**(*xml_source*)
Read in all shared strings in the table

openpyxl.reader.style module

openpyxl.reader.workbook module

openpyxl.reader.workbook.**read_content_types**(*archive*)
Read content types.

openpyxl.reader.workbook.**read_rels**(*archive*)
Read relationships for a workbook

openpyxl.reader.workbook.**read_sheets**(*archive*)
Read worksheet titles and ids for a workbook

openpyxl.reader.worksheet module

class openpyxl.reader.worksheet.**WorkSheetParser**(*wb*, *title*, *xml_source*, *shared_strings*)
Bases: object

CELL_TAG = '{http://schemas.openxmlformats.org/spreadsheetml/2006/main}c'

FORMULA_TAG = '{http://schemas.openxmlformats.org/spreadsheetml/2006/main}f'

INLINE_STRING = '{http://schemas.openxmlformats.org/spreadsheetml/2006/main}is'

MERGE_TAG = '{http://schemas.openxmlformats.org/spreadsheetml/2006/main}mergeCell'

VALUE_TAG = '{http://schemas.openxmlformats.org/spreadsheetml/2006/main}v'

parse()

parse_auto_filter(*element*)

parse_cell(*element*)

parse_column_dimensions(*col*)

parse_data_validation(*element*)

parse_extensions(*element*)

parse_header_footer(*element*)

parse_legacy_drawing(*element*)

parse_margins (*element*)

parse_merge (*element*)

parse_page_setup (*element*)

parse_print_options (*element*)

parse_properties (*element*)

parse_row_dimensions (*row*)

parse_sheet_protection (*element*)

parse_sheet_views (*element*)

parse_sort (*element*)

parser_conditional_formatting (*element*)

openpyxl.styles package

class openpyxl.styles.**Style** (*font=Font(color=Color(indexed=Values must be of type <class 'int'>, auto=Values must be of type <class 'bool'>, theme=Values must be of type <class 'int'>)), fill=, border=, alignment=, number_format=None, protection=*)

Bases: *openpyxl.styles.hashable.HashableObject*

Style object containing all formatting details.

alignment
Values must be of type <class 'openpyxl.styles.alignment.Alignment'>

border
Values must be of type <class 'openpyxl.styles.borders.Border'>

copy ()

fill
Values must be of type <class 'openpyxl.styles.fills.Fill'>

font
Values must be of type <class 'openpyxl.styles.fonts.Font'>

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

protection
Values must be of type <class 'openpyxl.styles.protection.Protection'>

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*)

Bases: *openpyxl.styles.hashable.HashableObject*

Alignment options for use in styles.

horizontal
Value must be one of {'center', 'general', 'right', 'centerContinuous', 'left', 'fill', 'justify', 'distributed'}

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'>

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}

vertical
Value must be one of {'justify', 'center', 'top', 'bottom', 'distributed'}

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

openpyxl.styles.borders module

class openpyxl.styles.borders.**Border**(*left=*, *right=*, *top=*, *bottom=*, *diagonal=*, *diagonal_direction=None*, *vertical=None*, *horizontal=None*, *diagonalUp=False*, *diagonalDown=False*, *outline=True*, *start=None*, *end=None*)
Bases: *openpyxl.styles.hashable.HashableObject*

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*)
Bases: *openpyxl.styles.hashable.HashableObject*

Border options for use in styles. Caution: if you do not specify a border_style, other attributes will have no effect !

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

style
Value must be one of {'dashed', 'thick', 'mediumDashed', 'thin', 'slantDashDot', 'mediumDashDotDot', 'dashDotDot', 'dashDot', 'medium', 'mediumDashDot', 'dotted', 'double', 'hair'}

openpyxl.styles.colors module

class openpyxl.styles.colors.**Color**(*rgb='00000000'*, *indexed=None*, *auto=None*, *theme=None*, *tint=0.0*, *index=None*, *type='rgb'*)
Bases: *openpyxl.styles.hashable.HashableObject*

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*)
Bases: *`openpyxl.descriptors.base.Typed`*

`expected_type`
alias of *`Color`*

class `openpyxl.styles.colors.`**`ColorList`**(*indexedColors=None*, *mruColors=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`index`

`indexedColors`
Values must be of type <class 'openpyxl.styles.colors.IndexedColorList'>

`mruColors`
Values must be of type <class 'openpyxl.styles.colors.MRUColorList'>

class `openpyxl.styles.colors.`**`IndexedColorList`**(*rgbColor=()*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

class `openpyxl.styles.colors.`**`MRUColorList`**(*color=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

class `openpyxl.styles.colors.`**`RGB`**(**args*, ***kw*)
Bases: *`openpyxl.descriptors.base.Typed`*

Descriptor for aRGB values If not supplied alpha is 00

`expected_type`
alias of `str`

class `openpyxl.styles.colors.`**`RgbColor`**(*rgb=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`rgb`

openpyxl.styles.differential module

class `openpyxl.styles.differential.`**`DifferentialStyle`**(*font=None*, *numFmt=None*, *fill=None*, *alignment=None*, *border=None*, *protection=None*, *extLst=None*)
Bases: *`openpyxl.styles.hashable.HashableObject`*

`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'

openpyxl.styles.fills module

class `openpyxl.styles.fills.`**Fill**
Bases: *openpyxl.styles.hashable.HashableObject*

Base class

classmethod from_tree (*el*)

tagname = 'fill'

class `openpyxl.styles.fills.`**GradientFill** (*type='linear'*, *degree=0*, *left=0*, *right=0*, *top=0*, *bottom=0*, *stop=()*, *fill_type=None*)
Bases: *openpyxl.styles.fills.Fill*

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

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

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

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

stop
A sequence of primitive types that are stored as a single attribute. "val" is the default attribute

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=Color(indexed=Values must be of type <class 'int'>*, *auto=Values must be of type <class 'bool'>*, *theme=Values must be of type <class 'int'>)*, *bgColor=Color(indexed=Values must be of type <class 'int'>*, *auto=Values must be of type <class 'bool'>*, *theme=Values must be of type <class 'int'>)*, *fill_type=None*, *start_color=None*, *end_color=None*)
Bases: *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'>

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

patternType
Value must be one of {'lightHorizontal', 'solid', 'darkTrellis', 'lightGray', 'darkGrid', 'gray125', 'lightUp', 'darkGray', 'darkVertical', 'lightVertical', 'gray0625', 'lightGrid', 'lightDown', 'lightTrellis', 'darkHorizontal', 'darkUp', 'mediumGray', 'darkDown'}

tagname = 'patternFill'

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

openpyxl.styles.fonts module

class openpyxl.styles.fonts.**Font** (*name='Calibri'*, *sz=11*, *b=False*, *i=False*, *charset=None*, *u=None*, *strike=False*, *color='00000000'*, *scheme=None*, *family=2*, *size=None*, *bold=None*, *italic=None*, *strikethrough=None*, *underline=None*, *vertAlign=None*, *outline=False*, *shadow=False*, *condense=False*, *extend=False*)

Bases: *openpyxl.styles.hashable.HashableObject*

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'>

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'>

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

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

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

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 = 'font'

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

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

openpyxl.styles.hashable module

class openpyxl.styles.hashable.**HashableObject**
Bases: *openpyxl.descriptors.serialisable.Serialisable*

Define how to hash property classes.

copy(***kwargs*)

key
Use a tuple of fields as the basis for a key

openpyxl.styles.named_styles module

class openpyxl.styles.named_styles.**NamedCellStyle**(*name=None*, *xfId=None*, *builtinId=None*, *iLevel=None*, *hidden=None*, *customBuiltin=None*, *extLst=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

Pointer-based representation of named styles in XML xfId refers to the corresponding CellStyleXf

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

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

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

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

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

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

tagname = 'cellStyle'

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

class openpyxl.styles.named_styles.**NamedCellStyleList**(*count=None*, *cellStyle=()*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

count

names
Convert to NamedStyle objects and remove duplicates

tagname = 'cellStyles'

class openpyxl.styles.named_styles.**NamedStyle**(*name='Normal', font=Font(color=Color(indexed=Values must be of type <class 'int'>, auto=Values must be of type <class 'bool'>, theme=Values must be of type <class 'int'>)), fill=, border=, alignment=, number_format=None, protection=, builtinId=0, hidden=False*)
Bases: *openpyxl.styles.hashable.HashableObject*

Named and editable styles

alignment
Values must be of type <class 'openpyxl.styles.alignment.Alignment'>

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'>

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

protection
Values must be of type <class 'openpyxl.styles.protection.Protection'>

openpyxl.styles.numbers module

class openpyxl.styles.numbers.**NumberFormat**(*numFmtId=None, formatCode=None*)
Bases: *openpyxl.styles.hashable.HashableObject*

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

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

class openpyxl.styles.numbers.**NumberFormatDescriptor**(**args, **kw*)
Bases: *openpyxl.descriptors.base.String*

class openpyxl.styles.numbers.**NumberFormatList**(*count=None, numFmt=()*)
Bases: *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.protection module

class `openpyxl.styles.protection.`**`Protection`**(*locked=True*, *hidden=False*)

Bases: *`openpyxl.styles.hashable.HashableObject`*

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*)

Bases: `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

openpyxl.styles.styleable module

class `openpyxl.styles.styleable.`**`NumberFormatDescriptor`**

Bases: `object`

`collection` **= '_number_formats'**

`key` **= 'numFmtId'**

class `openpyxl.styles.styleable.`**`StyleDescriptor`**(*collection*, *key*)

Bases: `object`

class `openpyxl.styles.styleable.`**`StyleableObject`**(*sheet*, *style_array=None*)

Bases: `object`

Base class for styleble objects implementing proxy and lookup functions

`has_style`

`parent`

`pivotButton`

`quotePrefix`

`style`

`style_id`

openpyxl.utils package

openpyxl.utils.**absolute_coordinate**(*coord_string*)
Convert a coordinate to an absolute coordinate string (B12 -> $B$12)

openpyxl.utils.**cols_from_range**(*range_string*)
Get individual addresses for every cell in a range. Yields one row at a time.

openpyxl.utils.**column_index_from_string**(*str_col*)
Convert a column name into a numerical index ('A' -> 1)

openpyxl.utils.**coordinate_from_string**(*coord_string*)
Convert a coordinate string like 'B12' to a tuple ('B', 12)

openpyxl.utils.**coordinate_to_tuple**(*coordinate*)
Convert an Excel style coordinate to (row, colum) tuple

openpyxl.utils.**get_column_interval**(*start*, *end*)

openpyxl.utils.**get_column_letter**(*idx*)
Convert a column index into a column letter (3 -> 'C')

openpyxl.utils.**quote_sheetname**(*sheetname*)

openpyxl.utils.**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.**range_to_tuple**(*range_string*)
Convert a worksheet range to the sheetname and maximum and minimum coordinate indices

openpyxl.utils.**rows_from_range**(*range_string*)
Get individual addresses for every cell in a range. Yields one row at a time.

Submodules

openpyxl.utils.bound_dictionary module

class openpyxl.utils.bound_dictionary.**BoundDictionary**(*reference=None*, **args*, ***kw*)
Bases: 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.datetime module

class openpyxl.utils.datetime.**GMT**
Bases: datetime.tzinfo

dst(*dt*)

tzname(*dt*)

utcoffset(*dt*)

openpyxl.utils.datetime.**W3CDTF_to_datetime**(*formatted_string*)
Convert from a timestamp string to a datetime object.

openpyxl.utils.datetime.**datetime_to_W3CDTF**(*dt*)
Convert from a datetime to a timestamp string.

openpyxl.utils.datetime.**days_to_time**(*value*)

openpyxl.utils.datetime.**from_excel**(*value*, *offset=2415018.5*)

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_excel**(*dt*, *offset=2415018.5*)

openpyxl.utils.exceptions module

exception openpyxl.utils.exceptions.**CellCoordinatesException**
Bases: `Exception`

Error for converting between numeric and A1-style cell references.

exception openpyxl.utils.exceptions.**IllegalCharacterError**
Bases: `Exception`

The data submitted which cannot be used directly in Excel files. It must be removed or escaped.

exception openpyxl.utils.exceptions.**InsufficientCoordinatesException**
Bases: `Exception`

Error for partially specified cell coordinates.

exception openpyxl.utils.exceptions.**InvalidFileException**
Bases: `Exception`

Error for trying to open a non-ooxml file.

exception openpyxl.utils.exceptions.**NamedRangeException**
Bases: `Exception`

Error for badly formatted named ranges.

exception openpyxl.utils.exceptions.**ReadOnlyWorkbookException**
Bases: `Exception`

Error for trying to modify a read-only workbook

exception openpyxl.utils.exceptions.**SheetTitleException**
Bases: `Exception`

Error for bad sheet names.

exception openpyxl.utils.exceptions.**WorkbookAlreadySaved**
Bases: `Exception`

Error when attempting to perform operations on a dump workbook while it has already been dumped once

openpyxl.utils.indexed_list module

class openpyxl.utils.indexed_list.**IndexedList**(*iterable=None*)
Bases: `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*)

index(*value*)

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-open-xml/

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.names package

Submodules

openpyxl.workbook.names.external module

class `openpyxl.workbook.names.external.`**`ExternalBook`**(*Id*, *Target*, *TargetMode=None*, *Type=None*)

Bases: *`openpyxl.descriptors.Strict`*

Map the relationship of one workbook to another

`Id`
Values must be of type <class 'str'>

`Target`
Values must be of type <class 'str'>

`TargetMode` = **'External'**

`Type` = **'http://schemas.openxmlformats.org/officeDocument/2006/relationships/externalLinkPath'**

class `openpyxl.workbook.names.external.`**`ExternalRange`**(*name*, *refersTo=None*, *sheetId=None*)

Bases: *`openpyxl.descriptors.Strict`*

Map external named ranges NB. the specification for these is different to named ranges within a workbook See 18.14.5

`name`
Values must be of type <class 'str'>

`refersTo`
Values must be of type <class 'str'>

`sheetId`
Values must be of type <class 'str'>

`openpyxl.workbook.names.external.`**`detect_external_links`**(*rels*, *archive*)

`openpyxl.workbook.names.external.`**`parse_books`**(*xml*)

`openpyxl.workbook.names.external.`**`parse_ranges`**(*xml*)

`openpyxl.workbook.names.external.`**`write_external_book_rel`**(*book*)
Serialise link to external file

`openpyxl.workbook.names.external.`**`write_external_link`**(*links*)
Serialise links to ranges in a single external worbook

openpyxl.workbook.names.named_range module

class `openpyxl.workbook.names.named_range.`**`NamedRange`**(*name*, *destinations*, *scope=None*)

Bases: *`openpyxl.workbook.names.named_range.NamedValue`*

A named group of cells

Scope is a worksheet object or None for workbook scope names (the default)

`destinations`

name

repr_format = '<%s "%s">'

scope

str_format = '%s!%s'

value

openpyxl.workbook.names.named_range.**NamedRangeContainingValue**
alias of *NamedValue*

class openpyxl.workbook.names.named_range.**NamedValue**(*name*, *value*)
Bases: object

A named value

localSheetId

name

scope

value

openpyxl.workbook.names.named_range.**external_range**(*range_string*)

openpyxl.workbook.names.named_range.**read_named_ranges**(*xml_source*, *workbook*)
Read named ranges, excluding poorly defined ranges.

openpyxl.workbook.names.named_range.**refers_to_range**(*range_string*)

openpyxl.workbook.names.named_range.**split_named_range**(*range_string*)
Separate a named range into its component parts

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

openpyxl.workbook.properties module

class openpyxl.workbook.properties.**CalcProperties**(*calcId=122211*, *calcMode='auto'*, *fullCalcOnLoad=True*, *refMode='A1'*, *iterate=False*, *iterateCount=None*, *iterateDelta=None*, *fullPrecision=None*, *calcCompleted=True*, *calcOnSave=True*, *concurrentCalc=True*, *concurrentManualCount=None*, *forceFullCalc=None*)
Bases: *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 {'auto', 'autoNoTable', 'manual'}

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*)
Bases: *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='userSet'*, *codeName=None*, *hidePivotFieldList=None*, *showPivotChartFilter=None*, *allowRefreshQuery=None*, *publishItems=None*, *checkCompatibility=None*, *autoCompressPictures=None*, *refreshAllConnections=None*, *defaultThemeVersion=None*)

Bases: *`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', 'never', 'always'}

openpyxl.workbook.workbook module

class `openpyxl.workbook.workbook.`**`Workbook`**(*write_only=False*)
Bases: `object`

Workbook is the container for all other parts of the document.

active
Get the currently active sheet

add_named_range(*named_range*)
Add an existing named_range to the list of named_ranges.

chartsheets

create_chartsheet(*title=None*, *index=None*)

create_named_range(*name*, *worksheet*, *range*, *scope=None*)
Create a new named_range on a worksheet

create_sheet(*title=None*, *index=None*)
Create a worksheet (at an optional index).

Parameters

- **title** – optional title of the sheet
- **index** (*int*) – optional position at which the sheet will be inserted

data_only

get_active_sheet()
Returns the current active sheet.

get_index(*worksheet*)
Return the index of the worksheet.

get_named_range(*name*)
Return the range specified by name.

get_named_ranges()
Return all named ranges

get_sheet_by_name(*name*)
Returns a worksheet by its name.

Parameters **name** (*string*) – the name of the worksheet to look for

get_sheet_names()

`read_only`

`remove_named_range`(*named_range*)
Remove a named_range from this workbook.

`remove_sheet`(*worksheet*)
Remove a worksheet from this workbook.

`save`(*filename*)
Save the current workbook under the given *filename*. Use this function instead of using an *ExcelWriter*.

> **Warning:** 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 the workbook.

Names are returned in the worksheets order.

Return type list of strings

`worksheets`

`write_only`

openpyxl.worksheet package

`openpyxl.worksheet.`**`isgenerator`**(*obj*)

Submodules

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=None*, *promptTitle=None*, *errorStyle=None*, *error=None*, *prompt=None*, *errorTitle=None*, *imeMode=None*, *operator=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

`add`(*cell*)
Adds a openpyxl.cell to this validator

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

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

`error`
Values must be of type <class 'str'>

errorStyle
Value must be one of {'warning', 'stop', 'information'}

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

formula1
Values must be of type <class 'str'>

formula2
Values must be of type <class 'str'>

imeMode
Value must be one of {'fullKatakana', 'halfKatakana', 'noControl', 'hiragana', 'off', 'disabled', 'on', 'halfAlpha', 'fullAlpha', 'fullHangul', 'halfHangul'}

operator
Value must be one of {'notEqual', 'between', 'greaterThan', 'lessThanOrEqual', 'notBetween', 'greaterThanOrEqual', 'lessThan', 'equal'}

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

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

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

tagname = 'dataValidation'

type
Value must be one of {'decimal', 'list', 'custom', 'time', 'date', 'textLength', 'whole'}

class openpyxl.worksheet.datavalidation.**DataValidationList**(*disablePrompts=None*, *xWindow=None*, *yWindow=None*, *count=None*, *dataValidation=()*)
Bases: *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'

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). More work to come.

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=None*, *bestFit=False*, *hidden=False*, *outlineLevel=0*, *outline_level=None*, *collapsed=False*, *style=None*, *min=None*, *max=None*, *customWidth=False*, *visible=None*, *auto_size=None*)
Bases: *openpyxl.worksheet.dimensions.Dimension*

Information about the display properties of a column.

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'>

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

class openpyxl.worksheet.dimensions.**Dimension**(*index*, *hidden*, *outlineLevel*, *collapsed*, *worksheet*, *visible=True*, *style=None*)
Bases: *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'>

visible

class openpyxl.worksheet.dimensions.**DimensionHolder**(*worksheet*, *reference='index'*, *default_factory=None*)

Bases: *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 columns together

Parameters

- **start** – first column to be grouped (mandatory)
- **end** – last column to be grouped (optional, default to start)
- **outline_level** – outline level
- **hidden** – should the group be hidden on workbook open or not

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*)

Bases: *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

ht

Values must be of type <class 'float'>

thickBot

Values must be of type <class 'bool'>

thickTop

Values must be of type <class 'bool'>

openpyxl.worksheet.drawing module

class openpyxl.worksheet.drawing.**Drawing**(*id=None*)

Bases: *openpyxl.descriptors.serialisable.Serialisable*

id

Values must be of type <class 'str'>

tagname = 'drawing'

openpyxl.worksheet.filters module

class openpyxl.worksheet.filters.**AutoFilter**(*ref=None*, *filterColumn=()*, *sortState=None*, *extLst=None*)

Bases: *openpyxl.descriptors.serialisable.Serialisable*

add_filter_column (*col_id*, *vals*, *blank=False*)

Add row filter for specified column.

Parameters

- **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.

Parameters

- **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

Values must be of type <class ‘str’>

sortState

Values must be of type <class ‘openpyxl.worksheet.filters.SortState’>

tagname = ‘autoFilter’

class openpyxl.worksheet.filters.**CellRange** (**args*, ***kw*)

Bases: *openpyxl.descriptors.base.Convertible*, *openpyxl.descriptors.base.MatchPattern*

allow_none = True

expected_type

alias of `str`

pattern = ‘\n[$]?(?P<min_col>[A-Z]+)\n[$]?(?P<min_row>\\d+)\n(:[$]?(?P<max_col>[A-Z]+)\n[$]?(?P<max_row>\\d+)

class openpyxl.worksheet.filters.**ColorFilter** (*dxfId=None*, *cellColor=None*)

Bases: *openpyxl.descriptors.serialisable.Serialisable*

cellColor

Values must be of type <class ‘bool’>

dxfId

Values must be of type <class ‘int’>

class openpyxl.worksheet.filters.**CustomFilter** (*operator=None*, *val=None*)

Bases: *openpyxl.descriptors.serialisable.Serialisable*

operator

Value must be one of {‘notEqual’, ‘greaterThan’, ‘lessThanOrEqual’, ‘greaterThanOrEqual’, ‘lessThan’, ‘equal’}

val

Values must be of type <class ‘str’>

class openpyxl.worksheet.filters.**CustomFilters** (*_and=None*, *customFilter=None*)

Bases: *openpyxl.descriptors.serialisable.Serialisable*

customFilter
Values must be of type <class 'openpyxl.worksheet.filters.CustomFilter'>

class `openpyxl.worksheet.filters.`**`DateGroupItem`**(*year=None*, *month=None*, *day=None*, *hour=None*, *minute=None*, *second=None*, *dateTimeGrouping=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

dateTimeGrouping
Value must be one of {'second', 'hour', 'year', 'minute', 'month', 'day'}

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

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

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

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

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

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

class `openpyxl.worksheet.filters.`**`DynamicFilter`**(*type=None*, *val=None*, *valIso=None*, *maxVal=None*, *maxValIso=None*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

maxValIso
Values must be of type <class 'datetime.datetime'>

type
Value must be one of {'M10', 'yesterday', 'tomorrow', 'nextMonth', 'null', 'M12', 'nextWeek', 'nextYear', 'today', 'lastYear', 'lastWeek', 'M3', 'M11', 'lastQuarter', 'lastMonth', 'Q4', 'thisYear', 'Q2', 'M6', 'M2', 'belowAverage', 'M4', 'Q3', 'yearToDate', 'M5', 'M9', 'thisQuarter', 'M1', 'M8', 'aboveAverage', 'Q1', 'M7', 'thisMonth', 'thisWeek', 'nextQuarter'}

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*)
Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

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=()*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

calendarType
Value must be one of {'gregorianMeFrench', 'japan', 'gregorianArabic', 'gregorianXlitEnglish', 'taiwan', 'gregorianUs', 'gregorian', 'korea', 'saka', 'gregorianXlitFrench', 'hijri', 'thai', '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

class openpyxl.worksheet.filters.**IconFilter**(*iconSet=None*, *iconId=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

iconSet
Value must be one of {'3TrafficLights1', '5Quarters', '4RedToBlack', '3Symbols2', '4Arrows', '3Symbols', '3TrafficLights2', '4Rating', '5Rating', '3Arrows', '5Arrows', '3ArrowsGray', '3Flags', '4TrafficLights', '5ArrowsGray', '4ArrowsGray', '3Signs'}

class openpyxl.worksheet.filters.**SortCondition**(*ref=None*, *descending=None*, *sortBy=None*, *customList=None*, *dxfId=None*, *iconSet=None*, *iconId=None*)
Bases: *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 {'3TrafficLights1', '5Quarters', '4RedToBlack', '3Symbols2', '4Arrows', '3Symbols', '3TrafficLights2', '4Rating', '5Rating', '3Arrows', '5Arrows', '3ArrowsGray', '3Flags', '4TrafficLights', '5ArrowsGray', '4ArrowsGray', '3Signs'}

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

sortBy
Value must be one of {'value', 'icon', 'cellColor', 'fontColor'}

tagname = 'sortCondition'

class openpyxl.worksheet.filters.**SortState**(*columnSort=None*, *caseSensitive=None*, *sortMethod=None*, *ref=None*, *sortCondition=()*, *extLst=None*)
Bases: *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
Values must be of type <class 'str'>

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

sortMethod
Value must be one of {'stroke', 'pinYin'}

tagname = 'sortState'

class openpyxl.worksheet.filters.**Top10**(*top=None*, *percent=None*, *val=None*, *filterVal=None*)
Bases: *openpyxl.descriptors.serialisable.Serialisable*

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

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

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**
Bases: object

Information about the header/footer for this sheet.

center_footer

center_header

getFooter()

getHeader()

hasFooter()

hasHeader()

left_footer

left_header

right_footer

right_header

setFooter(*item*)

setHeader(*item*)

class openpyxl.worksheet.header_footer.**HeaderFooterItem**(*type*)

Bases: object

Individual left/center/right header/footer items

Header & Footer ampersand codes:

- &A Inserts the worksheet name
- &B Toggles bold
- &D or &[Date] Inserts the current date
- &E Toggles double-underline
- &F or &[File] Inserts the workbook name
- &I Toggles italic
- &N or &[Pages] Inserts the total page count
- &S Toggles strikethrough
- &T Inserts the current time
- &[Tab] Inserts the worksheet name
- &U Toggles underline
- &X Toggles superscript
- &Y Toggles subscript
- &P or &[Page] Inserts the current page number
- &P+n Inserts the page number incremented by n
- &P-n Inserts the page number decremented by n
- &[Path] Inserts the workbook path
- && Escapes the ampersand character
- &"fontname" Selects the named font
- &nn Selects the specified 2-digit font point size

CENTER = 'C'

LEFT = 'L'

REPLACE_LIST = (('\n', '_x000D_'), ('&[Page]', '&P'), ('&[Pages]', '&N'), ('&[Date]', '&D'), ('&[Time]', '&T'), ('&[Pa

RIGHT = 'R'

font_color

font_name

font_size

get()

has()

set(*text*)
Convert a compound string into attributes # incomplete because formatting commands can be nested

text

type

openpyxl.worksheet.hyperlink module

class openpyxl.worksheet.hyperlink.**Hyperlink**(*ref=None*, *location=None*, *tooltip=None*, *display=None*, *id=None*, *target=None*)
Bases: *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'>

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*)
Bases: *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*)

Bases: *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'>

tag = '{http://schemas.openxmlformats.org/spreadsheetml/2006/main}printOptions'

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*)

Bases: *openpyxl.descriptors.serialisable.Serialisable*

Worksheet print page setup

autoPageBreaks

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

cellComments
Value must be one of {'asDisplayed', 'atEnd'}

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

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

errors
Value must be one of {'displayed', 'blank', '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*)

horizontalCentered()

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

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

options()

orientation
Value must be one of {'default', 'portrait', 'landscape'}

pageOrder
Value must be one of {'downThenOver', 'overThenDown'}

paperHeight

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

paperWidth

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

setup()

sheet_properties
Proxy property

tagname = 'pageSetup'

to_tree()

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

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

verticalCentered()

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*)

Bases: *`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.`**`PageBreak`**(*count=None*, *manualBreakCount=None*, *brk=[]*)

Bases: *`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.properties module

class `openpyxl.worksheet.properties.`**`Outline`**(*applyStyles=None*, *summaryBelow=None*, *summaryRight=None*, *showOutlineSymbols=None*)

Bases: *`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'>

`tag` = **'{http://schemas.openxmlformats.org/spreadsheetml/2006/main}outlinePr'**

`tagname` = **'outlinePr'**

class `openpyxl.worksheet.properties.`**`PageSetupProperties`**(*autoPageBreaks=None*, *fitToPage=None*)

Bases: *`openpyxl.descriptors.serialisable.Serialisable`*

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

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

tag = '{http://schemas.openxmlformats.org/spreadsheetml/2006/main}pageSetUpPr'

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*)

Bases: *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'>

tag = '{http://schemas.openxmlformats.org/spreadsheetml/2006/main}sheetPr'

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*)

Bases: *`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`()

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

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

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*)

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.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.worksheet.read_only module

class openpyxl.worksheet.read_only.**ReadOnlyWorksheet**(*parent_workbook*, *title*, *worksheet_path*, *xml_source*, *shared_strings*)
Bases: *openpyxl.worksheet.worksheet.Worksheet*

calculate_dimension(*force=False*)

columns

get_squared_range(*min_col*, *min_row*, *max_col*, *max_row*)
The source worksheet file may have columns or rows missing. Missing cells will be created.

max_column

max_row

min_column

min_row

rows

xml_source
Parse xml source on demand, default to Excel archive

openpyxl.worksheet.read_only.**read_dimension**(*source*)

openpyxl.worksheet.related module

class `openpyxl.worksheet.related.`**`Related`**(*id=None*)

Bases: `openpyxl.descriptors.serialisable.Serialisable`

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

`to_tree`(*tagname*)

openpyxl.worksheet.views module

class `openpyxl.worksheet.views.`**`Pane`**(*xSplit=None*, *ySplit=None*, *topLeftCell=None*, *activePane='topLeft'*, *state='split'*)

Bases: `openpyxl.descriptors.serialisable.Serialisable`

`activePane`
Value must be one of {'topLeft', 'bottomRight', 'bottomLeft', 'topRight'}

`state`
Value must be one of {'frozenSplit', 'frozen', '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'*)

Bases: `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 {'topLeft', 'bottomRight', 'bottomLeft', 'topRight'}

`sqref`
Values must be of type <class 'str'>

class `openpyxl.worksheet.views.`**`SheetView`**(*windowProtection=None*, *showFormulas=None*, *showGridLines=True*, *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*, *workbookViewId=0*, *selection=None*, *pane=None*)

Bases: `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 {'normal', 'pageBreakPreview', 'pageLayout'}

`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'>

openpyxl.worksheet.worksheet module

class `openpyxl.worksheet.worksheet.`**`Worksheet`**(*parent*, *title=None*)

Bases: `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_print_title`(*n*, *rows_or_cols='rows'*)

Print Titles are rows or columns that are repeated on each printed sheet. This adds n rows or columns at the top or left of the sheet

`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)

Parameters **`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.

`cell`(*coordinate=None*, *row=None*, *column=None*, *value=None*)
Returns a cell object based on the given coordinates.

Usage: cell(coodinate='A15') **or** cell(row=15, column=1)

If *coordinates* are not given, then row *and* column must be given.

Cells are kept in a dictionary which is empty at the worksheet creation. Calling *cell* creates the cell in memory when they are first accessed, to reduce memory usage.

Parameters

- **`coordinate`** (*string*) – coordinates of the cell (e.g. 'B12')
- **`row`** (*int*) – row index of the cell (e.g. 4)
- **`column`** (*int*) – column index of the cell (e.g. 3)

Raise InsufficientCoordinatesException when coordinate or (row and column) are not given

Return type :class:openpyxl.cell.Cell

`columns`
Iterate over all columns in the worksheet

`dimensions`

`freeze_panes`

`get_cell_collection`()
Return an unordered list of the cells in this worksheet.

`get_named_range`(*range_string*)
Returns a 2D array of cells, with optional row and column offsets.

Parameters **`range_string`** (*string*) – *named range* name

Return type tuples of tuples of `openpyxl.cell.Cell`

`get_squared_range`(*min_col*, *min_row*, *max_col*, *max_row*)
Returns a 2D array of cells

Parameters

- **`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*) – smallest row index (1-based index)

Return type generator

`iter_rows`(*range_string=None*, *row_offset=0*, *column_offset=0*)
Returns a squared range based on the *range_string* parameter, using generators. If no range is passed, will iterate over all cells in the worksheet

Parameters

- **`range_string`** (*string*) – range of cells (e.g. 'A1:C4')
- **`row_offset`** – additional rows (e.g. 4)
- **`column_offset`** – additonal columns (e.g. 3)

Return type generator

`max_column`
Get the largest value for column currently stored.

Return type int

`max_row`
Returns the maximum row index containing data

Return 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`
Public attribute for which cells have been merged

`merged_cells`
Utility for checking whether a cell has been merged or not

`min_column`

`min_row`

`point_pos`(*left=0*, *top=0*)
tells which cell is under the given coordinates (in pixels) counting from the top-left corner of the sheet. Can be used to locate images and charts on the worksheet

`rows`
Iterate over all rows in the worksheet

`selected_cell`

`set_printer_settings`(*paper_size*, *orientation*)
Set printer settings

`show_gridlines`

`show_summary_below`

`show_summary_right`

`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)

`vba_code`

`openpyxl.worksheet.worksheet.`**`flatten`**(*results*)
Return cell values row-by-row

`openpyxl.worksheet.worksheet.`**`isgenerator`**(*obj*)

openpyxl.writer package

Submodules

openpyxl.writer.etree_worksheet module

`openpyxl.writer.etree_worksheet.`**`get_rows_to_write`**(*worksheet*)
Return all rows, and any cells that they contain

`openpyxl.writer.etree_worksheet.`**`write_cell`**(*worksheet*, *cell*, *styled=None*)

`openpyxl.writer.etree_worksheet.`**`write_rows`**(*xf*, *worksheet*)
Write worksheet data to xml.

openpyxl.writer.excel module

class `openpyxl.writer.excel.`**`ExcelWriter`**(*workbook*)
Bases: `object`

Write a workbook object to an Excel file.

`comment_writer`
alias of `CommentWriter`

`save`(*filename*, *as_template=False*)
Write data into the archive.

`write_data`(*archive*, *as_template=False*)
Write the various xml files into the zip archive.

`openpyxl.writer.excel.`**`save_virtual_workbook`**(*workbook*, *as_template=False*)
Return an in-memory workbook, suitable for a Django response.

`openpyxl.writer.excel.`**`save_workbook`**(*workbook*, *filename*, *as_template=False*)
Save the given workbook on the filesystem under the name filename.

Parameters

- **`workbook`** (`openpyxl.workbook.Workbook`) – the workbook to save
- **`filename`** (*string*) – the path to which save the workbook

Return type bool

openpyxl.writer.lxml_worksheet module

`openpyxl.writer.lxml_worksheet.`**`write_cell`**(*xf*, *worksheet*, *cell*, *styled=False*)

`openpyxl.writer.lxml_worksheet.`**`write_rows`**(*xf*, *worksheet*)
Write worksheet data to xml.

openpyxl.writer.relations module

`openpyxl.writer.relations.`**`write_rels`**(*worksheet*, *comments_id=None*, *vba_controls_id=None*)
Write relationships for the worksheet to xml.

openpyxl.writer.strings module

`openpyxl.writer.strings.`**`write_string_table`**(*string_table*)
Write the string table xml.

openpyxl.writer.styles module

openpyxl.writer.theme module

`openpyxl.writer.theme.`**`write_theme`**`()`
: Write the theme xml.

openpyxl.writer.workbook module

`openpyxl.writer.workbook.`**`write_properties_app`**`(`*workbook*`)`
: Write the properties xml.

`openpyxl.writer.workbook.`**`write_root_rels`**`(`*workbook*`)`
: Write the relationships xml.

`openpyxl.writer.workbook.`**`write_workbook`**`(`*workbook*`)`
: Write the core workbook xml.

`openpyxl.writer.workbook.`**`write_workbook_rels`**`(`*workbook*`)`
: Write the workbook relationships xml.

openpyxl.writer.worksheet module

`openpyxl.writer.worksheet.`**`write_cols`**`(`*worksheet*`)`
: Write worksheet columns to xml.

  <cols> may never be empty - spec says must contain at least one child

`openpyxl.writer.worksheet.`**`write_conditional_formatting`**`(`*worksheet*`)`
: Write conditional formatting to xml.

`openpyxl.writer.worksheet.`**`write_drawing`**`(`*worksheet*`)`
: Add link to drawing if required

`openpyxl.writer.worksheet.`**`write_format`**`(`*worksheet*`)`

`openpyxl.writer.worksheet.`**`write_header_footer`**`(`*worksheet*`)`

`openpyxl.writer.worksheet.`**`write_hyperlinks`**`(`*worksheet*`)`
: Write worksheet hyperlinks to xml.

`openpyxl.writer.worksheet.`**`write_mergecells`**`(`*worksheet*`)`
: Write merged cells to xml.

`openpyxl.writer.worksheet.`**`write_worksheet`**`(`*worksheet*, *shared_strings*`)`
: Write a worksheet to an xml file.

openpyxl.writer.write_only module

`openpyxl.writer.write_only.`**`WriteOnlyCell`**`(`*ws=None*, *value=None*`)`

class `openpyxl.writer.write_only.`**`WriteOnlyWorksheet`**`(`*parent_workbook*, *title*`)`
: Bases: *`openpyxl.worksheet.worksheet.Worksheet`*

  Streaming worksheet using lxml Optimised to reduce memory by writing rows just in time Cells can be styled and have comments Styles for rows and columns must be applied before writing cells

  `append``(`*row*`)`

  Parameters **`row`** (*iterable*) – iterable containing values to append

  `cell``(`**args*, ***kw*`)`

  `close``()`

  `filename`

  `merge_cells``(`**args*, ***kw*`)`

range(**args*, ***kw*)

writer = None

openpyxl.writer.write_only.**create_temporary_file**(*suffix=''*)

openpyxl.writer.write_only.**isgenerator**(*obj*)

openpyxl.writer.write_only.**removed_method**(**args*, ***kw*)

openpyxl.writer.write_only.**save_dump**(*workbook*, *filename*)

openpyxl.xml package

openpyxl.xml.**lxml_available**()

openpyxl.xml.**lxml_env_set**()

Submodules

openpyxl.xml.constants module

openpyxl.xml.functions module

openpyxl.xml.functions.**ConditionalElement**(*node*, *tag*, *condition*, *attr=None*)
Utility function for adding nodes if certain criteria are fulfilled An optional attribute can be passed in which will always be serialised as '1'

openpyxl.xml.functions.**iterparse**(*source*, **args*, ***kw*)

openpyxl.xml.functions.**localname**(*node*)

openpyxl.xml.functions.**safe_iterator**(*node*, *tag=None*)
Return an iterator that is compatible with Python 2.6

openpyxl.xml.functions.**safe_iterparse**(*source*, **args*, ***kw*)

openpyxl.xml.namespace module

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

CHAPTER 11

Release Notes

11.1 2.4.0 (unreleased)

11.1.1 Minor changes

- Remove deprecated methods from DataValidation
- Convert AutoFilter to Serialisable and extend support for filters
- Add support for SortState
- Removed *use_iterators* keyword when loading workbooks. Use *read_only* instead.

11.1.2 Deprecations

Cell anchor method Worksheet point_pos method Comment text attribute

11.2 2.3.2 (unreleased)

11.3 2.3.1 (2015-11-20)

11.3.1 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

11.3.2 Minor changes

- PR 88 Fix page margins in parser.

11.4 2.3.0 (2015-10-20)

11.4.1 Major changes

- Support the creation of chartsheets

11.4.2 Bug fixes

- #532 Problems when cells have no style in read-only mode.

11.4.3 Minor changes

- PR 79 Make PlotArea editable in charts
- Use graphicalProperties as the alias for spPr

11.5 2.3.0-b2 (2015-09-04)

11.5.1 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

11.5.2 Minor changes

- Worksheets are now iterable by row.
- Assign individual cell styles only if they are explicitly set.

11.6 2.3.0-b1 (2015-06-29)

11.6.1 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

11.6.2 Minor changes

- Read-only and write-only worksheets renamed.
- Write-only workbooks support charts and images.
- PR76 Prevent comment images from conflicting with VBA

11.6.3 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

11.7 2.2.6 (unreleased)

11.7.1 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)

11.8 2.2.5 (2015-06-29)

11.8.1 Bug fixes

- #463 Unexpected keyword "mergeCell"
- #484 Unusual dimensions breaks read-only mode
- #485 Move return out of loop

11.9 2.2.4 (2015-06-17)

11.9.1 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

11.10 2.2.3 (2015-05-26)

11.10.1 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

11.11 2.2.2 (2015-04-28)

11.11.1 Bug fixes

- #447 Uppercase datetime number formats not recognised.
- #453 Borders broken in shared_styles.

11.12 2.2.1 (2015-03-31)

11.12.1 Minor changes

- PR54 Improved precision on times near midnight.
- PR55 Preserve macro buttons

11.12.2 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

11.13 2.2.0 (2015-03-11)

11.13.1 Bug fixes

- #415 Improved exception when passing in invalid in memory files.

11.14 2.2.0-b1 (2015-02-18)

11.14.1 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.)

11.14.2 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

11.14.3 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

11.15 2.1.5 (2015-02-18)

11.15.1 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

11.15.2 Minor changes

- Allow cells to be appended to standard worksheets for code compatibility with write-only mode.

11.16 2.1.4 (2014-12-16)

11.16.1 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

11.16.2 Minor changes

- Add relation namespace to root element for compatibility with iWork
- Serialize comments relation in LXML-backend

11.17 2.1.3 (2014-12-09)

11.17.1 Minor changes

- PR 31 Correct tutorial
- PR 32 See #380
- PR 37 Bind worksheet to ColumnDimension objects

11.17.2 Bug fixes

- #379 ws.append() doesn't set RowDimension Correctly
- #380 empty cells formatted as datetimes raise exceptions

11.18 2.1.2 (2014-10-23)

11.18.1 Minor changes

- PR 30 Fix regex for positive exponentials
- PR 28 Fix for #328

11.18.2 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

11.19 2.1.1 (2014-10-08)

11.19.1 Minor changes

- PR 20 Support different workbook code names
- Allow auto_axis keyword for ScatterCharts

11.19.2 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

11.20 2.1.0 (2014-09-21)

11.20.1 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

11.20.2 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

11.20.3 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

11.21 2.0.5 (2014-08-08)

11.21.1 Bug fixes

- #348 incorrect casting of boolean strings
- #349 roundtripping cells with formulae

11.22 2.0.4 (2014-06-25)

11.22.1 Minor changes

- Add a sample file illustrating colours

11.22.2 Bug fixes

- #331 DARKYELLOW was incorrect
- Correctly handle extend attribute for fonts

11.23 2.0.3 (2014-05-22)

11.23.1 Minor changes

- Updated docs

11.23.2 Bug fixes

- #319 Cannot load Workbooks with vertAlign styling for fonts

11.24 2.0.2 (2014-05-13)

11.25 2.0.1 (2014-05-13) brown bag

11.26 2.0.0 (2014-05-13) brown bag

11.26.1 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

11.26.2 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

11.26.3 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

11.26.4 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)

11.27 1.8.6 (2014-05-05)

11.27.1 Minor changes

Fixed typo for import Elementtree

11.27.2 Bugfixes

- #279 Incorrect path for comments files on Windows

11.28 1.8.5 (2014-03-25)

11.28.1 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

11.29 1.8.4 (2014-02-25)

11.29.1 Bugfixes

- #260 better handling of undimensioned worksheets
- #268 non-ascii in formualae
- #282 correct implementation of register_namepsace for Python 2.6

11.30 1.8.3 (2014-02-09)

11.30.1 Major changes

Always parse using cElementTree

11.30.2 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

11.31 1.8.2 (2014-01-17)

- #247 - iterable worksheets open too many files
- #252 - improved handling of lxml
- #253 - better handling of unique sheetnames

11.32 1.8.1 (2014-01-14)

- #246

11.33 1.8.0 (2014-01-08)

11.33.1 Compatibility

Support for Python 2.5 dropped.

11.33.2 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

11.33.3 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

11.34 1.7.0 (2013-10-31)

11.34.1 Major changes

Drops support for Python < 2.5 and last version to support Python 2.5

11.34.2 Compatibility

Tests run on Python 2.5, 2.6, 2.7, 3.2, 3.3

11.34.3 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

11.34.4 Known bugfixes

- #109
- #165
- #179
- #209
- #112
- #166
- #109
- #223
- #124
- #157

11.34.5 Miscellaneous

Performance improvements in optimised writer

Docs updated

O

openpyxl, 3
openpyxl.cell, 77
openpyxl.cell.cell, 77
openpyxl.cell.interface, 78
openpyxl.cell.read_only, 79
openpyxl.cell.text, 79
openpyxl.chart, 81
openpyxl.chart.area_chart, 81
openpyxl.chart.axis, 82
openpyxl.chart.bar_chart, 88
openpyxl.chart.bubble_chart, 89
openpyxl.chart.chartspace, 90
openpyxl.chart.data_source, 95
openpyxl.chart.descriptors, 96
openpyxl.chart.error_bar, 97
openpyxl.chart.label, 97
openpyxl.chart.layout, 99
openpyxl.chart.legend, 100
openpyxl.chart.line_chart, 100
openpyxl.chart.marker, 102
openpyxl.chart.picture, 103
openpyxl.chart.pie_chart, 103
openpyxl.chart.radar_chart, 105
openpyxl.chart.reference, 105
openpyxl.chart.scatter_chart, 106
openpyxl.chart.series, 106
openpyxl.chart.series_factory, 109
openpyxl.chart.shapes, 109
openpyxl.chart.stock_chart, 110
openpyxl.chart.surface_chart, 110
openpyxl.chart.text, 111
openpyxl.chart.title, 112
openpyxl.chart.trendline, 112
openpyxl.chart.updown_bars, 113
openpyxl.chartsheet, 114
openpyxl.chartsheet.chartsheet, 116
openpyxl.chartsheet.custom, 116
openpyxl.chartsheet.properties, 117
openpyxl.chartsheet.protection, 117
openpyxl.chartsheet.publish, 118
openpyxl.chartsheet.relation, 119
openpyxl.chartsheet.tests, 114
openpyxl.chartsheet.tests.test_chartsheet, 114
openpyxl.chartsheet.tests.test_custom, 114
openpyxl.chartsheet.tests.test_properties, 114
openpyxl.chartsheet.tests.test_protection, 115
openpyxl.chartsheet.tests.test_publish, 115
openpyxl.chartsheet.tests.test_relation, 115
openpyxl.chartsheet.tests.test_views, 115
openpyxl.chartsheet.views, 120
openpyxl.comments, 120
openpyxl.comments.author, 120
openpyxl.comments.comments, 121
openpyxl.comments.properties, 121
openpyxl.comments.reader, 122
openpyxl.comments.writer, 123
openpyxl.descriptors, 123
openpyxl.descriptors.base, 123
openpyxl.descriptors.excel, 125
openpyxl.descriptors.namespace, 126
openpyxl.descriptors.nested, 126
openpyxl.descriptors.sequence, 127
openpyxl.descriptors.serialisable, 127
openpyxl.drawing, 128
openpyxl.drawing.colors, 128
openpyxl.drawing.drawing, 132
openpyxl.drawing.effect, 132
openpyxl.drawing.fill, 139
openpyxl.drawing.graphic, 144
openpyxl.drawing.image, 150
openpyxl.drawing.line, 150
openpyxl.drawing.shape, 152
openpyxl.drawing.shapes, 153

openpyxl.drawing.spreadsheet_drawing, 158
openpyxl.drawing.text, 161
openpyxl.formatting, 170
openpyxl.formatting.formatting, 170
openpyxl.formatting.rule, 170
openpyxl.packaging, 173
openpyxl.packaging.manifest, 173
openpyxl.packaging.relationship, 174
openpyxl.reader, 174
openpyxl.reader.excel, 174
openpyxl.reader.strings, 175
openpyxl.reader.workbook, 175
openpyxl.reader.worksheet, 175
openpyxl.styles, 176
openpyxl.styles.alignment, 176
openpyxl.styles.borders, 177
openpyxl.styles.colors, 178
openpyxl.styles.differential, 179
openpyxl.styles.fills, 180
openpyxl.styles.fonts, 181
openpyxl.styles.hashable, 182
openpyxl.styles.named_styles, 182
openpyxl.styles.numbers, 183
openpyxl.styles.protection, 184
openpyxl.styles.proxy, 184
openpyxl.styles.styleable, 184
openpyxl.utils, 185
openpyxl.utils.bound_dictionary, 185
openpyxl.utils.datetime, 185
openpyxl.utils.exceptions, 186
openpyxl.utils.indexed_list, 186
openpyxl.utils.units, 187
openpyxl.workbook, 188
openpyxl.workbook.child, 189
openpyxl.workbook.names, 188
openpyxl.workbook.names.external, 188
openpyxl.workbook.names.named_range, 188
openpyxl.workbook.properties, 189
openpyxl.workbook.workbook, 192
openpyxl.worksheet, 193
openpyxl.worksheet.datavalidation, 193
openpyxl.worksheet.dimensions, 195
openpyxl.worksheet.drawing, 196
openpyxl.worksheet.filters, 196
openpyxl.worksheet.header_footer, 200
openpyxl.worksheet.hyperlink, 202
openpyxl.worksheet.page, 202
openpyxl.worksheet.pagebreak, 205
openpyxl.worksheet.properties, 205
openpyxl.worksheet.protection, 206
openpyxl.worksheet.read_only, 208
openpyxl.worksheet.related, 209
openpyxl.worksheet.views, 209
openpyxl.worksheet.worksheet, 211
openpyxl.writer, 214
openpyxl.writer.etree_worksheet, 214
openpyxl.writer.excel, 214
openpyxl.writer.lxml_worksheet, 214
openpyxl.writer.relations, 214
openpyxl.writer.strings, 214
openpyxl.writer.theme, 215
openpyxl.writer.workbook, 215
openpyxl.writer.worksheet, 215
openpyxl.writer.write_only, 215
openpyxl.xml, 216
openpyxl.xml.constants, 216
openpyxl.xml.functions, 216
openpyxl.xml.namespace, 216

A

a (openpyxl.drawing.effect.AlphaReplaceEffect attribute), 133
aboveAverage (openpyxl.formatting.rule.Rule attribute), 172
absolute_coordinate() (in module openpyxl.utils), 185
AbsoluteAnchor (class in openpyxl.drawing.spreadsheet_drawing), 158
absoluteAnchor (openpyxl.drawing.spreadsheet_drawing.SpreadsheetDrawing attribute), 160
AbstractCell (class in openpyxl.cell.interface), 78
accent1 (openpyxl.drawing.colors.ColorMapping attribute), 129
accent2 (openpyxl.drawing.colors.ColorMapping attribute), 129
accent3 (openpyxl.drawing.colors.ColorMapping attribute), 129
accent4 (openpyxl.drawing.colors.ColorMapping attribute), 129
accent5 (openpyxl.drawing.colors.ColorMapping attribute), 129
accent6 (openpyxl.drawing.colors.ColorMapping attribute), 129
action (openpyxl.drawing.text.Hyperlink attribute), 164
active (openpyxl.workbook.workbook.Workbook attribute), 192
active_cell (openpyxl.worksheet.worksheet.Worksheet attribute), 211
activeCell (openpyxl.worksheet.views.Selection attribute), 209
activeCellId (openpyxl.worksheet.views.Selection attribute), 209
activePane (openpyxl.worksheet.views.Pane attribute), 209
add() (openpyxl.formatting.formatting.ConditionalFormatting method), 170
add() (openpyxl.utils.indexed_list.IndexedList method), 186
add() (openpyxl.worksheet.datavalidation.DataValidation method), 193
add_chart() (openpyxl.chartsheet.chartsheet.Chartsheet method), 116
add_chart() (openpyxl.worksheet.worksheet.Worksheet method), 211
add_data_validation() (openpyxl.worksheet.worksheet.Worksheet method), 211
add_filter_column() (openpyxl.worksheet.filters.AutoFilter method), 196
add_image() (openpyxl.worksheet.worksheet.Worksheet method), 211
add_named_range() (openpyxl.workbook.workbook.Workbook method), 192
add_print_title() (openpyxl.worksheet.worksheet.Worksheet method), 211
add_sort_condition() (openpyxl.worksheet.filters.AutoFilter method), 197
AdjPoint2D (class in openpyxl.drawing.shapes), 153
AdjustHandleList (class in openpyxl.drawing.shapes), 153
ahLst (openpyxl.drawing.shapes.CustomGeometry2D attribute), 154
algn (openpyxl.drawing.effect.OuterShadowEffect attribute), 136
algn (openpyxl.drawing.effect.ReflectionEffect attribute), 138
algn (openpyxl.drawing.fill.TileInfoProperties attribute), 143
algn (openpyxl.drawing.line.LineProperties attribute), 151
algn (openpyxl.drawing.text.ParagraphProperties attribute), 166
algn (openpyxl.drawing.text.TabStop attribute), 169
algorithmName (openpyxl.chartsheet.protection.ChartsheetProtection attribute), 118
algorithmName (openpyxl.worksheet.protection.SheetProtection attribute), 207

Alias (class in openpyxl.descriptors.base), 123
Alignment (class in openpyxl.styles.alignment), 176
alignment (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
alignment (openpyxl.cell.text.PhoneticProperties attribute), 80
alignment (openpyxl.styles.differential.DifferentialStyle attribute), 179
alignment (openpyxl.styles.named_styles.NamedStyle attribute), 183
alignment (openpyxl.styles.Style attribute), 176
allow_blank (openpyxl.worksheet.datavalidation.DataValidation attribute), 193
allow_none (openpyxl.chart.descriptors.NestedGapAmount attribute), 96
allow_none (openpyxl.chart.descriptors.NestedOverlap attribute), 97
allow_none (openpyxl.chart.descriptors.NumberFormatDescriptor attribute), 97
allow_none (openpyxl.chart.title.TitleDescriptor attribute), 112
allow_none (openpyxl.descriptors.base.MatchPattern attribute), 124
allow_none (openpyxl.descriptors.base.Max attribute), 124
allow_none (openpyxl.descriptors.base.Min attribute), 124
allow_none (openpyxl.descriptors.base.Typed attribute), 125
allow_none (openpyxl.descriptors.excel.Relation attribute), 125
allow_none (openpyxl.drawing.colors.ColorChoiceDescriptor attribute), 129
allow_none (openpyxl.worksheet.filters.CellRange attribute), 197
allowBlank (openpyxl.worksheet.datavalidation.DataValidation attribute), 193
allowRefreshQuery (openpyxl.workbook.properties.WorkbookProperties attribute), 191
alpha (openpyxl.drawing.colors.SystemColor attribute), 130
alphaBiLevel (openpyxl.drawing.fill.Blip attribute), 140
AlphaBiLevelEffect (class in openpyxl.drawing.effect), 132
alphaCeiling (openpyxl.drawing.fill.Blip attribute), 140
AlphaCeilingEffect (class in openpyxl.drawing.effect), 132
alphaFloor (openpyxl.drawing.fill.Blip attribute), 140
AlphaFloorEffect (class in openpyxl.drawing.effect), 132
alphaInv (openpyxl.drawing.fill.Blip attribute), 140
AlphaInverseEffect (class in openpyxl.drawing.effect), 132
alphaMod (openpyxl.drawing.colors.SystemColor attribute), 130
alphaMod (openpyxl.drawing.fill.Blip attribute), 140
alphaModFix (openpyxl.drawing.fill.Blip attribute), 140
AlphaModulateEffect (class in openpyxl.drawing.effect), 132
AlphaModulateFixedEffect (class in openpyxl.drawing.effect), 132
alphaOff (openpyxl.drawing.colors.SystemColor attribute), 130
alphaRepl (openpyxl.drawing.fill.Blip attribute), 140
AlphaReplaceEffect (class in openpyxl.drawing.effect), 132
altLang (openpyxl.drawing.text.CharacterProperties attribute), 161
altText (openpyxl.comments.properties.Properties attribute), 122
amt (openpyxl.drawing.effect.AlphaModulateFixedEffect attribute), 132
amt (openpyxl.drawing.effect.TintEffect attribute), 139
anchor (openpyxl.cell.cell.Cell attribute), 77
anchor (openpyxl.comments.properties.Properties attribute), 122
anchor (openpyxl.drawing.drawing.Drawing attribute), 132
anchor (openpyxl.drawing.shapes.Backdrop attribute), 153
anchor (openpyxl.drawing.text.RichTextProperties attribute), 168
anchor() (openpyxl.drawing.image.Image method), 150
AnchorClientData (class in openpyxl.drawing.spreadsheet_drawing), 159
anchorCtr (openpyxl.drawing.text.RichTextProperties attribute), 168
AnchorMarker (class in openpyxl.drawing.spreadsheet_drawing), 159
ang (openpyxl.drawing.fill.LinearShadeProperties attribute), 142
ang (openpyxl.drawing.shapes.ConnectionSite attribute), 154
angle_to_degrees() (in module openpyxl.utils.units), 187
append() (openpyxl.packaging.relationship.RelationshipList method), 174
append() (openpyxl.utils.indexed_list.IndexedList method), 187
append() (openpyxl.worksheet.datavalidation.DataValidationList method), 194
append() (openpyxl.worksheet.pagebreak.PageBreak method), 205
append() (openpyxl.worksheet.worksheet.Worksheet method), 211
append() (openpyxl.writer.write_only.WriteOnlyWorksheet method), 215
applyStyles (openpyxl.worksheet.properties.Outline attribute), 205

applyToEnd (openpyxl.chart.picture.PictureOptions attribute), 103
applyToFront (openpyxl.chart.picture.PictureOptions attribute), 103
applyToSides (openpyxl.chart.picture.PictureOptions attribute), 103
appName (openpyxl.workbook.properties.FileVersion attribute), 190
area3DChart (openpyxl.chart.chartspace.PlotArea attribute), 93
AreaChart (class in openpyxl.chart.area_chart), 81
areaChart (openpyxl.chart.chartspace.PlotArea attribute), 93
AreaChart3D (class in openpyxl.chart.area_chart), 82
ASCII (class in openpyxl.descriptors.base), 123
attribute (openpyxl.descriptors.nested.Nested attribute), 126
attribute (openpyxl.descriptors.sequence.ValueSequence attribute), 127
author (openpyxl.comments.author.AuthorList attribute), 120
author (openpyxl.comments.properties.Comment attribute), 121
authorId (openpyxl.comments.properties.Comment attribute), 121
AuthorList (class in openpyxl.comments.author), 120
authors (openpyxl.comments.properties.CommentSheet attribute), 121
auto (openpyxl.chart.axis.DateAxis attribute), 82
auto (openpyxl.chart.axis.TextAxis attribute), 86
auto (openpyxl.styles.colors.Color attribute), 178
autoCompressPictures (openpyxl.workbook.properties.WorkbookProperties attribute), 191
autoFill (openpyxl.comments.properties.Properties attribute), 122
AutoFilter (class in openpyxl.worksheet.filters), 196
autoFilter (openpyxl.worksheet.protection.SheetProtection attribute), 207
autoLine (openpyxl.comments.properties.Properties attribute), 122
AutonumberBullet (class in openpyxl.drawing.text), 161
autoPageBreaks (openpyxl.worksheet.page.PrintPageSetup attribute), 203
autoPageBreaks (openpyxl.worksheet.properties.PageSetupProperties attribute), 205
autoRepublish (openpyxl.chartsheet.publish.WebPublishItem attribute), 118
autoScale (openpyxl.comments.properties.Properties attribute), 122
autoTitleDeleted (openpyxl.chart.chartspace.ChartContainer attribute), 90
autoUpdate (openpyxl.chart.chartspace.ExternalData attribute), 92
avLst (openpyxl.drawing.shapes.CustomGeometry2D attribute), 154
avLst (openpyxl.drawing.shapes.PresetGeometry2D attribute), 156
avLst (openpyxl.drawing.text.PresetTextShape attribute), 167
avoid_duplicate_name() (in module openpyxl.workbook.child), 189
AxDataSource (class in openpyxl.chart.data_source), 95
axId (openpyxl.chart.axis.DateAxis attribute), 82
axId (openpyxl.chart.axis.NumericAxis attribute), 84
axId (openpyxl.chart.axis.SeriesAxis attribute), 85
axId (openpyxl.chart.axis.TextAxis attribute), 87
axPos (openpyxl.chart.axis.DateAxis attribute), 82
axPos (openpyxl.chart.axis.NumericAxis attribute), 84
axPos (openpyxl.chart.axis.SeriesAxis attribute), 86
axPos (openpyxl.chart.axis.TextAxis attribute), 87

B

b (openpyxl.cell.text.InlineFont attribute), 79
b (openpyxl.drawing.colors.RGBPercent attribute), 130
b (openpyxl.drawing.fill.RelativeRect attribute), 143
b (openpyxl.drawing.shapes.GeomRect attribute), 155
b (openpyxl.drawing.text.CharacterProperties attribute), 161
b (openpyxl.styles.fonts.Font attribute), 181
Backdrop (class in openpyxl.drawing.shapes), 153
backdrop (openpyxl.drawing.shapes.Scene3D attribute), 157
backupFile (openpyxl.workbook.properties.WorkbookProperties attribute), 191
backWall (openpyxl.chart.bar_chart.BarChart3D attribute), 88
backWall (openpyxl.chart.chartspace.ChartContainer attribute), 90
backward (openpyxl.chart.trendline.Trendline attribute), 112
bandFmt (openpyxl.chart.surface_chart.BandFormatList attribute), 110
bandFmts (openpyxl.chart.surface_chart.SurfaceChart attribute), 111
bandFmts (openpyxl.chart.surface_chart.SurfaceChart3D attribute), 111
BandFormat (class in openpyxl.chart.surface_chart), 110
BandFormatList (class in openpyxl.chart.surface_chart), 110
bar3DChart (openpyxl.chart.chartspace.PlotArea attribute), 93
BarChart (class in openpyxl.chart.bar_chart), 88
barChart (openpyxl.chart.chartspace.PlotArea attribute), 93
BarChart3D (class in openpyxl.chart.bar_chart), 88
barDir (openpyxl.chart.bar_chart.BarChart attribute), 88

barDir (openpyxl.chart.bar_chart.BarChart3D attribute), 88
Base64Binary (class in openpyxl.descriptors.excel), 125
base_date (openpyxl.cell.cell.Cell attribute), 77
base_date (openpyxl.cell.interface.AbstractCell attribute), 78
base_date (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
baseline (openpyxl.drawing.text.CharacterProperties attribute), 161
baseTimeUnit (openpyxl.chart.axis.DateAxis attribute), 82
bestFit (openpyxl.worksheet.dimensions.ColumnDimension attribute), 195
Bevel (class in openpyxl.drawing.shapes), 153
bevel (openpyxl.drawing.line.LineProperties attribute), 151
bevelB (openpyxl.drawing.shapes.Shape3D attribute), 157
bevelT (openpyxl.drawing.shapes.Shape3D attribute), 157
bg1 (openpyxl.drawing.colors.ColorMapping attribute), 129
bg2 (openpyxl.drawing.colors.ColorMapping attribute), 129
bgClr (openpyxl.drawing.fill.PatternFillProperties attribute), 142
bgColor (openpyxl.styles.fills.PatternFill attribute), 180
biLevel (openpyxl.drawing.fill.Blip attribute), 140
BiLevelEffect (class in openpyxl.drawing.effect), 133
bIns (openpyxl.drawing.text.RichTextProperties attribute), 168
blackAndWhite (openpyxl.worksheet.page.PrintPageSetup attribute), 203
blank (openpyxl.worksheet.filters.Filters attribute), 199
blend (openpyxl.drawing.effect.FillOverlayEffect attribute), 134
Blip (class in openpyxl.drawing.fill), 139
blip (openpyxl.drawing.fill.BlipFillProperties attribute), 141
blipFill (openpyxl.drawing.graphic.PictureFrame attribute), 149
blipFill (openpyxl.drawing.text.CharacterProperties attribute), 161
BlipFillProperties (class in openpyxl.drawing.fill), 141
blue (openpyxl.drawing.colors.SystemColor attribute), 130
blueMod (openpyxl.drawing.colors.SystemColor attribute), 130
blueOff (openpyxl.drawing.colors.SystemColor attribute), 130
blur (openpyxl.drawing.effect.EffectList attribute), 133
blur (openpyxl.drawing.fill.Blip attribute), 140
BlurEffect (class in openpyxl.drawing.effect), 133
blurRad (openpyxl.drawing.effect.InnerShadowEffect attribute), 135
blurRad (openpyxl.drawing.effect.OuterShadowEffect attribute), 136
blurRad (openpyxl.drawing.effect.ReflectionEffect attribute), 138
bmk (openpyxl.drawing.text.CharacterProperties attribute), 161
bodyPr (openpyxl.chart.text.RichText attribute), 111
Bool (class in openpyxl.descriptors.base), 123
Border (class in openpyxl.styles.borders), 177
border (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
border (openpyxl.styles.differential.DifferentialStyle attribute), 179
border (openpyxl.styles.named_styles.NamedStyle attribute), 183
border (openpyxl.styles.Style attribute), 176
border_color (openpyxl.drawing.shape.Shape attribute), 153
border_width (openpyxl.drawing.shape.Shape attribute), 153
bottom (openpyxl.formatting.rule.Rule attribute), 172
bottom (openpyxl.styles.borders.Border attribute), 177
bottom (openpyxl.styles.fills.GradientFill attribute), 180
bottom (openpyxl.worksheet.page.PageMargins attribute), 202
BoundDictionary (class in openpyxl.utils.bound_dictionary), 185
bounding_box() (in module openpyxl.drawing.image), 150
br (openpyxl.drawing.text.Paragraph attribute), 165
Break (class in openpyxl.worksheet.pagebreak), 205
BREAK_COLUMN (openpyxl.worksheet.worksheet.Worksheet attribute), 211
BREAK_NONE (openpyxl.worksheet.worksheet.Worksheet attribute), 211
BREAK_ROW (openpyxl.worksheet.worksheet.Worksheet attribute), 211
bright (openpyxl.drawing.effect.LuminanceEffect attribute), 136
brk (openpyxl.worksheet.pagebreak.PageBreak attribute), 205
buAutoNum (openpyxl.drawing.text.ParagraphProperties attribute), 166
bubble3D (openpyxl.chart.bubble_chart.BubbleChart attribute), 89
bubble3D (openpyxl.chart.marker.DataPoint attribute), 102
bubble3D (openpyxl.chart.series.Series attribute), 107
bubble3D (openpyxl.chart.series.XYSeries attribute), 108
BubbleChart (class in openpyxl.chart.bubble_chart), 89

bubbleChart (openpyxl.chart.chartspace.PlotArea attribute), 93
bubbleScale (openpyxl.chart.bubble_chart.BubbleChart attribute), 89
bubbleSize (openpyxl.chart.series.Series attribute), 107
bubbleSize (openpyxl.chart.series.XYSeries attribute), 108
buBlip (openpyxl.drawing.text.ParagraphProperties attribute), 166
buChar (openpyxl.drawing.text.ParagraphProperties attribute), 166
buClr (openpyxl.drawing.text.ParagraphProperties attribute), 166
buClrTx (openpyxl.drawing.text.ParagraphProperties attribute), 166
buFont (openpyxl.drawing.text.ParagraphProperties attribute), 166
buFontTx (openpyxl.drawing.text.ParagraphProperties attribute), 166
builtin_format_code() (in module openpyxl.styles.numbers), 183
builtin_format_id() (in module openpyxl.styles.numbers), 183
builtinId (openpyxl.styles.named_styles.NamedCellStyle attribute), 182
builtinId (openpyxl.styles.named_styles.NamedStyle attribute), 183
builtInUnit (openpyxl.chart.axis.DisplayUnitsLabelList attribute), 84
buNone (openpyxl.drawing.text.ParagraphProperties attribute), 166
buSzPct (openpyxl.drawing.text.ParagraphProperties attribute), 166
buSzPts (openpyxl.drawing.text.ParagraphProperties attribute), 166
buSzTx (openpyxl.drawing.text.ParagraphProperties attribute), 166
bwMode (openpyxl.chart.shapes.GraphicalProperties attribute), 109
bwMode (openpyxl.drawing.graphic.GroupShapeProperties attribute), 146

C

calcCompleted (openpyxl.workbook.properties.CalcProperties attribute), 189
calcId (openpyxl.workbook.properties.CalcProperties attribute), 189
calcMode (openpyxl.workbook.properties.CalcProperties attribute), 189
calcOnSave (openpyxl.workbook.properties.CalcProperties attribute), 189
CalcProperties (class in openpyxl.workbook.properties), 189
calculate_dimension() (openpyxl.worksheet.read_only.ReadOnlyWorksheet method), 208
calculate_dimension() (openpyxl.worksheet.worksheet.Worksheet method), 212
calendarType (openpyxl.worksheet.filters.Filters attribute), 199
Camera (class in openpyxl.drawing.shapes), 153
camera (openpyxl.drawing.shapes.Scene3D attribute), 157
cap (openpyxl.drawing.line.LineProperties attribute), 151
cap (openpyxl.drawing.text.CharacterProperties attribute), 162
caseSensitive (openpyxl.worksheet.filters.SortState attribute), 200
cat (openpyxl.chart.series.Series attribute), 107
catAx (openpyxl.chart.chartspace.PlotArea attribute), 93
Cell (class in openpyxl.cell.cell), 77
cell() (openpyxl.worksheet.worksheet.Worksheet method), 212
cell() (openpyxl.writer.write_only.WriteOnlyWorksheet method), 215
CELL_TAG (openpyxl.reader.worksheet.WorkSheetParser attribute), 175
cellColor (openpyxl.worksheet.filters.ColorFilter attribute), 197
cellComments (openpyxl.worksheet.page.PrintPageSetup attribute), 203
CellCoordinatesException, 186
CellIsRule() (in module openpyxl.formatting.rule), 170
CellRange (class in openpyxl.worksheet.filters), 197
cells (openpyxl.chart.reference.Reference attribute), 105
cellStyle (openpyxl.styles.named_styles.NamedCellStyleList attribute), 182
CENTER (openpyxl.worksheet.header_footer.HeaderFooterItem attribute), 201
center_footer (openpyxl.worksheet.header_footer.HeaderFooter attribute), 201
center_header (openpyxl.worksheet.header_footer.HeaderFooter attribute), 201
cfe (openpyxl.chartsheet.relation.DrawingHF attribute), 119
cff (openpyxl.chartsheet.relation.DrawingHF attribute), 119
cfo (openpyxl.chartsheet.relation.DrawingHF attribute), 119
cfvo (openpyxl.formatting.rule.RuleType attribute), 173
CharacterProperties (class in openpyxl.drawing.text), 161
charset (openpyxl.cell.text.InlineFont attribute), 80
charset (openpyxl.drawing.text.Font attribute), 163
charset (openpyxl.styles.fonts.Font attribute), 181
chart (openpyxl.chart.chartspace.ChartSpace attribute), 91

chart (openpyxl.drawing.graphic.GraphicData attribute), 144
ChartContainer (class in openpyxl.chart.chartspace), 90
ChartLines (class in openpyxl.chart.axis), 82
chartObject (openpyxl.chart.chartspace.Protection attribute), 94
ChartRelation (class in openpyxl.drawing.graphic), 144
Chartsheet (class in openpyxl.chartsheet.chartsheet), 116
Chartsheet() (in module openpyxl.chartsheet.tests.test_chartsheet), 114
ChartsheetProperties (class in openpyxl.chartsheet.properties), 117
ChartsheetProperties() (in module openpyxl.chartsheet.tests.test_properties), 114
ChartsheetProtection (class in openpyxl.chartsheet.protection), 117
ChartsheetProtection() (in module openpyxl.chartsheet.tests.test_protection), 115
chartsheets (openpyxl.workbook.workbook.Workbook attribute), 192
ChartsheetView (class in openpyxl.chartsheet.views), 120
ChartsheetView() (in module openpyxl.chartsheet.tests.test_views), 115
ChartsheetViewList (class in openpyxl.chartsheet.views), 120
ChartsheetViewList() (in module openpyxl.chartsheet.tests.test_views), 115
ChartSpace (class in openpyxl.chart.chartspace), 91
che (openpyxl.chartsheet.relation.DrawingHF attribute), 119
check_error() (openpyxl.cell.cell.Cell method), 77
check_string() (openpyxl.cell.cell.Cell method), 78
checkCompatibility (openpyxl.workbook.properties.WorkbookProperties attribute), 191
chExt (openpyxl.drawing.graphic.GroupTransform2D attribute), 146
chf (openpyxl.chartsheet.relation.DrawingHF attribute), 119
cho (openpyxl.chartsheet.relation.DrawingHF attribute), 119
chOff (openpyxl.drawing.graphic.GroupTransform2D attribute), 146
clientData (openpyxl.drawing.spreadsheet_drawing.AbsoluteAnchor attribute), 159
clientData (openpyxl.drawing.spreadsheet_drawing.OneCellAnchor attribute), 159
clientData (openpyxl.drawing.spreadsheet_drawing.TwoCellAnchor attribute), 160
close() (openpyxl.writer.write_only.WriteOnlyWorksheet method), 215
clrChange (openpyxl.drawing.fill.Blip attribute), 140
clrFrom (openpyxl.drawing.effect.ColorChangeEffect attribute), 133
clrMapOvr (openpyxl.chart.chartspace.ChartSpace attribute), 91
clrRepl (openpyxl.drawing.fill.Blip attribute), 140
clrTo (openpyxl.drawing.effect.ColorChangeEffect attribute), 133
cm_to_dxa() (in module openpyxl.utils.units), 187
cm_to_EMU() (in module openpyxl.utils.units), 187
cmpd (openpyxl.drawing.line.LineProperties attribute), 151
cNvCxnSpPr (openpyxl.drawing.graphic.ConnectorNonVisual attribute), 144
cNvGraphicFramePr (openpyxl.drawing.graphic.NonVisualGraphicFrame attribute), 148
cNvGrpSpPr (openpyxl.drawing.graphic.NonVisualGroupShape attribute), 148
cNvPicPr (openpyxl.drawing.graphic.PictureNonVisual attribute), 150
cNvPr (openpyxl.drawing.graphic.ConnectorNonVisual attribute), 144
cNvPr (openpyxl.drawing.graphic.NonVisualGraphicFrame attribute), 148
cNvPr (openpyxl.drawing.graphic.NonVisualGroupShape attribute), 148
cNvPr (openpyxl.drawing.graphic.PictureNonVisual attribute), 150
codeName (openpyxl.chartsheet.properties.ChartsheetProperties attribute), 117
codeName (openpyxl.workbook.properties.FileVersion attribute), 190
codeName (openpyxl.workbook.properties.WorkbookProperties attribute), 191
codeName (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
col (openpyxl.drawing.spreadsheet_drawing.AnchorMarker attribute), 159
col_idx (openpyxl.cell.cell.Cell attribute), 78
colHidden (openpyxl.comments.properties.Properties attribute), 122
colId (openpyxl.worksheet.filters.FilterColumn attribute), 198
collapse_cell_addresses() (in module openpyxl.worksheet.datavalidation), 194
collapsed (openpyxl.worksheet.dimensions.ColumnDimension attribute), 195
collapsed (openpyxl.worksheet.dimensions.Dimension attribute), 195
collection (openpyxl.styles.styleable.NumberFormatDescriptor attribute), 184
colOff (openpyxl.drawing.spreadsheet_drawing.AnchorMarker attribute), 159
Color (class in openpyxl.drawing.effect), 133
Color (class in openpyxl.styles.colors), 178
color (openpyxl.cell.text.InlineFont attribute), 80

color (openpyxl.drawing.shape.Shape attribute), 153
color (openpyxl.formatting.rule.ColorScale attribute), 170
color (openpyxl.formatting.rule.DataBar attribute), 171
color (openpyxl.styles.borders.Side attribute), 178
color (openpyxl.styles.colors.MRUColorList attribute), 179
color (openpyxl.styles.fonts.Font attribute), 181
ColorChangeEffect (class in openpyxl.drawing.effect), 133
ColorChoice (class in openpyxl.drawing.colors), 128
ColorChoiceDescriptor (class in openpyxl.drawing.colors), 128
ColorDescriptor (class in openpyxl.styles.colors), 179
ColorFilter (class in openpyxl.worksheet.filters), 197
colorFilter (openpyxl.worksheet.filters.FilterColumn attribute), 198
colorId (openpyxl.worksheet.views.SheetView attribute), 209
ColorList (class in openpyxl.styles.colors), 179
ColorMapping (class in openpyxl.drawing.colors), 129
ColorReplaceEffect (class in openpyxl.drawing.effect), 133
ColorScale (class in openpyxl.formatting.rule), 170
colorScale (openpyxl.formatting.rule.Rule attribute), 172
ColorScaleRule() (in module openpyxl.formatting.rule), 170
cols (openpyxl.chart.reference.Reference attribute), 105
cols_from_range() (in module openpyxl.utils), 185
column (openpyxl.cell.cell.Cell attribute), 78
column (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
column_index_from_string() (in module openpyxl.utils), 185
ColumnDimension (class in openpyxl.worksheet.dimensions), 195
columns (openpyxl.worksheet.read_only.ReadOnlyWorksheet attribute), 208
columns (openpyxl.worksheet.worksheet.Worksheet attribute), 212
columnSort (openpyxl.worksheet.filters.SortState attribute), 200
Comment (class in openpyxl.comments.comments), 121
Comment (class in openpyxl.comments.properties), 121
comment (openpyxl.cell.cell.Cell attribute), 78
comment (openpyxl.cell.interface.AbstractCell attribute), 78
comment_writer (openpyxl.writer.excel.ExcelWriter attribute), 214
commentList (openpyxl.comments.properties.CommentSheet attribute), 121
commentPr (openpyxl.comments.properties.Comment attribute), 121
CommentSheet (class in openpyxl.comments.properties), 121
CommentWriter (class in openpyxl.comments.writer), 123
comp (openpyxl.drawing.colors.SystemColor attribute), 130
compatLnSpc (openpyxl.drawing.text.RichTextProperties attribute), 168
concurrentCalc (openpyxl.workbook.properties.CalcProperties attribute), 190
concurrentManualCount (openpyxl.workbook.properties.CalcProperties attribute), 190
condense (openpyxl.cell.text.InlineFont attribute), 80
condense (openpyxl.styles.fonts.Font attribute), 181
ConditionalElement() (in module openpyxl.xml.functions), 216
ConditionalFormatting (class in openpyxl.formatting.formatting), 170
Connection (class in openpyxl.drawing.graphic), 144
ConnectionSite (class in openpyxl.drawing.shapes), 154
ConnectionSiteList (class in openpyxl.drawing.shapes), 154
Connector (class in openpyxl.drawing.graphic), 144
ConnectorLocking (class in openpyxl.drawing.graphic), 144
ConnectorNonVisual (class in openpyxl.drawing.graphic), 144
cont (openpyxl.drawing.effect.AlphaModulateEffect attribute), 132
content (openpyxl.cell.text.Text attribute), 81
content (openpyxl.chartsheet.protection.ChartsheetProtection attribute), 118
content (openpyxl.comments.properties.Comment attribute), 121
contentPart (openpyxl.drawing.spreadsheet_drawing.AbsoluteAnchor attribute), 159
contentPart (openpyxl.drawing.spreadsheet_drawing.OneCellAnchor attribute), 160
contentPart (openpyxl.drawing.spreadsheet_drawing.TwoCellAnchor attribute), 160
ContentType (openpyxl.packaging.manifest.FileExtension attribute), 173
ContentType (openpyxl.packaging.manifest.Override attribute), 173
contourClr (openpyxl.drawing.shapes.Shape3D attribute), 157
contourW (openpyxl.drawing.shapes.Shape3D attribute), 157
contrast (openpyxl.drawing.effect.LuminanceEffect attribute), 136
Convertible (class in openpyxl.descriptors.base), 123
coordinate (openpyxl.cell.cell.Cell attribute), 78
coordinate (openpyxl.cell.interface.AbstractCell attribute), 78

coordinate (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
coordinate_from_string() (in module openpyxl.utils), 185
coordinate_to_tuple() (in module openpyxl.utils), 185
coordinates (openpyxl.drawing.shape.Shape attribute), 153
copies (openpyxl.worksheet.page.PrintPageSetup attribute), 203
copy() (openpyxl.styles.hashable.HashableObject method), 182
copy() (openpyxl.styles.proxy.StyleProxy method), 184
copy() (openpyxl.styles.Style method), 176
count (openpyxl.chartsheet.publish.WebPublishItems attribute), 118
count (openpyxl.descriptors.sequence.NestedSequence attribute), 127
count (openpyxl.drawing.drawing.Drawing attribute), 132
count (openpyxl.styles.named_styles.NamedCellStyleList attribute), 182
count (openpyxl.styles.numbers.NumberFormatList attribute), 183
count (openpyxl.worksheet.datavalidation.DataValidationList attribute), 194
count (openpyxl.worksheet.pagebreak.PageBreak attribute), 205
create_chartsheet() (openpyxl.workbook.workbook.Workbook method), 192
create_named_range() (openpyxl.workbook.workbook.Workbook method), 192
create_sheet() (openpyxl.workbook.workbook.Workbook method), 192
create_temporary_file() (in module openpyxl.writer.write_only), 216
crossAx (openpyxl.chart.axis.DateAxis attribute), 83
crossAx (openpyxl.chart.axis.NumericAxis attribute), 84
crossAx (openpyxl.chart.axis.SeriesAxis attribute), 86
crossAx (openpyxl.chart.axis.TextAxis attribute), 87
crossBetween (openpyxl.chart.axis.NumericAxis attribute), 84
crosses (openpyxl.chart.axis.DateAxis attribute), 83
crosses (openpyxl.chart.axis.NumericAxis attribute), 84
crosses (openpyxl.chart.axis.SeriesAxis attribute), 86
crosses (openpyxl.chart.axis.TextAxis attribute), 87
crossesAt (openpyxl.chart.axis.DateAxis attribute), 83
crossesAt (openpyxl.chart.axis.NumericAxis attribute), 84
crossesAt (openpyxl.chart.axis.SeriesAxis attribute), 86
crossesAt (openpyxl.chart.axis.TextAxis attribute), 87
cs (openpyxl.drawing.text.CharacterProperties attribute), 162
cstate (openpyxl.drawing.fill.Blip attribute), 140
custDash (openpyxl.drawing.line.LineProperties attribute), 151
custGeom (openpyxl.chart.shapes.GraphicalProperties attribute), 109
customBuiltin (openpyxl.styles.named_styles.NamedCellStyle attribute), 182
CustomChartsheetView (class in openpyxl.chartsheet.custom), 116
CustomChartsheetView() (in module openpyxl.chartsheet.tests.test_custom), 114
CustomChartsheetViews (class in openpyxl.chartsheet.custom), 117
CustomChartsheetViews() (in module openpyxl.chartsheet.tests.test_custom), 114
CustomFilter (class in openpyxl.worksheet.filters), 197
customFilter (openpyxl.worksheet.filters.CustomFilters attribute), 197
CustomFilters (class in openpyxl.worksheet.filters), 197
customFilters (openpyxl.worksheet.filters.FilterColumn attribute), 198
customFormat (openpyxl.worksheet.dimensions.RowDimension attribute), 196
CustomGeometry2D (class in openpyxl.drawing.shapes), 154
customHeight (openpyxl.worksheet.dimensions.RowDimension attribute), 196
customList (openpyxl.worksheet.filters.SortCondition attribute), 199
customSheetView (openpyxl.chartsheet.custom.CustomChartsheetViews attribute), 117
customSheetViews (openpyxl.chartsheet.chartsheet.Chartsheet attribute), 116
CustomSplit (class in openpyxl.chart.pie_chart), 103
customWidth (openpyxl.worksheet.dimensions.ColumnDimension attribute), 195
custSplit (openpyxl.chart.pie_chart.ProjectedPieChart attribute), 104
custUnit (openpyxl.chart.axis.DisplayUnitsLabelList attribute), 84
cx (openpyxl.drawing.shapes.PositiveSize2D attribute), 156
cxn (openpyxl.drawing.shapes.ConnectionSiteList attribute), 154
cxnLst (openpyxl.drawing.shapes.CustomGeometry2D attribute), 154
cxnSp (openpyxl.drawing.spreadsheet_drawing.AbsoluteAnchor attribute), 159
cxnSp (openpyxl.drawing.spreadsheet_drawing.OneCellAnchor attribute), 160
cxnSp (openpyxl.drawing.spreadsheet_drawing.TwoCellAnchor attribute), 160
cxnSpLocks (openpyxl.drawing.graphic.NonVisualConnectorProperties

attribute), 147
cy (openpyxl.drawing.shapes.PositiveSize2D attribute), 156

D

d (openpyxl.drawing.line.DashStop attribute), 150
DashStop (class in openpyxl.drawing.line), 150
DashStopList (class in openpyxl.drawing.line), 150
data (openpyxl.chart.chartspace.Protection attribute), 94
data_only (openpyxl.workbook.workbook.Workbook attribute), 192
data_type (openpyxl.cell.cell.Cell attribute), 78
data_type (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
DataBar (class in openpyxl.formatting.rule), 170
dataBar (openpyxl.formatting.rule.Rule attribute), 172
DataBarRule() (in module openpyxl.formatting.rule), 171
DataLabel (class in openpyxl.chart.label), 97
DataLabelList (class in openpyxl.chart.label), 98
DataPoint (class in openpyxl.chart.marker), 102
DataTable (class in openpyxl.chart.chartspace), 91
DataValidation (class in openpyxl.worksheet.datavalidation), 193
dataValidation (openpyxl.worksheet.datavalidation.DataValidationList attribute), 194
DataValidationList (class in openpyxl.worksheet.datavalidation), 194
date1904 (openpyxl.chart.chartspace.ChartSpace attribute), 91
date1904 (openpyxl.workbook.properties.WorkbookProperties attribute), 191
dateAx (openpyxl.chart.chartspace.PlotArea attribute), 93
DateAxis (class in openpyxl.chart.axis), 82
dateCompatibility (openpyxl.workbook.properties.WorkbookProperties attribute), 191
DateGroupItem (class in openpyxl.worksheet.filters), 198
dateGroupItem (openpyxl.worksheet.filters.Filters attribute), 199
DateTime (class in openpyxl.descriptors.base), 123
datetime_to_W3CDTF() (in module openpyxl.utils.datetime), 185
dateTimeGrouping (openpyxl.worksheet.filters.DateGroupItem attribute), 198
day (openpyxl.worksheet.filters.DateGroupItem attribute), 198
days_to_time() (in module openpyxl.utils.datetime), 186
Default (class in openpyxl.descriptors.base), 123
Default (openpyxl.packaging.manifest.Manifest attribute), 173
DEFAULT_HEADER (in module openpyxl.utils.units), 187
defaultGridColor (openpyxl.worksheet.views.SheetView attribute), 209
defaultSize (openpyxl.comments.properties.Properties attribute), 122
defaultThemeVersion (openpyxl.workbook.properties.WorkbookProperties attribute), 191
defPPr (openpyxl.drawing.text.ListStyle attribute), 165
defRPr (openpyxl.drawing.text.ParagraphProperties attribute), 166
defTabSz (openpyxl.drawing.text.ParagraphProperties attribute), 166
degree (openpyxl.styles.fills.GradientFill attribute), 180
degrees_to_angle() (in module openpyxl.utils.units), 187
delete (openpyxl.chart.axis.DateAxis attribute), 83
delete (openpyxl.chart.axis.NumericAxis attribute), 84
delete (openpyxl.chart.axis.SeriesAxis attribute), 86
delete (openpyxl.chart.axis.TextAxis attribute), 87
delete (openpyxl.chart.legend.LegendEntry attribute), 100
deleteColumns (openpyxl.worksheet.protection.SheetProtection attribute), 207
deleteRows (openpyxl.worksheet.protection.SheetProtection attribute), 207
descending (openpyxl.worksheet.filters.SortCondition attribute), 199
descr (openpyxl.drawing.graphic.NonVisualDrawingProps attribute), 147
Descriptor (class in openpyxl.descriptors.base), 123
destinationFile (openpyxl.chartsheet.publish.WebPublishItem attribute), 118
destinations (openpyxl.workbook.names.named_range.NamedRange attribute), 188
detect_external_links() (in module openpyxl.workbook.names.external), 188
diagonal (openpyxl.styles.borders.Border attribute), 177
diagonalDown (openpyxl.styles.borders.Border attribute), 177
diagonalUp (openpyxl.styles.borders.Border attribute), 177
DifferentialStyle (class in openpyxl.styles.differential), 179
Dimension (class in openpyxl.worksheet.dimensions), 195
DimensionHolder (class in openpyxl.worksheet.dimensions), 196
dimensions (openpyxl.worksheet.worksheet.Worksheet attribute), 212
dir (openpyxl.drawing.effect.InnerShadowEffect attribute), 135
dir (openpyxl.drawing.effect.OuterShadowEffect attribute), 136
dir (openpyxl.drawing.effect.PresetShadowEffect attribute), 137

dir (openpyxl.drawing.effect.ReflectionEffect attribute), 138
dir (openpyxl.drawing.shapes.LightRig attribute), 155
dirty (openpyxl.drawing.text.CharacterProperties attribute), 162
disable() (openpyxl.worksheet.protection.SheetProtection method), 207
disabled (openpyxl.comments.properties.Properties attribute), 122
disablePrompts (openpyxl.worksheet.datavalidation.DataValidationList attribute), 194
dispBlanksAs (openpyxl.chart.chartspace.ChartContainer attribute), 90
dispEq (openpyxl.chart.trendline.Trendline attribute), 112
display (openpyxl.worksheet.hyperlink.Hyperlink attribute), 202
DisplayUnitsLabel (class in openpyxl.chart.axis), 83
DisplayUnitsLabelList (class in openpyxl.chart.axis), 84
dispRSqr (openpyxl.chart.trendline.Trendline attribute), 112
dispUnits (openpyxl.chart.axis.NumericAxis attribute), 84
dispUnitsLbl (openpyxl.chart.axis.DisplayUnitsLabelList attribute), 84
dist (openpyxl.drawing.effect.InnerShadowEffect attribute), 135
dist (openpyxl.drawing.effect.OuterShadowEffect attribute), 136
dist (openpyxl.drawing.effect.PresetShadowEffect attribute), 137
dist (openpyxl.drawing.effect.ReflectionEffect attribute), 139
divId (openpyxl.chartsheet.publish.WebPublishItem attribute), 118
dLbl (openpyxl.chart.chartspace.PivotFormat attribute), 92
dLbl (openpyxl.chart.label.DataLabelList attribute), 98
dLblPos (openpyxl.chart.label.DataLabel attribute), 97
dLblPos (openpyxl.chart.label.DataLabelList attribute), 98
dLbls (openpyxl.chart.area_chart.AreaChart attribute), 81
dLbls (openpyxl.chart.area_chart.AreaChart3D attribute), 82
dLbls (openpyxl.chart.bar_chart.BarChart attribute), 88
dLbls (openpyxl.chart.bar_chart.BarChart3D attribute), 88
dLbls (openpyxl.chart.bubble_chart.BubbleChart attribute), 89
dLbls (openpyxl.chart.line_chart.LineChart attribute), 100
dLbls (openpyxl.chart.line_chart.LineChart3D attribute), 101
dLbls (openpyxl.chart.pie_chart.DoughnutChart attribute), 103
dLbls (openpyxl.chart.pie_chart.PieChart attribute), 104
dLbls (openpyxl.chart.pie_chart.PieChart3D attribute), 104
dLbls (openpyxl.chart.pie_chart.ProjectedPieChart attribute), 104
dLbls (openpyxl.chart.radar_chart.RadarChart attribute), 105
dLbls (openpyxl.chart.scatter_chart.ScatterChart attribute), 106
dLbls (openpyxl.chart.series.Series attribute), 107
dLbls (openpyxl.chart.series.XYSeries attribute), 108
dLbls (openpyxl.chart.stock_chart.StockChart attribute), 110
DoughnutChart (class in openpyxl.chart.pie_chart), 103
doughnutChart (openpyxl.chart.chartspace.PlotArea attribute), 93
downBars (openpyxl.chart.updown_bars.UpDownBars attribute), 113
dpi (openpyxl.drawing.fill.BlipFillProperties attribute), 141
dPt (openpyxl.chart.series.Series attribute), 107
dPt (openpyxl.chart.series.XYSeries attribute), 108
draft (openpyxl.worksheet.page.PrintPageSetup attribute), 203
Drawing (class in openpyxl.drawing.drawing), 132
Drawing (class in openpyxl.worksheet.drawing), 196
drawing (openpyxl.chartsheet.chartsheet.Chartsheet attribute), 116
DrawingHF (class in openpyxl.chartsheet.relation), 119
drawingHF (openpyxl.chartsheet.chartsheet.Chartsheet attribute), 116
DrawingHF() (in module openpyxl.chartsheet.tests.test_relation), 115
dropLines (openpyxl.chart.area_chart.AreaChart attribute), 81
dropLines (openpyxl.chart.area_chart.AreaChart3D attribute), 82
dropLines (openpyxl.chart.line_chart.LineChart attribute), 101
dropLines (openpyxl.chart.line_chart.LineChart3D attribute), 101
dropLines (openpyxl.chart.stock_chart.StockChart attribute), 110
ds (openpyxl.drawing.line.DashStopList attribute), 150
dst() (openpyxl.utils.datetime.GMT method), 185
dTable (openpyxl.chart.chartspace.PlotArea attribute), 93
DummyWorkbook (class in openpyxl.chartsheet.tests.test_chartsheet), 114
DummyWorksheet (class in openpyxl.chart.reference), 105
duotone (openpyxl.drawing.fill.Blip attribute), 140
DuotoneEffect (class in openpyxl.drawing.effect), 133
dx (openpyxl.drawing.shapes.Vector3D attribute), 158
dxa_to_cm() (in module openpyxl.utils.units), 187

dxa_to_inch() (in module openpyxl.utils.units), 187
dxf (openpyxl.formatting.rule.Rule attribute), 172
dxfId (openpyxl.formatting.rule.Rule attribute), 172
dxfId (openpyxl.worksheet.filters.ColorFilter attribute), 197
dxfId (openpyxl.worksheet.filters.SortCondition attribute), 199
dy (openpyxl.drawing.shapes.Vector3D attribute), 158
DynamicFilter (class in openpyxl.worksheet.filters), 198
dynamicFilter (openpyxl.worksheet.filters.FilterColumn attribute), 199
dz (openpyxl.drawing.shapes.Vector3D attribute), 158

E

ea (openpyxl.drawing.text.CharacterProperties attribute), 162
eaLnBrk (openpyxl.drawing.text.ParagraphProperties attribute), 166
eb (openpyxl.cell.text.PhoneticText attribute), 80
editAs (openpyxl.drawing.spreadsheet_drawing.TwoCellAnchor attribute), 160
EffectContainer (class in openpyxl.drawing.effect), 133
effectDag (openpyxl.drawing.text.CharacterProperties attribute), 162
EffectList (class in openpyxl.drawing.effect), 133
effectLst (openpyxl.drawing.text.CharacterProperties attribute), 162
effectRef (openpyxl.drawing.shapes.ShapeStyle attribute), 157
embed (openpyxl.drawing.fill.Blip attribute), 140
EmbeddedWAVAudioFile (class in openpyxl.drawing.text), 163
EmptyTag (class in openpyxl.descriptors.nested), 126
EMU_to_cm() (in module openpyxl.utils.units), 187
EMU_to_inch() (in module openpyxl.utils.units), 187
EMU_to_pixels() (in module openpyxl.utils.units), 187
enable() (openpyxl.worksheet.protection.SheetProtection method), 207
enableFormatConditionsCalculation (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
encoding (openpyxl.cell.cell.Cell attribute), 78
encoding (openpyxl.cell.interface.AbstractCell attribute), 78
end (openpyxl.styles.borders.Border attribute), 177
endA (openpyxl.drawing.effect.ReflectionEffect attribute), 139
endCxn (openpyxl.drawing.graphic.NonVisualConnectorProperties attribute), 147
endParaRPr (openpyxl.drawing.text.Paragraph attribute), 165
endPos (openpyxl.drawing.effect.ReflectionEffect attribute), 139
endSnd (openpyxl.drawing.text.Hyperlink attribute), 164
equalAverage (openpyxl.formatting.rule.Rule attribute), 172
err (openpyxl.drawing.text.CharacterProperties attribute), 162
errBars (openpyxl.chart.series.Series attribute), 107
errBars (openpyxl.chart.series.XYSeries attribute), 108
errBarType (openpyxl.chart.error_bar.ErrorBars attribute), 97
errDir (openpyxl.chart.error_bar.ErrorBars attribute), 97
error (openpyxl.worksheet.datavalidation.DataValidation attribute), 193
ERROR_CODES (openpyxl.cell.cell.Cell attribute), 77
ErrorBars (class in openpyxl.chart.error_bar), 97
errors (openpyxl.worksheet.page.PrintPageSetup attribute), 204
errorStyle (openpyxl.worksheet.datavalidation.DataValidation attribute), 193
errorTitle (openpyxl.worksheet.datavalidation.DataValidation attribute), 194
errValType (openpyxl.chart.error_bar.ErrorBars attribute), 97
ExcelWriter (class in openpyxl.writer.excel), 214
expand_cell_ranges() (in module openpyxl.worksheet.datavalidation), 195
expected_type (openpyxl.chart.descriptors.NumberFormatDescriptor attribute), 97
expected_type (openpyxl.chart.title.TitleDescriptor attribute), 112
expected_type (openpyxl.descriptors.base.ASCII attribute), 123
expected_type (openpyxl.descriptors.base.Bool attribute), 123
expected_type (openpyxl.descriptors.base.DateTime attribute), 123
expected_type (openpyxl.descriptors.base.Float attribute), 124
expected_type (openpyxl.descriptors.base.Integer attribute), 124
expected_type (openpyxl.descriptors.base.Max attribute), 124
expected_type (openpyxl.descriptors.base.Min attribute), 124
expected_type (openpyxl.descriptors.base.String attribute), 124
expected_type (openpyxl.descriptors.base.Tuple attribute), 124
expected_type (openpyxl.descriptors.base.Typed attribute), 125
expected_type (openpyxl.descriptors.excel.TextPoint attribute), 125
expected_type (openpyxl.descriptors.sequence.Sequence attribute), 127
expected_type (openpyxl.drawing.colors.ColorChoiceDescriptor attribute), 129

expected_type (openpyxl.styles.colors.ColorDescriptor attribute), 179
expected_type (openpyxl.styles.colors.RGB attribute), 179
expected_type (openpyxl.worksheet.filters.CellRange attribute), 197
explosion (openpyxl.chart.marker.DataPoint attribute), 102
explosion (openpyxl.chart.series.Series attribute), 107
ext (openpyxl.descriptors.excel.ExtensionList attribute), 125
ext (openpyxl.drawing.graphic.GroupTransform2D attribute), 147
ext (openpyxl.drawing.shapes.Transform2D attribute), 158
ext (openpyxl.drawing.spreadsheet_drawing.AbsoluteAnchor attribute), 159
ext (openpyxl.drawing.spreadsheet_drawing.OneCellAnchor attribute), 160
extend (openpyxl.cell.text.InlineFont attribute), 80
extend (openpyxl.styles.fonts.Font attribute), 181
Extension (class in openpyxl.descriptors.excel), 125
Extension (openpyxl.packaging.manifest.FileExtension attribute), 173
ExtensionList (class in openpyxl.descriptors.excel), 125
extensions (openpyxl.packaging.manifest.Manifest attribute), 173
external_range() (in module openpyxl.workbook.names.named_range), 189
ExternalBook (class in openpyxl.workbook.names.external), 188
ExternalData (class in openpyxl.chart.chartspace), 92
externalData (openpyxl.chart.chartspace.ChartSpace attribute), 91
ExternalRange (class in openpyxl.workbook.names.external), 188
extLst (openpyxl.chart.area_chart.AreaChart attribute), 81
extLst (openpyxl.chart.axis.DateAxis attribute), 83
extLst (openpyxl.chart.axis.DisplayUnitsLabelList attribute), 84
extLst (openpyxl.chart.axis.NumericAxis attribute), 84
extLst (openpyxl.chart.axis.Scaling attribute), 85
extLst (openpyxl.chart.axis.SeriesAxis attribute), 86
extLst (openpyxl.chart.axis.TextAxis attribute), 87
extLst (openpyxl.chart.bar_chart.BarChart attribute), 88
extLst (openpyxl.chart.bar_chart.BarChart3D attribute), 88
extLst (openpyxl.chart.bubble_chart.BubbleChart attribute), 89
extLst (openpyxl.chart.chartspace.ChartContainer attribute), 90
extLst (openpyxl.chart.chartspace.ChartSpace attribute), 91
extLst (openpyxl.chart.chartspace.DataTable attribute), 91
extLst (openpyxl.chart.chartspace.PivotFormat attribute), 92
extLst (openpyxl.chart.chartspace.PivotSource attribute), 92
extLst (openpyxl.chart.chartspace.PlotArea attribute), 93
extLst (openpyxl.chart.data_source.NumData attribute), 95
extLst (openpyxl.chart.data_source.NumRef attribute), 95
extLst (openpyxl.chart.data_source.StrData attribute), 96
extLst (openpyxl.chart.data_source.StrRef attribute), 96
extLst (openpyxl.chart.error_bar.ErrorBars attribute), 97
extLst (openpyxl.chart.label.DataLabel attribute), 98
extLst (openpyxl.chart.label.DataLabelList attribute), 98
extLst (openpyxl.chart.layout.Layout attribute), 99
extLst (openpyxl.chart.layout.ManualLayout attribute), 99
extLst (openpyxl.chart.legend.Legend attribute), 100
extLst (openpyxl.chart.legend.LegendEntry attribute), 100
extLst (openpyxl.chart.line_chart.LineChart attribute), 101
extLst (openpyxl.chart.line_chart.LineChart3D attribute), 101
extLst (openpyxl.chart.marker.DataPoint attribute), 102
extLst (openpyxl.chart.marker.Marker attribute), 102
extLst (openpyxl.chart.pie_chart.DoughnutChart attribute), 103
extLst (openpyxl.chart.pie_chart.PieChart attribute), 104
extLst (openpyxl.chart.pie_chart.PieChart3D attribute), 104
extLst (openpyxl.chart.pie_chart.ProjectedPieChart attribute), 104
extLst (openpyxl.chart.radar_chart.RadarChart attribute), 105
extLst (openpyxl.chart.scatter_chart.ScatterChart attribute), 106
extLst (openpyxl.chart.series.Series attribute), 107
extLst (openpyxl.chart.shapes.GraphicalProperties attribute), 109
extLst (openpyxl.chart.stock_chart.StockChart attribute), 110
extLst (openpyxl.chart.surface_chart.SurfaceChart attribute), 111
extLst (openpyxl.chart.surface_chart.SurfaceChart3D attribute), 111
extLst (openpyxl.chart.title.Title attribute), 112
extLst (openpyxl.chart.trendline.Trendline attribute), 112
extLst (openpyxl.chart.trendline.TrendlineLabel attribute), 113
extLst (openpyxl.chart.updown_bars.UpDownBars attribute), 113
extLst (openpyxl.chartsheet.chartsheet.Chartsheet at-

tribute), 116
extLst (openpyxl.chartsheet.views.ChartsheetView attribute), 120
extLst (openpyxl.chartsheet.views.ChartsheetViewList attribute), 120
extLst (openpyxl.comments.properties.CommentSheet attribute), 121
extLst (openpyxl.drawing.colors.ColorMapping attribute), 129
extLst (openpyxl.drawing.fill.Blip attribute), 140
extLst (openpyxl.drawing.graphic.ConnectorLocking attribute), 144
extLst (openpyxl.drawing.graphic.GraphicFrameLocking attribute), 145
extLst (openpyxl.drawing.graphic.GroupLocking attribute), 146
extLst (openpyxl.drawing.graphic.GroupShapeProperties attribute), 146
extLst (openpyxl.drawing.graphic.NonVisualConnectorProperties attribute), 147
extLst (openpyxl.drawing.graphic.NonVisualDrawingProps attribute), 147
extLst (openpyxl.drawing.graphic.NonVisualGraphicFrameProperties attribute), 148
extLst (openpyxl.drawing.graphic.NonVisualGroupDrawingShapeProps attribute), 148
extLst (openpyxl.drawing.graphic.NonVisualPictureProperties attribute), 148
extLst (openpyxl.drawing.graphic.PictureLocking attribute), 149
extLst (openpyxl.drawing.line.LineProperties attribute), 151
extLst (openpyxl.drawing.shapes.Backdrop attribute), 153
extLst (openpyxl.drawing.shapes.Scene3D attribute), 157
extLst (openpyxl.drawing.shapes.Shape3D attribute), 157
extLst (openpyxl.drawing.text.CharacterProperties attribute), 162
extLst (openpyxl.drawing.text.Hyperlink attribute), 164
extLst (openpyxl.drawing.text.ListStyle attribute), 165
extLst (openpyxl.drawing.text.ParagraphProperties attribute), 166
extLst (openpyxl.drawing.text.RichTextProperties attribute), 168
extLst (openpyxl.formatting.rule.FormatObject attribute), 171
extLst (openpyxl.formatting.rule.Rule attribute), 172
extLst (openpyxl.styles.named_styles.NamedCellStyle attribute), 182
extLst (openpyxl.worksheet.filters.AutoFilter attribute), 197
extLst (openpyxl.worksheet.filters.FilterColumn attribute), 199
extLst (openpyxl.worksheet.filters.SortState attribute), 200
extrusionClr (openpyxl.drawing.shapes.Shape3D attribute), 157
extrusionH (openpyxl.drawing.shapes.Shape3D attribute), 157
extrusionOk (openpyxl.drawing.shapes.Path2D attribute), 155

F

f (openpyxl.chart.data_source.NumRef attribute), 96
f (openpyxl.chart.data_source.StrRef attribute), 96
fadeDir (openpyxl.drawing.effect.ReflectionEffect attribute), 139
family (openpyxl.cell.text.InlineFont attribute), 80
family (openpyxl.styles.fonts.Font attribute), 181
fgClr (openpyxl.drawing.fill.PatternFillProperties attribute), 142
fgColor (openpyxl.styles.fills.PatternFill attribute), 180
FileExtension (class in openpyxl.packaging.manifest), 173
filename (openpyxl.writer.write_only.WriteOnlyWorksheet attribute), 215
filenames (openpyxl.packaging.manifest.Manifest attribute), 173
FileVersion (class in openpyxl.workbook.properties), 190
Fill (class in openpyxl.styles.fills), 180
fill (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
fill (openpyxl.drawing.shapes.Path2D attribute), 155
fill (openpyxl.styles.differential.DifferentialStyle attribute), 179
fill (openpyxl.styles.named_styles.NamedStyle attribute), 183
fill (openpyxl.styles.Style attribute), 176
fillOverlay (openpyxl.drawing.effect.EffectList attribute), 133
fillOverlay (openpyxl.drawing.fill.Blip attribute), 140
FillOverlayEffect (class in openpyxl.drawing.effect), 134
fillRect (openpyxl.drawing.fill.StretchInfoProperties attribute), 143
fillRef (openpyxl.drawing.shapes.ShapeStyle attribute), 157
fillToRect (openpyxl.drawing.fill.PathShadeProperties attribute), 142
filter (openpyxl.worksheet.filters.Filters attribute), 199
FilterColumn (class in openpyxl.worksheet.filters), 198
filterColumn (openpyxl.worksheet.filters.AutoFilter attribute), 197
filterMode (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
filterPrivacy (openpyxl.workbook.properties.WorkbookProperties attribute), 191
Filters (class in openpyxl.worksheet.filters), 199
filters (openpyxl.worksheet.filters.FilterColumn attribute), 199

filterVal (openpyxl.worksheet.filters.Top10 attribute), 200
firstPageNumber (openpyxl.worksheet.page.PrintPageSetup attribute), 204
firstSliceAng (openpyxl.chart.pie_chart.DoughnutChart attribute), 103
firstSliceAng (openpyxl.chart.pie_chart.PieChart attribute), 104
fitToHeight (openpyxl.worksheet.page.PrintPageSetup attribute), 204
fitToPage (openpyxl.worksheet.page.PrintPageSetup attribute), 204
fitToPage (openpyxl.worksheet.properties.PageSetupProperties attribute), 205
fitToWidth (openpyxl.worksheet.page.PrintPageSetup attribute), 204
flatten() (in module openpyxl.worksheet.worksheet), 213
flatTx (openpyxl.drawing.text.RichTextProperties attribute), 168
fld (openpyxl.drawing.text.Paragraph attribute), 165
flip (openpyxl.drawing.fill.GradientFillProperties attribute), 141
flip (openpyxl.drawing.fill.TileInfoProperties attribute), 143
flipH (openpyxl.drawing.graphic.GroupTransform2D attribute), 147
flipH (openpyxl.drawing.shapes.Transform2D attribute), 158
flipV (openpyxl.drawing.graphic.GroupTransform2D attribute), 147
flipV (openpyxl.drawing.shapes.Transform2D attribute), 158
Float (class in openpyxl.descriptors.base), 123
fLocksWithSheet (openpyxl.drawing.spreadsheet_drawing.AnchorClientData attribute), 159
floor (openpyxl.chart.bar_chart.BarChart3D attribute), 88
floor (openpyxl.chart.chartspace.ChartContainer attribute), 90
fmla (openpyxl.drawing.shapes.GeomGuide attribute), 154
fmla (openpyxl.drawing.text.GeomGuide attribute), 164
fmtId (openpyxl.chart.chartspace.PivotSource attribute), 92
folHlink (openpyxl.drawing.colors.ColorMapping attribute), 129
Font (class in openpyxl.drawing.text), 163
Font (class in openpyxl.styles.fonts), 181
font (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
font (openpyxl.styles.differential.DifferentialStyle attribute), 179
font (openpyxl.styles.named_styles.NamedStyle attribute), 183
font (openpyxl.styles.Style attribute), 176
font_color (openpyxl.worksheet.header_footer.HeaderFooterItem attribute), 202
FONT_HEIGHT (openpyxl.drawing.shape.Shape attribute), 152
font_name (openpyxl.worksheet.header_footer.HeaderFooterItem attribute), 202
font_size (openpyxl.worksheet.header_footer.HeaderFooterItem attribute), 202
FONT_WIDTH (openpyxl.drawing.shape.Shape attribute), 152
fontAlgn (openpyxl.drawing.text.ParagraphProperties attribute), 166
fontId (openpyxl.cell.text.PhoneticProperties attribute), 80
fontRef (openpyxl.drawing.shapes.ShapeStyle attribute), 158
FontReference (class in openpyxl.drawing.shapes), 154
fontScale (openpyxl.drawing.text.TextNormalAutofit attribute), 170
footer (openpyxl.worksheet.page.PageMargins attribute), 202
forceAA (openpyxl.drawing.text.RichTextProperties attribute), 168
forceFullCalc (openpyxl.workbook.properties.CalcProperties attribute), 190
formatCells (openpyxl.worksheet.protection.SheetProtection attribute), 207
formatCode (openpyxl.chart.data_source.NumData attribute), 95
formatCode (openpyxl.chart.data_source.NumFmt attribute), 95
formatCode (openpyxl.chart.data_source.NumVal attribute), 96
formatCode (openpyxl.styles.numbers.NumberFormat attribute), 183
formatColumns (openpyxl.worksheet.protection.SheetProtection attribute), 207
FormatObject (class in openpyxl.formatting.rule), 171
formatRows (openpyxl.worksheet.protection.SheetProtection attribute), 207
formatting (openpyxl.chart.chartspace.Protection attribute), 94
formula (openpyxl.formatting.rule.Rule attribute), 172
formula1 (openpyxl.worksheet.datavalidation.DataValidation attribute), 194
formula2 (openpyxl.worksheet.datavalidation.DataValidation attribute), 194
FORMULA_TAG (openpyxl.reader.worksheet.WorkSheetParser attribute), 175
FormulaRule() (in module openpyxl.formatting.rule), 171
forward (openpyxl.chart.trendline.Trendline attribute), 112
fov (openpyxl.drawing.shapes.Camera attribute), 153

fPrintsWithSheet (openpyxl.drawing.spreadsheet_drawing.AnchorClientData attribute), 159
fPublished (openpyxl.drawing.graphic.Connector attribute), 144
fPublished (openpyxl.drawing.graphic.GraphicFrame attribute), 145
fPublished (openpyxl.drawing.graphic.PictureFrame attribute), 149
freeze_panes (openpyxl.worksheet.worksheet.Worksheet attribute), 212
from_excel() (in module openpyxl.utils.datetime), 186
from_tree() (openpyxl.descriptors.nested.EmptyTag method), 126
from_tree() (openpyxl.descriptors.nested.Nested method), 126
from_tree() (openpyxl.descriptors.nested.NestedBool method), 126
from_tree() (openpyxl.descriptors.nested.NestedText method), 126
from_tree() (openpyxl.descriptors.sequence.NestedSequence method), 127
from_tree() (openpyxl.descriptors.sequence.ValueSequence method), 127
from_tree() (openpyxl.descriptors.serialisable.Serialisable class method), 127
from_tree() (openpyxl.styles.fills.Fill class method), 180
from_tree() (openpyxl.worksheet.page.PrintPageSetup class method), 204
fromWordArt (openpyxl.drawing.text.RichTextProperties attribute), 168
fullCalcOnLoad (openpyxl.workbook.properties.CalcProperties attribute), 190
fullPrecision (openpyxl.workbook.properties.CalcProperties attribute), 190

G

g (openpyxl.drawing.colors.RGBPercent attribute), 130
gamma (openpyxl.drawing.colors.SystemColor attribute), 131
gapDepth (openpyxl.chart.area_chart.AreaChart3D attribute), 82
gapDepth (openpyxl.chart.bar_chart.BarChart3D attribute), 88
gapDepth (openpyxl.chart.line_chart.LineChart3D attribute), 101
gapWidth (openpyxl.chart.bar_chart.BarChart attribute), 88
gapWidth (openpyxl.chart.bar_chart.BarChart3D attribute), 89
gapWidth (openpyxl.chart.pie_chart.ProjectedPieChart attribute), 104
gapWidth (openpyxl.chart.updown_bars.UpDownBars attribute), 113
gd (openpyxl.drawing.shapes.GeomGuideList attribute), 155
gd (openpyxl.drawing.text.GeomGuideList attribute), 164
gdLst (openpyxl.drawing.shapes.CustomGeometry2D attribute), 154
GeomGuide (class in openpyxl.drawing.shapes), 154
GeomGuide (class in openpyxl.drawing.text), 164
GeomGuideList (class in openpyxl.drawing.shapes), 154
GeomGuideList (class in openpyxl.drawing.text), 164
GeomRect (class in openpyxl.drawing.shapes), 155
get() (openpyxl.worksheet.header_footer.HeaderFooterItem method), 202
get_active_sheet() (openpyxl.workbook.workbook.Workbook method), 192
get_cell_collection() (openpyxl.worksheet.worksheet.Worksheet method), 212
get_column_interval() (in module openpyxl.utils), 185
get_column_letter() (in module openpyxl.utils), 185
get_comments_file() (in module openpyxl.comments.reader), 122
get_dependents() (in module openpyxl.packaging.relationship), 174
get_emu_dimensions() (openpyxl.drawing.drawing.Drawing method), 132
get_index() (openpyxl.workbook.workbook.Workbook method), 192
get_named_range() (openpyxl.workbook.workbook.Workbook method), 192
get_named_range() (openpyxl.worksheet.worksheet.Worksheet method), 212
get_named_ranges() (openpyxl.workbook.workbook.Workbook method), 192
get_rows_to_write() (in module openpyxl.writer.etree_worksheet), 214
get_sheet_by_name() (openpyxl.workbook.workbook.Workbook method), 192
get_sheet_names() (openpyxl.workbook.workbook.Workbook method), 192
get_squared_range() (openpyxl.worksheet.read_only.ReadOnlyWorksheet method), 208
get_squared_range() (openpyxl.worksheet.worksheet.Worksheet method), 212

getFooter() (openpyxl.worksheet.header_footer.HeaderFooter method), 201
getHeader() (openpyxl.worksheet.header_footer.HeaderFooter method), 201
glow (openpyxl.drawing.effect.EffectList attribute), 133
GlowEffect (class in openpyxl.drawing.effect), 134
GMT (class in openpyxl.utils.datetime), 185
gradFill (openpyxl.chart.shapes.GraphicalProperties attribute), 109
gradFill (openpyxl.drawing.line.LineProperties attribute), 151
gradFill (openpyxl.drawing.text.CharacterProperties attribute), 162
GradientFill (class in openpyxl.styles.fills), 180
GradientFillProperties (class in openpyxl.drawing.fill), 141
GradientStop (class in openpyxl.drawing.fill), 142
GradientStopList (class in openpyxl.drawing.fill), 142
graphic (openpyxl.drawing.graphic.GraphicFrame attribute), 145
GraphicalProperties (class in openpyxl.chart.shapes), 109
GraphicData (class in openpyxl.drawing.graphic), 144
graphicData (openpyxl.drawing.graphic.GraphicObject attribute), 145
GraphicFrame (class in openpyxl.drawing.graphic), 145
graphicFrame (openpyxl.drawing.spreadsheet_drawing.AbsoluteAnchor attribute), 159
graphicFrame (openpyxl.drawing.spreadsheet_drawing.OneCellAnchor attribute), 160
graphicFrame (openpyxl.drawing.spreadsheet_drawing.TwoCellAnchor attribute), 160
GraphicFrameLocking (class in openpyxl.drawing.graphic), 145
graphicFrameLocks (openpyxl.drawing.graphic.NonVisualGraphicFrameProperties attribute), 148
GraphicObject (class in openpyxl.drawing.graphic), 145
gray (openpyxl.drawing.colors.SystemColor attribute), 131
GrayscaleEffect (class in openpyxl.drawing.effect), 135
grayscl (openpyxl.drawing.fill.Blip attribute), 140
green (openpyxl.drawing.colors.SystemColor attribute), 131
greenMod (openpyxl.drawing.colors.SystemColor attribute), 131
greenOff (openpyxl.drawing.colors.SystemColor attribute), 131
gridLines (openpyxl.worksheet.page.PrintOptions attribute), 203
gridLinesSet (openpyxl.worksheet.page.PrintOptions attribute), 203
group() (openpyxl.worksheet.dimensions.DimensionHolder method), 196
grouping (openpyxl.chart.area_chart.AreaChart attribute), 81
grouping (openpyxl.chart.area_chart.AreaChart3D attribute), 82
grouping (openpyxl.chart.bar_chart.BarChart attribute), 88
grouping (openpyxl.chart.bar_chart.BarChart3D attribute), 89
grouping (openpyxl.chart.line_chart.LineChart attribute), 101
grouping (openpyxl.chart.line_chart.LineChart3D attribute), 101
GroupLocking (class in openpyxl.drawing.graphic), 145
GroupShape (class in openpyxl.drawing.graphic), 146
GroupShapeProperties (class in openpyxl.drawing.graphic), 146
GroupTransform2D (class in openpyxl.drawing.graphic), 146
grow (openpyxl.drawing.effect.BlurEffect attribute), 133
grpFill (openpyxl.drawing.text.CharacterProperties attribute), 162
grpSp (openpyxl.drawing.spreadsheet_drawing.AbsoluteAnchor attribute), 159
grpSp (openpyxl.drawing.spreadsheet_drawing.OneCellAnchor attribute), 160
grpSp (openpyxl.drawing.spreadsheet_drawing.TwoCellAnchor attribute), 160
grpSpLocks (openpyxl.drawing.graphic.NonVisualGroupDrawingShapePro attribute), 148
grpSpPr (openpyxl.drawing.graphic.GroupShape attribute), 146
gs (openpyxl.drawing.fill.GradientStopList attribute), 142
gsLst (openpyxl.drawing.fill.GradientFillProperties attribute), 142
gte (openpyxl.formatting.rule.FormatObject attribute), 171
guess_types (openpyxl.cell.cell.Cell attribute), 78
guess_types (openpyxl.cell.interface.AbstractCell attribute), 79
Guid (class in openpyxl.descriptors.excel), 125
guid (openpyxl.chartsheet.custom.CustomChartsheetView attribute), 117
guid (openpyxl.comments.properties.Comment attribute), 121

H

h (openpyxl.chart.layout.ManualLayout attribute), 99
h (openpyxl.drawing.shapes.Bevel attribute), 153
h (openpyxl.drawing.shapes.Path2D attribute), 155
hangingPunct (openpyxl.drawing.text.ParagraphProperties attribute), 167
has() (openpyxl.worksheet.header_footer.HeaderFooterItem method), 202
has_style (openpyxl.styles.styleable.StyleableObject attribute), 184

hasFooter() (openpyxl.worksheet.header_footer.HeaderFooter method), 201
hash_password() (in module openpyxl.worksheet.protection), 208
hash_password() (openpyxl.chartsheet.protection.ChartsheetProtection method), 118
HashableObject (class in openpyxl.styles.hashable), 182
hasHeader() (openpyxl.worksheet.header_footer.HeaderFooter method), 201
hashValue (openpyxl.chartsheet.protection.ChartsheetProtection attribute), 118
hashValue (openpyxl.worksheet.protection.SheetProtection attribute), 207
headEnd (openpyxl.drawing.line.LineProperties attribute), 151
header (openpyxl.worksheet.page.PageMargins attribute), 202
HeaderFooter (class in openpyxl.worksheet.header_footer), 200
headerFooter (openpyxl.chart.chartspace.PrintSettings attribute), 94
headerFooter (openpyxl.chartsheet.chartsheet.Chartsheet attribute), 116
headerFooter (openpyxl.chartsheet.custom.CustomChartsheetView attribute), 117
HeaderFooterItem (class in openpyxl.worksheet.header_footer), 201
headings (openpyxl.worksheet.page.PrintOptions attribute), 203
height (openpyxl.drawing.drawing.Drawing attribute), 132
HexBinary (class in openpyxl.descriptors.excel), 125
hidden (openpyxl.drawing.graphic.NonVisualDrawingProps attribute), 147
hidden (openpyxl.styles.named_styles.NamedCellStyle attribute), 182
hidden (openpyxl.styles.named_styles.NamedStyle attribute), 183
hidden (openpyxl.styles.protection.Protection attribute), 184
hidden (openpyxl.worksheet.dimensions.Dimension attribute), 195
hiddenButton (openpyxl.worksheet.filters.FilterColumn attribute), 199
hidePivotFieldList (openpyxl.workbook.properties.WorkbookProperties attribute), 191
highlight (openpyxl.drawing.text.CharacterProperties attribute), 162
highlightClick (openpyxl.drawing.text.Hyperlink attribute), 164
hiLowLines (openpyxl.chart.line_chart.LineChart attribute), 101
hiLowLines (openpyxl.chart.line_chart.LineChart3D attribute), 101
hiLowLines (openpyxl.chart.stock_chart.StockChart attribute), 110
history (openpyxl.drawing.text.Hyperlink attribute), 164
hlink (openpyxl.drawing.colors.ColorMapping attribute), 129
hlinkClick (openpyxl.drawing.graphic.NonVisualDrawingProps attribute), 147
hlinkClick (openpyxl.drawing.text.CharacterProperties attribute), 162
hlinkHover (openpyxl.drawing.graphic.NonVisualDrawingProps attribute), 147
hlinkMouseOver (openpyxl.drawing.text.CharacterProperties attribute), 162
hMode (openpyxl.chart.layout.ManualLayout attribute), 99
holeSize (openpyxl.chart.pie_chart.DoughnutChart attribute), 103
horizontal (openpyxl.styles.alignment.Alignment attribute), 176
horizontal (openpyxl.styles.borders.Border attribute), 177
horizontalCentered (openpyxl.worksheet.page.PrintOptions attribute), 203
horizontalCentered() (openpyxl.worksheet.page.PrintPageSetup method), 204
horizontalDpi (openpyxl.worksheet.page.PrintPageSetup attribute), 204
horzOverflow (openpyxl.drawing.text.RichTextProperties attribute), 168
hour (openpyxl.worksheet.filters.DateGroupItem attribute), 198
hsl (openpyxl.drawing.fill.Blip attribute), 140
hslClr (openpyxl.drawing.colors.ColorChoice attribute), 128
hslClr (openpyxl.drawing.effect.GlowEffect attribute), 134
hslClr (openpyxl.drawing.effect.InnerShadowEffect attribute), 135
hslClr (openpyxl.drawing.effect.OuterShadowEffect attribute), 136
hslClr (openpyxl.drawing.effect.PresetShadowEffect attribute), 137
HSLColor (class in openpyxl.drawing.colors), 130
HSLEffect (class in openpyxl.drawing.effect), 135
ht (openpyxl.worksheet.dimensions.RowDimension attribute), 196
hue (openpyxl.drawing.colors.HSLColor attribute), 130
hue (openpyxl.drawing.colors.SystemColor attribute), 131
hue (openpyxl.drawing.effect.HSLEffect attribute), 135

hue (openpyxl.drawing.effect.TintEffect attribute), 139
hueMod (openpyxl.drawing.colors.SystemColor attribute), 131
hueOff (openpyxl.drawing.colors.SystemColor attribute), 131
Hyperlink (class in openpyxl.drawing.text), 164
Hyperlink (class in openpyxl.worksheet.hyperlink), 202
hyperlink (openpyxl.cell.cell.Cell attribute), 78

I

i (openpyxl.cell.text.InlineFont attribute), 80
i (openpyxl.drawing.text.CharacterProperties attribute), 162
i (openpyxl.styles.fonts.Font attribute), 181
IconFilter (class in openpyxl.worksheet.filters), 199
iconFilter (openpyxl.worksheet.filters.FilterColumn attribute), 199
iconId (openpyxl.worksheet.filters.IconFilter attribute), 199
iconId (openpyxl.worksheet.filters.SortCondition attribute), 200
IconSet (class in openpyxl.formatting.rule), 171
iconSet (openpyxl.formatting.rule.IconSet attribute), 171
iconSet (openpyxl.formatting.rule.Rule attribute), 172
iconSet (openpyxl.worksheet.filters.IconFilter attribute), 199
iconSet (openpyxl.worksheet.filters.SortCondition attribute), 200
IconSetRule() (in module openpyxl.formatting.rule), 171
id (openpyxl.chart.chartspace.ExternalData attribute), 92
id (openpyxl.chartsheet.publish.WebPublishItem attribute), 118
id (openpyxl.chartsheet.relation.DrawingHF attribute), 119
id (openpyxl.chartsheet.relation.SheetBackgroundPicture attribute), 120
id (openpyxl.drawing.graphic.ChartRelation attribute), 144
id (openpyxl.drawing.graphic.Connection attribute), 144
id (openpyxl.drawing.graphic.NonVisualDrawingProps attribute), 147
id (openpyxl.drawing.text.TextField attribute), 169
Id (openpyxl.packaging.relationship.Relationship attribute), 174
Id (openpyxl.workbook.names.external.ExternalBook attribute), 188
id (openpyxl.worksheet.drawing.Drawing attribute), 196
id (openpyxl.worksheet.hyperlink.Hyperlink attribute), 202
id (openpyxl.worksheet.page.PrintPageSetup attribute), 204
id (openpyxl.worksheet.pagebreak.Break attribute), 205
id (openpyxl.worksheet.related.Related attribute), 209
idx (openpyxl.chart.chartspace.PivotFormat attribute), 92
idx (openpyxl.chart.data_source.NumVal attribute), 96
idx (openpyxl.chart.data_source.StrVal attribute), 96
idx (openpyxl.chart.label.DataLabel attribute), 98
idx (openpyxl.chart.legend.LegendEntry attribute), 100
idx (openpyxl.chart.marker.DataPoint attribute), 102
idx (openpyxl.chart.series.Series attribute), 107
idx (openpyxl.chart.series.XYSeries attribute), 108
idx (openpyxl.chart.surface_chart.BandFormat attribute), 110
idx (openpyxl.drawing.graphic.Connection attribute), 144
idx (openpyxl.drawing.shapes.FontReference attribute), 154
idx (openpyxl.drawing.shapes.StyleMatrixReference attribute), 158
idx_base (openpyxl.descriptors.sequence.Sequence attribute), 127
idx_base (openpyxl.descriptors.serialisable.Serialisable attribute), 127
iLevel (openpyxl.styles.named_styles.NamedCellStyle attribute), 182
IllegalCharacterError, 186
Image (class in openpyxl.drawing.image), 150
imeMode (openpyxl.worksheet.datavalidation.DataValidation attribute), 194
inch_to_dxa() (in module openpyxl.utils.units), 187
inch_to_EMU() (in module openpyxl.utils.units), 187
indent (openpyxl.drawing.text.ParagraphProperties attribute), 167
indent (openpyxl.styles.alignment.Alignment attribute), 177
index (openpyxl.styles.colors.Color attribute), 178
index (openpyxl.styles.colors.ColorList attribute), 179
index (openpyxl.worksheet.dimensions.ColumnDimension attribute), 195
index (openpyxl.worksheet.dimensions.Dimension attribute), 195
index() (openpyxl.utils.indexed_list.IndexedList method), 187
indexed (openpyxl.styles.colors.Color attribute), 178
IndexedColorList (class in openpyxl.styles.colors), 179
indexedColors (openpyxl.styles.colors.ColorList attribute), 179
IndexedList (class in openpyxl.utils.indexed_list), 186
INLINE_STRING (openpyxl.reader.worksheet.WorkSheetParser attribute), 175
InlineFont (class in openpyxl.cell.text), 79
InnerShadowEffect (class in openpyxl.drawing.effect), 135
innerShdw (openpyxl.drawing.effect.EffectList attribute), 134
insertColumns (openpyxl.worksheet.protection.SheetProtection attribute), 207
insertHyperlinks (open-

pyxl.worksheet.protection.SheetProtection attribute), 207
insertRows (openpyxl.worksheet.protection.SheetProtection attribute), 207
InsufficientCoordinatesException, 186
Integer (class in openpyxl.descriptors.base), 124
intercept (openpyxl.chart.trendline.Trendline attribute), 113
internal_value (openpyxl.cell.cell.Cell attribute), 78
internal_value (openpyxl.cell.interface.AbstractCell attribute), 79
internal_value (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
inv (openpyxl.drawing.colors.SystemColor attribute), 131
InvalidFileException, 186
invalidUrl (openpyxl.drawing.text.Hyperlink attribute), 164
invertIfNegative (openpyxl.chart.marker.DataPoint attribute), 102
invertIfNegative (openpyxl.chart.series.Series attribute), 107
invertIfNegative (openpyxl.chart.series.XYSeries attribute), 108
invGamma (openpyxl.drawing.colors.SystemColor attribute), 131
is_builtin() (in module openpyxl.styles.numbers), 184
is_date (openpyxl.cell.cell.Cell attribute), 78
is_date (openpyxl.cell.interface.AbstractCell attribute), 79
is_date (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
is_date_format() (in module openpyxl.styles.numbers), 184
isgenerator() (in module openpyxl.worksheet), 193
isgenerator() (in module openpyxl.worksheet.worksheet), 213
isgenerator() (in module openpyxl.writer.write_only), 216
iter_rows() (openpyxl.worksheet.worksheet.Worksheet method), 213
iterate (openpyxl.workbook.properties.CalcProperties attribute), 190
iterateCount (openpyxl.workbook.properties.CalcProperties attribute), 190
iterateDelta (openpyxl.workbook.properties.CalcProperties attribute), 190
iterparse() (in module openpyxl.xml.functions), 216

J

justifyLastLine (openpyxl.styles.alignment.Alignment attribute), 177
justLastX (openpyxl.comments.properties.Properties attribute), 122

K

kern (openpyxl.drawing.text.CharacterProperties attribute), 162
key (openpyxl.styles.hashable.HashableObject attribute), 182
key (openpyxl.styles.styleable.NumberFormatDescriptor attribute), 184
kumimoji (openpyxl.drawing.text.CharacterProperties attribute), 162
kx (openpyxl.drawing.effect.OuterShadowEffect attribute), 136
kx (openpyxl.drawing.effect.ReflectionEffect attribute), 139
ky (openpyxl.drawing.effect.OuterShadowEffect attribute), 136
ky (openpyxl.drawing.effect.ReflectionEffect attribute), 139

L

l (openpyxl.drawing.fill.RelativeRect attribute), 143
l (openpyxl.drawing.shapes.GeomRect attribute), 155
lang (openpyxl.chart.chartspace.ChartSpace attribute), 91
lang (openpyxl.drawing.text.CharacterProperties attribute), 162
lastClr (openpyxl.drawing.colors.SystemColor attribute), 131
lastEdited (openpyxl.workbook.properties.FileVersion attribute), 190
lat (openpyxl.drawing.shapes.SphereCoords attribute), 158
latin (openpyxl.drawing.text.CharacterProperties attribute), 162
latinLnBrk (openpyxl.drawing.text.ParagraphProperties attribute), 167
Layout (class in openpyxl.chart.layout), 99
layout (openpyxl.chart.axis.DisplayUnitsLabel attribute), 84
layout (openpyxl.chart.chartspace.PlotArea attribute), 93
layout (openpyxl.chart.legend.Legend attribute), 100
layout (openpyxl.chart.title.Title attribute), 112
layout (openpyxl.chart.trendline.TrendlineLabel attribute), 113
layoutTarget (openpyxl.chart.layout.ManualLayout attribute), 99
lblAlgn (openpyxl.chart.axis.TextAxis attribute), 87
lblOffset (openpyxl.chart.axis.DateAxis attribute), 83
lblOffset (openpyxl.chart.axis.TextAxis attribute), 87
left (openpyxl.styles.borders.Border attribute), 178
left (openpyxl.styles.fills.GradientFill attribute), 180
LEFT (openpyxl.worksheet.header_footer.HeaderFooterItem attribute), 202
left (openpyxl.worksheet.page.PageMargins attribute), 203

left_footer (openpyxl.worksheet.header_footer.HeaderFooter attribute), 201
left_header (openpyxl.worksheet.header_footer.HeaderFooter attribute), 201
Legend (class in openpyxl.chart.legend), 100
legend (openpyxl.chart.chartspace.ChartContainer attribute), 90
LegendEntry (class in openpyxl.chart.legend), 100
legendEntry (openpyxl.chart.legend.Legend attribute), 100
legendPos (openpyxl.chart.legend.Legend attribute), 100
len (openpyxl.drawing.line.LineEndProperties attribute), 150
Length (class in openpyxl.descriptors.base), 124
lfe (openpyxl.chartsheet.relation.DrawingHF attribute), 119
lff (openpyxl.chartsheet.relation.DrawingHF attribute), 119
lfo (openpyxl.chartsheet.relation.DrawingHF attribute), 119
lhe (openpyxl.chartsheet.relation.DrawingHF attribute), 119
lhf (openpyxl.chartsheet.relation.DrawingHF attribute), 119
lho (openpyxl.chartsheet.relation.DrawingHF attribute), 119
LightRig (class in openpyxl.drawing.shapes), 155
lightRig (openpyxl.drawing.shapes.Scene3D attribute), 157
lim (openpyxl.drawing.line.LineJoinMiterProperties attribute), 151
lin (openpyxl.drawing.fill.GradientFillProperties attribute), 142
line3DChart (openpyxl.chart.chartspace.PlotArea attribute), 93
LinearShadeProperties (class in openpyxl.drawing.fill), 142
LineBreak (class in openpyxl.drawing.text), 164
LineChart (class in openpyxl.chart.line_chart), 100
lineChart (openpyxl.chart.chartspace.PlotArea attribute), 93
LineChart3D (class in openpyxl.chart.line_chart), 101
LineEndProperties (class in openpyxl.drawing.line), 150
LineJoinMiterProperties (class in openpyxl.drawing.line), 151
LineProperties (class in openpyxl.drawing.line), 151
link (openpyxl.drawing.fill.Blip attribute), 140
lIns (openpyxl.drawing.text.RichTextProperties attribute), 168
ListStyle (class in openpyxl.drawing.text), 164
ln (openpyxl.chart.shapes.GraphicalProperties attribute), 109
ln (openpyxl.drawing.text.CharacterProperties attribute), 162
lnRef (openpyxl.drawing.shapes.ShapeStyle attribute), 158
lnSpc (openpyxl.drawing.text.ParagraphProperties attribute), 167
lnSpcReduction (openpyxl.drawing.text.TextNormalAutofit attribute), 170
load_workbook() (in module openpyxl.reader.excel), 174
localname() (in module openpyxl.xml.functions), 216
localSheetId (openpyxl.workbook.names.named_range.NamedValue attribute), 189
location (openpyxl.worksheet.hyperlink.Hyperlink attribute), 202
locked (openpyxl.comments.properties.Properties attribute), 122
locked (openpyxl.styles.protection.Protection attribute), 184
lockText (openpyxl.comments.properties.Properties attribute), 122
logBase (openpyxl.chart.axis.Scaling attribute), 85
lon (openpyxl.drawing.shapes.SphereCoords attribute), 158
lowestEdited (openpyxl.workbook.properties.FileVersion attribute), 190
lstStyle (openpyxl.chart.text.RichText attribute), 111
lum (openpyxl.drawing.colors.HSLColor attribute), 130
lum (openpyxl.drawing.colors.SystemColor attribute), 131
lum (openpyxl.drawing.effect.HSLEffect attribute), 135
lum (openpyxl.drawing.fill.Blip attribute), 140
LuminanceEffect (class in openpyxl.drawing.effect), 136
lumMod (openpyxl.drawing.colors.SystemColor attribute), 131
lumOff (openpyxl.drawing.colors.SystemColor attribute), 131
lvl (openpyxl.drawing.text.ParagraphProperties attribute), 167
lvl1pPr (openpyxl.drawing.text.ListStyle attribute), 165
lvl2pPr (openpyxl.drawing.text.ListStyle attribute), 165
lvl3pPr (openpyxl.drawing.text.ListStyle attribute), 165
lvl4pPr (openpyxl.drawing.text.ListStyle attribute), 165
lvl5pPr (openpyxl.drawing.text.ListStyle attribute), 165
lvl6pPr (openpyxl.drawing.text.ListStyle attribute), 165
lvl7pPr (openpyxl.drawing.text.ListStyle attribute), 165
lvl8pPr (openpyxl.drawing.text.ListStyle attribute), 165
lvl9pPr (openpyxl.drawing.text.ListStyle attribute), 165
lxml_available() (in module openpyxl.xml), 216
lxml_env_set() (in module openpyxl.xml), 216

M

macro (openpyxl.drawing.graphic.Connector attribute), 144
macro (openpyxl.drawing.graphic.GraphicFrame attribute), 145

macro (openpyxl.drawing.graphic.PictureFrame attribute), 149
majorGridlines (openpyxl.chart.axis.DateAxis attribute), 83
majorGridlines (openpyxl.chart.axis.NumericAxis attribute), 85
majorGridlines (openpyxl.chart.axis.SeriesAxis attribute), 86
majorGridlines (openpyxl.chart.axis.TextAxis attribute), 87
majorTickMark (openpyxl.chart.axis.DateAxis attribute), 83
majorTickMark (openpyxl.chart.axis.NumericAxis attribute), 85
majorTickMark (openpyxl.chart.axis.SeriesAxis attribute), 86
majorTickMark (openpyxl.chart.axis.TextAxis attribute), 87
majorTimeUnit (openpyxl.chart.axis.DateAxis attribute), 83
majorUnit (openpyxl.chart.axis.DateAxis attribute), 83
majorUnit (openpyxl.chart.axis.NumericAxis attribute), 85
man (openpyxl.worksheet.pagebreak.Break attribute), 205
Manifest (class in openpyxl.packaging.manifest), 173
manualBreakCount (openpyxl.worksheet.pagebreak.PageBreak attribute), 205
ManualLayout (class in openpyxl.chart.layout), 99
manualLayout (openpyxl.chart.layout.Layout attribute), 99
MARGIN_BOTTOM (openpyxl.drawing.shape.Shape attribute), 152
MARGIN_LEFT (openpyxl.drawing.shape.Shape attribute), 152
Marker (class in openpyxl.chart.marker), 102
marker (openpyxl.chart.chartspace.PivotFormat attribute), 92
marker (openpyxl.chart.line_chart.LineChart attribute), 101
marker (openpyxl.chart.line_chart.LineChart3D attribute), 101
marker (openpyxl.chart.marker.DataPoint attribute), 102
marker (openpyxl.chart.series.Series attribute), 107
marker (openpyxl.chart.series.XYSeries attribute), 108
marL (openpyxl.drawing.text.ParagraphProperties attribute), 167
marR (openpyxl.drawing.text.ParagraphProperties attribute), 167
MatchPattern (class in openpyxl.descriptors.base), 124
Max (class in openpyxl.descriptors.base), 124
max (openpyxl.chart.axis.Scaling attribute), 85
max (openpyxl.chart.descriptors.NestedGapAmount attribute), 97
max (openpyxl.chart.descriptors.NestedOverlap attribute), 97
max (openpyxl.descriptors.excel.TextPoint attribute), 125
max (openpyxl.worksheet.dimensions.ColumnDimension attribute), 195
max (openpyxl.worksheet.pagebreak.Break attribute), 205
max_col (openpyxl.chart.reference.Reference attribute), 106
max_column (openpyxl.worksheet.read_only.ReadOnlyWorksheet attribute), 208
max_column (openpyxl.worksheet.worksheet.Worksheet attribute), 213
max_row (openpyxl.chart.reference.Reference attribute), 106
max_row (openpyxl.worksheet.read_only.ReadOnlyWorksheet attribute), 208
max_row (openpyxl.worksheet.worksheet.Worksheet attribute), 213
maxLength (openpyxl.formatting.rule.DataBar attribute), 171
maxVal (openpyxl.worksheet.filters.DynamicFilter attribute), 198
maxValIso (openpyxl.worksheet.filters.DynamicFilter attribute), 198
merge_cells() (openpyxl.worksheet.worksheet.Worksheet method), 213
merge_cells() (openpyxl.writer.write_only.WriteOnlyWorksheet method), 215
MERGE_TAG (openpyxl.reader.worksheet.WorkSheetParser attribute), 175
merged_cell_ranges (openpyxl.worksheet.worksheet.Worksheet attribute), 213
merged_cells (openpyxl.worksheet.worksheet.Worksheet attribute), 213
MetaSerialisable (class in openpyxl.descriptors), 123
MetaStrict (class in openpyxl.descriptors), 123
Min (class in openpyxl.descriptors.base), 124
min (openpyxl.chart.axis.Scaling attribute), 85
min (openpyxl.chart.descriptors.NestedGapAmount attribute), 97
min (openpyxl.chart.descriptors.NestedOverlap attribute), 97
min (openpyxl.descriptors.excel.TextPoint attribute), 125
min (openpyxl.worksheet.dimensions.ColumnDimension attribute), 195
min (openpyxl.worksheet.pagebreak.Break attribute), 205
min_col (openpyxl.chart.reference.Reference attribute), 106
min_column (openpyxl.worksheet.read_only.ReadOnlyWorksheet attribute), 208
min_column (openpyxl.worksheet.worksheet.Worksheet

attribute), 213
min_row (openpyxl.chart.reference.Reference attribute), 106
min_row (openpyxl.worksheet.read_only.ReadOnlyWorksheet attribute), 208
min_row (openpyxl.worksheet.worksheet.Worksheet attribute), 213
minLength (openpyxl.formatting.rule.DataBar attribute), 171
MinMax (class in openpyxl.descriptors.base), 124
minorGridlines (openpyxl.chart.axis.DateAxis attribute), 83
minorGridlines (openpyxl.chart.axis.NumericAxis attribute), 85
minorGridlines (openpyxl.chart.axis.SeriesAxis attribute), 86
minorGridlines (openpyxl.chart.axis.TextAxis attribute), 87
minorTickMark (openpyxl.chart.axis.DateAxis attribute), 83
minorTickMark (openpyxl.chart.axis.NumericAxis attribute), 85
minorTickMark (openpyxl.chart.axis.SeriesAxis attribute), 86
minorTickMark (openpyxl.chart.axis.TextAxis attribute), 87
minorTimeUnit (openpyxl.chart.axis.DateAxis attribute), 83
minorUnit (openpyxl.chart.axis.DateAxis attribute), 83
minorUnit (openpyxl.chart.axis.NumericAxis attribute), 85
minus (openpyxl.chart.error_bar.ErrorBars attribute), 97
minute (openpyxl.worksheet.filters.DateGroupItem attribute), 198
miter (openpyxl.drawing.line.LineProperties attribute), 151
month (openpyxl.worksheet.filters.DateGroupItem attribute), 198
moveWithCells (openpyxl.comments.properties.ObjectAnchor attribute), 121
MRUColorList (class in openpyxl.styles.colors), 179
mruColors (openpyxl.styles.colors.ColorList attribute), 179

N

name (openpyxl.chart.chartspace.PivotSource attribute), 93
name (openpyxl.chart.trendline.Trendline attribute), 113
name (openpyxl.drawing.effect.EffectContainer attribute), 133
name (openpyxl.drawing.graphic.NonVisualDrawingProps attribute), 147
name (openpyxl.drawing.shapes.GeomGuide attribute), 154
name (openpyxl.drawing.text.EmbeddedWAVAudioFile attribute), 163
name (openpyxl.drawing.text.GeomGuide attribute), 164
name (openpyxl.styles.fonts.Font attribute), 181
name (openpyxl.styles.named_styles.NamedCellStyle attribute), 182
name (openpyxl.workbook.names.external.ExternalRange attribute), 188
name (openpyxl.workbook.names.named_range.NamedRange attribute), 189
name (openpyxl.workbook.names.named_range.NamedValue attribute), 189
NamedCellStyle (class in openpyxl.styles.named_styles), 182
NamedCellStyleList (class in openpyxl.styles.named_styles), 182
NamedRange (class in openpyxl.workbook.names.named_range), 188
NamedRangeContainingValue (in module openpyxl.workbook.names.named_range), 189
NamedRangeException, 186
NamedStyle (class in openpyxl.styles.named_styles), 183
NamedValue (class in openpyxl.workbook.names.named_range), 189
names (openpyxl.styles.named_styles.NamedCellStyleList attribute), 182
namespace (openpyxl.descriptors.excel.Relation attribute), 125
namespace (openpyxl.descriptors.serialisable.Serialisable attribute), 127
namespace (openpyxl.drawing.colors.ColorChoice attribute), 128
namespace (openpyxl.drawing.fill.Blip attribute), 140
namespace (openpyxl.drawing.fill.PatternFillProperties attribute), 142
namespace (openpyxl.drawing.fill.RelativeRect attribute), 143
namespace (openpyxl.drawing.fill.StretchInfoProperties attribute), 143
namespace (openpyxl.drawing.graphic.ChartRelation attribute), 144
namespace (openpyxl.drawing.graphic.GraphicData attribute), 144
namespace (openpyxl.drawing.graphic.GraphicObject attribute), 145
namespace (openpyxl.drawing.graphic.PictureLocking attribute), 149
namespace (openpyxl.drawing.line.DashStop attribute), 150
namespace (openpyxl.drawing.line.LineEndProperties attribute), 150
namespace (openpyxl.drawing.line.LineJoinMiterProperties attribute), 151
namespace (openpyxl.drawing.line.LineProperties

attribute), 151
namespace (openpyxl.drawing.shapes.PresetGeometry2D attribute), 156
namespace (openpyxl.drawing.text.CharacterProperties attribute), 162
namespace (openpyxl.drawing.text.Font attribute), 163
namespace (openpyxl.drawing.text.ListStyle attribute), 165
namespace (openpyxl.drawing.text.Paragraph attribute), 165
namespace (openpyxl.drawing.text.ParagraphProperties attribute), 167
namespace (openpyxl.drawing.text.RegularTextRun attribute), 167
namespace (openpyxl.drawing.text.RichTextProperties attribute), 168
namespaced() (in module openpyxl.descriptors.namespace), 126
Nested (class in openpyxl.descriptors.nested), 126
nested (openpyxl.descriptors.base.Typed attribute), 125
nested (openpyxl.descriptors.nested.Nested attribute), 126
NestedBool (class in openpyxl.descriptors.nested), 126
NestedFloat (class in openpyxl.descriptors.nested), 126
NestedGapAmount (class in openpyxl.chart.descriptors), 96
NestedInteger (class in openpyxl.descriptors.nested), 126
NestedMinMax (class in openpyxl.descriptors.nested), 126
NestedNoneSet (class in openpyxl.descriptors.nested), 126
NestedOverlap (class in openpyxl.chart.descriptors), 97
NestedSequence (class in openpyxl.descriptors.sequence), 127
NestedSet (class in openpyxl.descriptors.nested), 126
NestedString (class in openpyxl.descriptors.nested), 126
NestedText (class in openpyxl.descriptors.nested), 126
NestedValue (class in openpyxl.descriptors.nested), 126
noAdjustHandles (openpyxl.drawing.fill.Blip attribute), 140
noAdjustHandles (openpyxl.drawing.graphic.PictureLocking attribute), 149
noAutofit (openpyxl.drawing.text.RichTextProperties attribute), 168
noChangeArrowheads (openpyxl.drawing.fill.Blip attribute), 141
noChangeArrowheads (openpyxl.drawing.graphic.PictureLocking attribute), 149
noChangeAspect (openpyxl.drawing.fill.Blip attribute), 141
noChangeAspect (openpyxl.drawing.graphic.GraphicFrameLocking attribute), 145
noChangeAspect (openpyxl.drawing.graphic.GroupLocking attribute), 146
noChangeAspect (openpyxl.drawing.graphic.PictureLocking attribute), 149
noChangeShapeType (openpyxl.drawing.fill.Blip attribute), 141
noChangeShapeType (openpyxl.drawing.graphic.PictureLocking attribute), 149
noCrop (openpyxl.drawing.graphic.PictureLocking attribute), 149
noDrilldown (openpyxl.drawing.graphic.GraphicFrameLocking attribute), 145
noEditPoints (openpyxl.drawing.fill.Blip attribute), 141
noEditPoints (openpyxl.drawing.graphic.PictureLocking attribute), 149
noEndCap (openpyxl.chart.error_bar.ErrorBars attribute), 97
noFill (openpyxl.chart.shapes.GraphicalProperties attribute), 109
noFill (openpyxl.drawing.line.LineProperties attribute), 151
noFill (openpyxl.drawing.text.CharacterProperties attribute), 162
noGrp (openpyxl.drawing.fill.Blip attribute), 141
noGrp (openpyxl.drawing.graphic.GraphicFrameLocking attribute), 145
noGrp (openpyxl.drawing.graphic.GroupLocking attribute), 146
noGrp (openpyxl.drawing.graphic.PictureLocking attribute), 149
noMove (openpyxl.drawing.fill.Blip attribute), 141
noMove (openpyxl.drawing.graphic.GraphicFrameLocking attribute), 145
noMove (openpyxl.drawing.graphic.GroupLocking attribute), 146
noMove (openpyxl.drawing.graphic.PictureLocking attribute), 149
noMultiLvlLbl (openpyxl.chart.axis.TextAxis attribute), 87
NoneSet (class in openpyxl.descriptors.base), 124
NonVisualConnectorProperties (class in openpyxl.drawing.graphic), 147
NonVisualDrawingProps (class in openpyxl.drawing.graphic), 147
NonVisualGraphicFrame (class in openpyxl.drawing.graphic), 148
NonVisualGraphicFrameProperties (class in openpyxl.drawing.graphic), 148
NonVisualGroupDrawingShapeProps (class in openpyxl.drawing.graphic), 148

NonVisualGroupShape (class in openpyxl.drawing.graphic), 148
NonVisualPictureProperties (class in openpyxl.drawing.graphic), 148
noProof (openpyxl.drawing.text.CharacterProperties attribute), 162
noResize (openpyxl.drawing.fill.Blip attribute), 141
noResize (openpyxl.drawing.graphic.GraphicFrameLocking attribute), 145
noResize (openpyxl.drawing.graphic.GroupLocking attribute), 146
noResize (openpyxl.drawing.graphic.PictureLocking attribute), 149
norm (openpyxl.drawing.shapes.Backdrop attribute), 153
normalizeH (openpyxl.drawing.text.CharacterProperties attribute), 162
normAutofit (openpyxl.drawing.text.RichTextProperties attribute), 168
noRot (openpyxl.drawing.fill.Blip attribute), 141
noRot (openpyxl.drawing.graphic.GroupLocking attribute), 146
noRot (openpyxl.drawing.graphic.PictureLocking attribute), 150
noSelect (openpyxl.drawing.fill.Blip attribute), 141
noSelect (openpyxl.drawing.graphic.GraphicFrameLocking attribute), 145
noSelect (openpyxl.drawing.graphic.GroupLocking attribute), 146
noSelect (openpyxl.drawing.graphic.PictureLocking attribute), 150
noUngrp (openpyxl.drawing.graphic.GroupLocking attribute), 146
number_format (openpyxl.cell.interface.AbstractCell attribute), 79
number_format (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
number_format (openpyxl.styles.named_styles.NamedStyle attribute), 183
number_format (openpyxl.styles.Style attribute), 176
NumberFormat (class in openpyxl.styles.numbers), 183
NumberFormatDescriptor (class in openpyxl.chart.descriptors), 97
NumberFormatDescriptor (class in openpyxl.styles.numbers), 183
NumberFormatDescriptor (class in openpyxl.styles.styleable), 184
NumberFormatList (class in openpyxl.styles.numbers), 183
numCache (openpyxl.chart.data_source.NumRef attribute), 96
numCol (openpyxl.drawing.text.RichTextProperties attribute), 168
NumData (class in openpyxl.chart.data_source), 95
NumDataSource (class in openpyxl.chart.data_source), 95
NumericAxis (class in openpyxl.chart.axis), 84
NumFmt (class in openpyxl.chart.data_source), 95
numFmt (openpyxl.chart.axis.DateAxis attribute), 83
numFmt (openpyxl.chart.axis.NumericAxis attribute), 85
numFmt (openpyxl.chart.axis.SeriesAxis attribute), 86
numFmt (openpyxl.chart.axis.TextAxis attribute), 87
numFmt (openpyxl.chart.label.DataLabel attribute), 98
numFmt (openpyxl.chart.label.DataLabelList attribute), 98
numFmt (openpyxl.chart.trendline.TrendlineLabel attribute), 113
numFmt (openpyxl.styles.differential.DifferentialStyle attribute), 179
numFmt (openpyxl.styles.numbers.NumberFormatList attribute), 183
numFmtId (openpyxl.styles.numbers.NumberFormat attribute), 183
numLit (openpyxl.chart.data_source.AxDataSource attribute), 95
numLit (openpyxl.chart.data_source.NumDataSource attribute), 95
NumRef (class in openpyxl.chart.data_source), 95
numRef (openpyxl.chart.data_source.AxDataSource attribute), 95
numRef (openpyxl.chart.data_source.NumDataSource attribute), 95
NumVal (class in openpyxl.chart.data_source), 96
nvCxnSpPr (openpyxl.drawing.graphic.Connector attribute), 144
nvGraphicFramePr (openpyxl.drawing.graphic.GraphicFrame attribute), 145
nvGrpSpPr (openpyxl.drawing.graphic.GroupShape attribute), 146
nvPicPr (openpyxl.drawing.graphic.PictureFrame attribute), 149

O

ObjectAnchor (class in openpyxl.comments.properties), 121
objects (openpyxl.chartsheet.protection.ChartsheetProtection attribute), 118
objects (openpyxl.worksheet.protection.SheetProtection attribute), 207
off (openpyxl.drawing.graphic.GroupTransform2D attribute), 147
off (openpyxl.drawing.shapes.Transform2D attribute), 158
offset() (openpyxl.cell.cell.Cell method), 78
offset() (openpyxl.cell.interface.AbstractCell method), 79
ofPieChart (openpyxl.chart.chartspace.PlotArea attribute), 93

ofPieType (openpyxl.chart.pie_chart.ProjectedPieChart attribute), 104
OneCellAnchor (class in openpyxl.drawing.spreadsheet_drawing), 159
oneCellAnchor (openpyxl.drawing.spreadsheet_drawing.SpreadsheetDrawing attribute), 160
openpyxl (module), 1, 77
openpyxl.cell (module), 77
openpyxl.cell.cell (module), 77
openpyxl.cell.interface (module), 78
openpyxl.cell.read_only (module), 79
openpyxl.cell.text (module), 79
openpyxl.chart (module), 81
openpyxl.chart.area_chart (module), 81
openpyxl.chart.axis (module), 82
openpyxl.chart.bar_chart (module), 88
openpyxl.chart.bubble_chart (module), 89
openpyxl.chart.chartspace (module), 90
openpyxl.chart.data_source (module), 95
openpyxl.chart.descriptors (module), 96
openpyxl.chart.error_bar (module), 97
openpyxl.chart.label (module), 97
openpyxl.chart.layout (module), 99
openpyxl.chart.legend (module), 100
openpyxl.chart.line_chart (module), 100
openpyxl.chart.marker (module), 102
openpyxl.chart.picture (module), 103
openpyxl.chart.pie_chart (module), 103
openpyxl.chart.radar_chart (module), 105
openpyxl.chart.reference (module), 105
openpyxl.chart.scatter_chart (module), 106
openpyxl.chart.series (module), 106
openpyxl.chart.series_factory (module), 109
openpyxl.chart.shapes (module), 109
openpyxl.chart.stock_chart (module), 110
openpyxl.chart.surface_chart (module), 110
openpyxl.chart.text (module), 111
openpyxl.chart.title (module), 112
openpyxl.chart.trendline (module), 112
openpyxl.chart.updown_bars (module), 113
openpyxl.chartsheet (module), 114
openpyxl.chartsheet.chartsheet (module), 116
openpyxl.chartsheet.custom (module), 116
openpyxl.chartsheet.properties (module), 117
openpyxl.chartsheet.protection (module), 117
openpyxl.chartsheet.publish (module), 118
openpyxl.chartsheet.relation (module), 119
openpyxl.chartsheet.tests (module), 114
openpyxl.chartsheet.tests.test_chartsheet (module), 114
openpyxl.chartsheet.tests.test_custom (module), 114
openpyxl.chartsheet.tests.test_properties (module), 114
openpyxl.chartsheet.tests.test_protection (module), 115
openpyxl.chartsheet.tests.test_publish (module), 115
openpyxl.chartsheet.tests.test_relation (module), 115
openpyxl.chartsheet.tests.test_views (module), 115
openpyxl.chartsheet.views (module), 120
openpyxl.comments (module), 120
openpyxl.comments.author (module), 120
openpyxl.comments.comments (module), 121
openpyxl.comments.properties (module), 121
openpyxl.comments.reader (module), 122
openpyxl.comments.writer (module), 123
openpyxl.descriptors (module), 123
openpyxl.descriptors.base (module), 123
openpyxl.descriptors.excel (module), 125
openpyxl.descriptors.namespace (module), 126
openpyxl.descriptors.nested (module), 126
openpyxl.descriptors.sequence (module), 127
openpyxl.descriptors.serialisable (module), 127
openpyxl.drawing (module), 128
openpyxl.drawing.colors (module), 128
openpyxl.drawing.drawing (module), 132
openpyxl.drawing.effect (module), 132
openpyxl.drawing.fill (module), 139
openpyxl.drawing.graphic (module), 144
openpyxl.drawing.image (module), 150
openpyxl.drawing.line (module), 150
openpyxl.drawing.shape (module), 152
openpyxl.drawing.shapes (module), 153
openpyxl.drawing.spreadsheet_drawing (module), 158
openpyxl.drawing.text (module), 161
openpyxl.formatting (module), 170
openpyxl.formatting.formatting (module), 170
openpyxl.formatting.rule (module), 170
openpyxl.packaging (module), 173
openpyxl.packaging.manifest (module), 173
openpyxl.packaging.relationship (module), 174
openpyxl.reader (module), 174
openpyxl.reader.excel (module), 174
openpyxl.reader.strings (module), 175
openpyxl.reader.workbook (module), 175
openpyxl.reader.worksheet (module), 175
openpyxl.styles (module), 176
openpyxl.styles.alignment (module), 176
openpyxl.styles.borders (module), 177
openpyxl.styles.colors (module), 178
openpyxl.styles.differential (module), 179
openpyxl.styles.fills (module), 180
openpyxl.styles.fonts (module), 181
openpyxl.styles.hashable (module), 182
openpyxl.styles.named_styles (module), 182
openpyxl.styles.numbers (module), 183
openpyxl.styles.protection (module), 184
openpyxl.styles.proxy (module), 184
openpyxl.styles.styleable (module), 184
openpyxl.utils (module), 185
openpyxl.utils.bound_dictionary (module), 185
openpyxl.utils.datetime (module), 185

openpyxl.utils.exceptions (module), 186
openpyxl.utils.indexed_list (module), 186
openpyxl.utils.units (module), 187
openpyxl.workbook (module), 188
openpyxl.workbook.child (module), 189
openpyxl.workbook.names (module), 188
openpyxl.workbook.names.external (module), 188
openpyxl.workbook.names.named_range (module), 188
openpyxl.workbook.properties (module), 189
openpyxl.workbook.workbook (module), 192
openpyxl.worksheet (module), 193
openpyxl.worksheet.datavalidation (module), 193
openpyxl.worksheet.dimensions (module), 195
openpyxl.worksheet.drawing (module), 196
openpyxl.worksheet.filters (module), 196
openpyxl.worksheet.header_footer (module), 200
openpyxl.worksheet.hyperlink (module), 202
openpyxl.worksheet.page (module), 202
openpyxl.worksheet.pagebreak (module), 205
openpyxl.worksheet.properties (module), 205
openpyxl.worksheet.protection (module), 206
openpyxl.worksheet.read_only (module), 208
openpyxl.worksheet.related (module), 209
openpyxl.worksheet.views (module), 209
openpyxl.worksheet.worksheet (module), 211
openpyxl.writer (module), 214
openpyxl.writer.etree_worksheet (module), 214
openpyxl.writer.excel (module), 214
openpyxl.writer.lxml_worksheet (module), 214
openpyxl.writer.relations (module), 214
openpyxl.writer.strings (module), 214
openpyxl.writer.theme (module), 215
openpyxl.writer.workbook (module), 215
openpyxl.writer.worksheet (module), 215
openpyxl.writer.write_only (module), 215
openpyxl.xml (module), 216
openpyxl.xml.constants (module), 216
openpyxl.xml.functions (module), 216
openpyxl.xml.namespace (module), 216
operator (openpyxl.formatting.rule.Rule attribute), 172
operator (openpyxl.worksheet.datavalidation.DataValidation attribute), 194
operator (openpyxl.worksheet.filters.CustomFilter attribute), 197
options() (openpyxl.worksheet.page.PrintPageSetup method), 204
order (openpyxl.chart.series.Series attribute), 107
order (openpyxl.chart.series.XYSeries attribute), 108
order (openpyxl.chart.trendline.Trendline attribute), 113
orientation (openpyxl.chart.axis.Scaling attribute), 85
orientation (openpyxl.worksheet.page.PrintPageSetup attribute), 204
ORIENTATION_LANDSCAPE (openpyxl.worksheet.worksheet.Worksheet attribute), 211
ORIENTATION_PORTRAIT (openpyxl.worksheet.worksheet.Worksheet attribute), 211
OuterShadowEffect (class in openpyxl.drawing.effect), 136
outerShdw (openpyxl.drawing.effect.EffectList attribute), 134
Outline (class in openpyxl.worksheet.properties), 205
outline (openpyxl.cell.text.InlineFont attribute), 80
outline (openpyxl.styles.borders.Border attribute), 178
outline (openpyxl.styles.fonts.Font attribute), 181
outlineLevel (openpyxl.worksheet.dimensions.Dimension attribute), 195
outlinePr (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
overlap (openpyxl.chart.bar_chart.BarChart attribute), 88
overlay (openpyxl.chart.legend.Legend attribute), 100
overlay (openpyxl.chart.title.Title attribute), 112
Override (class in openpyxl.packaging.manifest), 173
Override (openpyxl.packaging.manifest.Manifest attribute), 173

P

p (openpyxl.chart.text.RichText attribute), 111
PageBreak (class in openpyxl.worksheet.pagebreak), 205
PageMargins (class in openpyxl.worksheet.page), 202
pageMargins (openpyxl.chart.chartspace.PrintSettings attribute), 94
pageMargins (openpyxl.chartsheet.chartsheet.Chartsheet attribute), 116
pageMargins (openpyxl.chartsheet.custom.CustomChartsheetView attribute), 117
pageOrder (openpyxl.worksheet.page.PrintPageSetup attribute), 204
pageSetup (openpyxl.chart.chartspace.PrintSettings attribute), 94
pageSetup (openpyxl.chartsheet.chartsheet.Chartsheet attribute), 116
pageSetup (openpyxl.chartsheet.custom.CustomChartsheetView attribute), 117
pageSetUpPr (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
PageSetupProperties (class in openpyxl.worksheet.properties), 205
Pane (class in openpyxl.worksheet.views), 209
pane (openpyxl.worksheet.views.Selection attribute), 209
pane (openpyxl.worksheet.views.SheetView attribute), 210
panose (openpyxl.drawing.text.Font attribute), 163
paperHeight (openpyxl.worksheet.page.PrintPageSetup attribute), 204
paperSize (openpyxl.worksheet.page.PrintPageSetup attribute), 204

PAPERSIZE_A3 (openpyxl.worksheet.worksheet.Worksheet attribute), 211
PAPERSIZE_A4 (openpyxl.worksheet.worksheet.Worksheet attribute), 211
PAPERSIZE_A4_SMALL (openpyxl.worksheet.worksheet.Worksheet attribute), 211
PAPERSIZE_A5 (openpyxl.worksheet.worksheet.Worksheet attribute), 211
PAPERSIZE_EXECUTIVE (openpyxl.worksheet.worksheet.Worksheet attribute), 211
PAPERSIZE_LEDGER (openpyxl.worksheet.worksheet.Worksheet attribute), 211
PAPERSIZE_LEGAL (openpyxl.worksheet.worksheet.Worksheet attribute), 211
PAPERSIZE_LETTER (openpyxl.worksheet.worksheet.Worksheet attribute), 211
PAPERSIZE_LETTER_SMALL (openpyxl.worksheet.worksheet.Worksheet attribute), 211
PAPERSIZE_STATEMENT (openpyxl.worksheet.worksheet.Worksheet attribute), 211
PAPERSIZE_TABLOID (openpyxl.worksheet.worksheet.Worksheet attribute), 211
paperWidth (openpyxl.worksheet.page.PrintPageSetup attribute), 204
Paragraph (class in openpyxl.drawing.text), 165
ParagraphProperties (class in openpyxl.drawing.text), 165
parent (openpyxl.cell.cell.Cell attribute), 78
parent (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
parent (openpyxl.comments.comments.Comment attribute), 121
parent (openpyxl.styles.styleable.StyleableObject attribute), 184
parse() (openpyxl.reader.worksheet.WorkSheetParser method), 175
parse_auto_filter() (openpyxl.reader.worksheet.WorkSheetParser method), 175
parse_books() (in module openpyxl.workbook.names.external), 188
parse_cell() (openpyxl.reader.worksheet.WorkSheetParser method), 175
parse_column_dimensions() (openpyxl.reader.worksheet.WorkSheetParser method), 175
parse_data_validation() (openpyxl.reader.worksheet.WorkSheetParser method), 175
parse_extensions() (openpyxl.reader.worksheet.WorkSheetParser method), 175
parse_header_footer() (openpyxl.reader.worksheet.WorkSheetParser method), 175
parse_legacy_drawing() (openpyxl.reader.worksheet.WorkSheetParser method), 175
parse_margins() (openpyxl.reader.worksheet.WorkSheetParser method), 175
parse_merge() (openpyxl.reader.worksheet.WorkSheetParser method), 176
parse_page_setup() (openpyxl.reader.worksheet.WorkSheetParser method), 176
parse_print_options() (openpyxl.reader.worksheet.WorkSheetParser method), 176
parse_properties() (openpyxl.reader.worksheet.WorkSheetParser method), 176
parse_ranges() (in module openpyxl.workbook.names.external), 188
parse_row_dimensions() (openpyxl.reader.worksheet.WorkSheetParser method), 176
parse_sheet_protection() (openpyxl.reader.worksheet.WorkSheetParser method), 176
parse_sheet_views() (openpyxl.reader.worksheet.WorkSheetParser method), 176
parse_sort() (openpyxl.reader.worksheet.WorkSheetParser method), 176
parser_conditional_formatting() (openpyxl.reader.worksheet.WorkSheetParser method), 176
PartName (openpyxl.packaging.manifest.Override attribute), 173
path (openpyxl.drawing.fill.GradientFillProperties attribute), 142
path (openpyxl.drawing.fill.PathShadeProperties attribute), 142
path (openpyxl.drawing.shapes.Path2DList attribute), 155
Path2D (class in openpyxl.drawing.shapes), 155
Path2DList (class in openpyxl.drawing.shapes), 155
pathLst (openpyxl.drawing.shapes.CustomGeometry2D

attribute), 154
PathShadeProperties (class in openpyxl.drawing.fill), 142
pattern (openpyxl.descriptors.excel.Base64Binary attribute), 125
pattern (openpyxl.descriptors.excel.Guid attribute), 125
pattern (openpyxl.descriptors.excel.HexBinary attribute), 125
pattern (openpyxl.descriptors.excel.Percentage attribute), 125
pattern (openpyxl.descriptors.excel.UniversalMeasure attribute), 126
pattern (openpyxl.worksheet.filters.CellRange attribute), 197
PatternFill (class in openpyxl.styles.fills), 180
PatternFillProperties (class in openpyxl.drawing.fill), 142
patternType (openpyxl.styles.fills.PatternFill attribute), 180
pattFill (openpyxl.chart.shapes.GraphicalProperties attribute), 109
pattFill (openpyxl.drawing.line.LineProperties attribute), 151
pattFill (openpyxl.drawing.text.CharacterProperties attribute), 163
percent (openpyxl.formatting.rule.IconSet attribute), 171
percent (openpyxl.formatting.rule.Rule attribute), 172
percent (openpyxl.worksheet.filters.Top10 attribute), 200
Percentage (class in openpyxl.descriptors.excel), 125
period (openpyxl.chart.trendline.Trendline attribute), 113
phoneticPr (openpyxl.cell.text.Text attribute), 81
PhoneticProperties (class in openpyxl.cell.text), 80
PhoneticText (class in openpyxl.cell.text), 80
pic (openpyxl.drawing.spreadsheet_drawing.AbsoluteAnchor attribute), 159
pic (openpyxl.drawing.spreadsheet_drawing.OneCellAnchor attribute), 160
pic (openpyxl.drawing.spreadsheet_drawing.TwoCellAnchor attribute), 160
picLocks (openpyxl.drawing.graphic.NonVisualPictureProperties attribute), 148
picture (openpyxl.chartsheet.chartsheet.Chartsheet attribute), 116
pictureFormat (openpyxl.chart.picture.PictureOptions attribute), 103
PictureFrame (class in openpyxl.drawing.graphic), 148
PictureLocking (class in openpyxl.drawing.graphic), 149
PictureNonVisual (class in openpyxl.drawing.graphic), 150
PictureOptions (class in openpyxl.chart.picture), 103
pictureOptions (openpyxl.chart.marker.DataPoint attribute), 102
pictureOptions (openpyxl.chart.series.Series attribute), 107
pictureStackUnit (openpyxl.chart.picture.PictureOptions attribute), 103
pie3DChart (openpyxl.chart.chartspace.PlotArea attribute), 93
PieChart (class in openpyxl.chart.pie_chart), 104
pieChart (openpyxl.chart.chartspace.PlotArea attribute), 94
PieChart3D (class in openpyxl.chart.pie_chart), 104
pitchFamily (openpyxl.drawing.text.Font attribute), 164
pivotButton (openpyxl.styles.styleable.StyleableObject attribute), 184
pivotFmt (openpyxl.chart.chartspace.PivotFormatList attribute), 92
pivotFmts (openpyxl.chart.chartspace.ChartContainer attribute), 90
PivotFormat (class in openpyxl.chart.chartspace), 92
PivotFormatList (class in openpyxl.chart.chartspace), 92
PivotSource (class in openpyxl.chart.chartspace), 92
pivotSource (openpyxl.chart.chartspace.ChartSpace attribute), 91
pivotTables (openpyxl.worksheet.protection.SheetProtection attribute), 207
pixels_to_EMU() (in module openpyxl.utils.units), 187
pixels_to_points() (in module openpyxl.utils.units), 187
PlotArea (class in openpyxl.chart.chartspace), 93
plotArea (openpyxl.chart.chartspace.ChartContainer attribute), 90
plotVisOnly (openpyxl.chart.chartspace.ChartContainer attribute), 90
plus (openpyxl.chart.error_bar.ErrorBars attribute), 97
Point2D (class in openpyxl.drawing.shapes), 155
Point3D (class in openpyxl.drawing.shapes), 156
point_pos() (openpyxl.worksheet.worksheet.Worksheet method), 213
points_to_pixels() (in module openpyxl.utils.units), 187
pop() (openpyxl.chart.reference.Reference method), 106
pos (openpyxl.drawing.fill.GradientStop attribute), 142
pos (openpyxl.drawing.shapes.ConnectionSite attribute), 154
pos (openpyxl.drawing.spreadsheet_drawing.AbsoluteAnchor attribute), 159
pos (openpyxl.drawing.text.TabStop attribute), 169
PositiveSize2D (class in openpyxl.drawing.shapes), 156
pPr (openpyxl.drawing.text.Paragraph attribute), 165
pPr (openpyxl.drawing.text.TextField attribute), 169
preferRelativeResize (openpyxl.drawing.graphic.NonVisualPictureProperties attribute), 148
PresetGeometry2D (class in openpyxl.drawing.shapes), 156
PresetShadowEffect (class in openpyxl.drawing.effect), 137
PresetTextShape (class in openpyxl.drawing.text), 167
PrintOptions (class in openpyxl.worksheet.page), 203
PrintPageSetup (class in openpyxl.worksheet.page), 203
PrintSettings (class in openpyxl.chart.chartspace), 94

printSettings (openpyxl.chart.chartspace.ChartSpace attribute), 91
priority (openpyxl.formatting.rule.Rule attribute), 172
ProjectedPieChart (class in openpyxl.chart.pie_chart), 104
prompt (openpyxl.worksheet.datavalidation.DataValidation attribute), 194
promptedSolutions (openpyxl.workbook.properties.WorkbookProperties attribute), 191
promptTitle (openpyxl.worksheet.datavalidation.DataValidation attribute), 194
Properties (class in openpyxl.comments.properties), 122
Protection (class in openpyxl.chart.chartspace), 94
Protection (class in openpyxl.styles.protection), 184
protection (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
protection (openpyxl.chart.chartspace.ChartSpace attribute), 91
protection (openpyxl.styles.differential.DifferentialStyle attribute), 179
protection (openpyxl.styles.named_styles.NamedStyle attribute), 183
protection (openpyxl.styles.Style attribute), 176
prst (openpyxl.drawing.effect.PresetShadowEffect attribute), 137
prst (openpyxl.drawing.fill.PatternFillProperties attribute), 142
prst (openpyxl.drawing.shapes.Bevel attribute), 153
prst (openpyxl.drawing.shapes.Camera attribute), 154
prst (openpyxl.drawing.shapes.PresetGeometry2D attribute), 156
prst (openpyxl.drawing.text.PresetTextShape attribute), 167
prstClr (openpyxl.drawing.colors.ColorChoice attribute), 128
prstClr (openpyxl.drawing.effect.GlowEffect attribute), 134
prstClr (openpyxl.drawing.effect.InnerShadowEffect attribute), 135
prstClr (openpyxl.drawing.effect.OuterShadowEffect attribute), 136
prstClr (openpyxl.drawing.effect.PresetShadowEffect attribute), 138
prstDash (openpyxl.drawing.line.LineProperties attribute), 151
prstGeom (openpyxl.chart.shapes.GraphicalProperties attribute), 109
prstMaterial (openpyxl.drawing.shapes.Shape3D attribute), 157
prstShdw (openpyxl.drawing.effect.EffectList attribute), 134
prstTxWarp (openpyxl.drawing.text.RichTextProperties attribute), 168
pt (openpyxl.chart.data_source.NumData attribute), 95
pt (openpyxl.chart.data_source.StrData attribute), 96
pt (openpyxl.worksheet.pagebreak.Break attribute), 205
ptCount (openpyxl.chart.data_source.NumData attribute), 95
ptCount (openpyxl.chart.data_source.StrData attribute), 96
published (openpyxl.chartsheet.properties.ChartsheetProperties attribute), 117
published (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
publishItems (openpyxl.workbook.properties.WorkbookProperties attribute), 191

Q

quote_sheetname() (in module openpyxl.utils), 185
quotePrefix (openpyxl.styles.styleable.StyleableObject attribute), 184

R

r (openpyxl.cell.text.Text attribute), 81
r (openpyxl.drawing.colors.RGBPercent attribute), 130
r (openpyxl.drawing.fill.RelativeRect attribute), 143
r (openpyxl.drawing.shapes.GeomRect attribute), 155
r (openpyxl.drawing.text.Paragraph attribute), 165
rad (openpyxl.drawing.effect.BlurEffect attribute), 133
rad (openpyxl.drawing.effect.GlowEffect attribute), 134
rad (openpyxl.drawing.effect.SoftEdgesEffect attribute), 139
RadarChart (class in openpyxl.chart.radar_chart), 105
radarChart (openpyxl.chart.chartspace.PlotArea attribute), 94
radarStyle (openpyxl.chart.radar_chart.RadarChart attribute), 105
range() (openpyxl.writer.write_only.WriteOnlyWorksheet method), 215
range_boundaries() (in module openpyxl.utils), 185
range_string (openpyxl.chart.reference.Reference attribute), 106
range_to_tuple() (in module openpyxl.utils), 185
rank (openpyxl.formatting.rule.Rule attribute), 172
read_comments() (in module openpyxl.comments.reader), 122
read_content_types() (in module openpyxl.reader.workbook), 175
read_dimension() (in module openpyxl.worksheet.read_only), 208
read_named_ranges() (in module openpyxl.workbook.names.named_range), 189
read_only (openpyxl.workbook.workbook.Workbook attribute), 192
read_rels() (in module openpyxl.reader.workbook), 175
read_sheets() (in module openpyxl.reader.workbook), 175

read_string_table() (in module openpyxl.reader.strings), 175
readingOrder (openpyxl.styles.alignment.Alignment attribute), 177
ReadOnlyCell (class in openpyxl.cell.read_only), 79
ReadOnlyWorkbookException, 186
ReadOnlyWorksheet (class in openpyxl.worksheet.read_only), 208
RECT (openpyxl.drawing.shape.Shape attribute), 152
rect (openpyxl.drawing.shapes.CustomGeometry2D attribute), 154
red (openpyxl.drawing.colors.SystemColor attribute), 131
redMod (openpyxl.drawing.colors.SystemColor attribute), 131
redOff (openpyxl.drawing.colors.SystemColor attribute), 131
ref (openpyxl.comments.properties.Comment attribute), 121
ref (openpyxl.worksheet.filters.AutoFilter attribute), 197
ref (openpyxl.worksheet.filters.SortCondition attribute), 200
ref (openpyxl.worksheet.filters.SortState attribute), 200
ref (openpyxl.worksheet.hyperlink.Hyperlink attribute), 202
Reference (class in openpyxl.chart.reference), 105
refers_to_range() (in module openpyxl.workbook.names.named_range), 189
refersTo (openpyxl.workbook.names.external.ExternalRange attribute), 188
reflection (openpyxl.drawing.effect.EffectList attribute), 134
ReflectionEffect (class in openpyxl.drawing.effect), 138
refMode (openpyxl.workbook.properties.CalcProperties attribute), 190
refreshAllConnections (openpyxl.workbook.properties.WorkbookProperties attribute), 191
RegularTextRun (class in openpyxl.drawing.text), 167
Related (class in openpyxl.worksheet.related), 209
Relation (class in openpyxl.descriptors.excel), 125
Relationship (class in openpyxl.packaging.relationship), 174
Relationship (openpyxl.packaging.relationship.RelationshipList attribute), 174
RelationshipList (class in openpyxl.packaging.relationship), 174
relativeIndent (openpyxl.styles.alignment.Alignment attribute), 177
RelativeRect (class in openpyxl.drawing.fill), 143
RelId (class in openpyxl.chart.chartspace), 95
remove_named_range() (openpyxl.workbook.workbook.Workbook method), 193
remove_sheet() (openpyxl.workbook.workbook.Workbook method), 193
removed_method() (in module openpyxl.writer.write_only), 216
repair_central_directory() (in module openpyxl.reader.excel), 175
REPLACE_LIST (openpyxl.worksheet.header_footer.HeaderFooterItem attribute), 202
repr_format (openpyxl.workbook.names.named_range.NamedRange attribute), 189
rev (openpyxl.drawing.shapes.SphereCoords attribute), 158
reverse (openpyxl.formatting.rule.IconSet attribute), 171
rfe (openpyxl.chartsheet.relation.DrawingHF attribute), 119
rff (openpyxl.chartsheet.relation.DrawingHF attribute), 119
rfo (openpyxl.chartsheet.relation.DrawingHF attribute), 119
rFont (openpyxl.cell.text.InlineFont attribute), 80
RGB (class in openpyxl.styles.colors), 179
rgb (openpyxl.styles.colors.Color attribute), 178
rgb (openpyxl.styles.colors.RgbColor attribute), 179
RgbColor (class in openpyxl.styles.colors), 179
rgbColor (openpyxl.styles.colors.IndexedColorList attribute), 179
RGBPercent (class in openpyxl.drawing.colors), 130
rhe (openpyxl.chartsheet.relation.DrawingHF attribute), 119
rhf (openpyxl.chartsheet.relation.DrawingHF attribute), 119
rho (openpyxl.chartsheet.relation.DrawingHF attribute), 120
rich (openpyxl.chart.text.Text attribute), 112
RichText (class in openpyxl.cell.text), 81
RichText (class in openpyxl.chart.text), 111
RichTextProperties (class in openpyxl.drawing.text), 167
rig (openpyxl.drawing.shapes.LightRig attribute), 155
right (openpyxl.styles.borders.Border attribute), 178
right (openpyxl.styles.fills.GradientFill attribute), 180
RIGHT (openpyxl.worksheet.header_footer.HeaderFooterItem attribute), 202
right (openpyxl.worksheet.page.PageMargins attribute), 203
right_footer (openpyxl.worksheet.header_footer.HeaderFooter attribute), 201
right_header (openpyxl.worksheet.header_footer.HeaderFooter attribute), 201
rightToLeft (openpyxl.worksheet.views.SheetView attribute), 210
rIns (openpyxl.drawing.text.RichTextProperties attribute), 168
rot (openpyxl.drawing.graphic.GroupTransform2D attribute), 147

rot (openpyxl.drawing.shapes.Camera attribute), 154
rot (openpyxl.drawing.shapes.LightRig attribute), 155
rot (openpyxl.drawing.shapes.Transform2D attribute), 158
rot (openpyxl.drawing.text.RichTextProperties attribute), 168
rotWithShape (openpyxl.drawing.effect.OuterShadowEffect attribute), 137
rotWithShape (openpyxl.drawing.effect.ReflectionEffect attribute), 139
rotWithShape (openpyxl.drawing.fill.BlipFillProperties attribute), 141
rotWithShape (openpyxl.drawing.fill.GradientFillProperties attribute), 142
round (openpyxl.drawing.line.LineProperties attribute), 151
ROUND_RECT (openpyxl.drawing.shape.Shape attribute), 153
roundedCorners (openpyxl.chart.chartspace.ChartSpace attribute), 91
row (openpyxl.cell.cell.Cell attribute), 78
row (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
row (openpyxl.drawing.spreadsheet_drawing.AnchorMarker attribute), 159
RowDimension (class in openpyxl.worksheet.dimensions), 196
rowHidden (openpyxl.comments.properties.Properties attribute), 122
rowOff (openpyxl.drawing.spreadsheet_drawing.AnchorMarker attribute), 159
rows (openpyxl.chart.reference.Reference attribute), 106
rows (openpyxl.worksheet.read_only.ReadOnlyWorksheet attribute), 208
rows (openpyxl.worksheet.worksheet.Worksheet attribute), 213
rows_from_range() (in module openpyxl.utils), 185
rPh (openpyxl.cell.text.Text attribute), 81
rPr (openpyxl.cell.text.RichText attribute), 81
rPr (openpyxl.drawing.text.LineBreak attribute), 164
rPr (openpyxl.drawing.text.RegularTextRun attribute), 167
rPr (openpyxl.drawing.text.TextField attribute), 170
rtl (openpyxl.drawing.text.CharacterProperties attribute), 163
rtl (openpyxl.drawing.text.ParagraphProperties attribute), 167
rtlCol (openpyxl.drawing.text.RichTextProperties attribute), 168
Rule (class in openpyxl.formatting.rule), 172
RuleType (class in openpyxl.formatting.rule), 173
rupBuild (openpyxl.workbook.properties.FileVersion attribute), 190

S

safe_iterator() (in module openpyxl.xml.functions), 216
safe_iterparse() (in module openpyxl.xml.functions), 216
saltValue (openpyxl.chartsheet.protection.ChartsheetProtection attribute), 118
saltValue (openpyxl.worksheet.protection.SheetProtection attribute), 207
sat (openpyxl.drawing.colors.HSLColor attribute), 130
sat (openpyxl.drawing.colors.SystemColor attribute), 131
sat (openpyxl.drawing.effect.HSLEffect attribute), 135
satMod (openpyxl.drawing.colors.SystemColor attribute), 131
satOff (openpyxl.drawing.colors.SystemColor attribute), 131
save() (openpyxl.workbook.workbook.Workbook method), 193
save() (openpyxl.writer.excel.ExcelWriter method), 214
save_dump() (in module openpyxl.writer.write_only), 216
save_virtual_workbook() (in module openpyxl.writer.excel), 214
save_workbook() (in module openpyxl.writer.excel), 214
saveExternalLinkValues (openpyxl.workbook.properties.WorkbookProperties attribute), 191
sb (openpyxl.cell.text.PhoneticText attribute), 80
scale (openpyxl.chartsheet.custom.CustomChartsheetView attribute), 117
scale (openpyxl.worksheet.page.PrintPageSetup attribute), 204
scaled (openpyxl.drawing.fill.LinearShadeProperties attribute), 142
Scaling (class in openpyxl.chart.axis), 85
scaling (openpyxl.chart.axis.DateAxis attribute), 83
scaling (openpyxl.chart.axis.NumericAxis attribute), 85
scaling (openpyxl.chart.axis.SeriesAxis attribute), 86
scaling (openpyxl.chart.axis.TextAxis attribute), 87
ScatterChart (class in openpyxl.chart.scatter_chart), 106
scatterChart (openpyxl.chart.chartspace.PlotArea attribute), 94
scatterStyle (openpyxl.chart.scatter_chart.ScatterChart attribute), 106
scenarios (openpyxl.worksheet.protection.SheetProtection attribute), 208
Scene3D (class in openpyxl.drawing.shapes), 157
scene3d (openpyxl.chart.shapes.GraphicalProperties attribute), 109
scene3d (openpyxl.drawing.graphic.GroupShapeProperties attribute), 146
scene3d (openpyxl.drawing.text.RichTextProperties attribute), 169
scheme (openpyxl.cell.text.InlineFont attribute), 80
scheme (openpyxl.styles.fonts.Font attribute), 181

schemeClr (openpyxl.drawing.colors.ColorChoice attribute), 128
schemeClr (openpyxl.drawing.effect.GlowEffect attribute), 134
schemeClr (openpyxl.drawing.effect.InnerShadowEffect attribute), 136
schemeClr (openpyxl.drawing.effect.OuterShadowEffect attribute), 137
schemeClr (openpyxl.drawing.effect.PresetShadowEffect attribute), 138
scope (openpyxl.workbook.names.named_range.NamedRange attribute), 189
scope (openpyxl.workbook.names.named_range.NamedValue attribute), 189
scrgbClr (openpyxl.drawing.colors.ColorChoice attribute), 128
scrgbClr (openpyxl.drawing.effect.GlowEffect attribute), 135
scrgbClr (openpyxl.drawing.effect.InnerShadowEffect attribute), 136
scrgbClr (openpyxl.drawing.effect.OuterShadowEffect attribute), 137
scrgbClr (openpyxl.drawing.effect.PresetShadowEffect attribute), 138
second (openpyxl.worksheet.filters.DateGroupItem attribute), 198
secondPiePt (openpyxl.chart.pie_chart.CustomSplit attribute), 103
secondPieSize (openpyxl.chart.pie_chart.ProjectedPieChart attribute), 105
selected_cell (openpyxl.worksheet.worksheet.Worksheet attribute), 213
Selection (class in openpyxl.worksheet.views), 209
selection (openpyxl.chart.chartspace.Protection attribute), 94
selection (openpyxl.worksheet.views.SheetView attribute), 210
selectLockedCells (openpyxl.worksheet.protection.SheetProtection attribute), 208
selectUnlockedCells (openpyxl.worksheet.protection.SheetProtection attribute), 208
separator (openpyxl.chart.label.DataLabel attribute), 98
separator (openpyxl.chart.label.DataLabelList attribute), 98
seq_types (openpyxl.descriptors.sequence.Sequence attribute), 127
Sequence (class in openpyxl.descriptors.sequence), 127
ser (openpyxl.chart.area_chart.AreaChart attribute), 81
ser (openpyxl.chart.area_chart.AreaChart3D attribute), 82
ser (openpyxl.chart.bar_chart.BarChart attribute), 88
ser (openpyxl.chart.bar_chart.BarChart3D attribute), 89
ser (openpyxl.chart.bubble_chart.BubbleChart attribute), 89
ser (openpyxl.chart.line_chart.LineChart attribute), 101
ser (openpyxl.chart.line_chart.LineChart3D attribute), 101
ser (openpyxl.chart.pie_chart.DoughnutChart attribute), 103
ser (openpyxl.chart.pie_chart.PieChart attribute), 104
ser (openpyxl.chart.pie_chart.PieChart3D attribute), 104
ser (openpyxl.chart.pie_chart.ProjectedPieChart attribute), 105
ser (openpyxl.chart.radar_chart.RadarChart attribute), 105
ser (openpyxl.chart.scatter_chart.ScatterChart attribute), 106
ser (openpyxl.chart.stock_chart.StockChart attribute), 110
ser (openpyxl.chart.surface_chart.SurfaceChart attribute), 111
ser (openpyxl.chart.surface_chart.SurfaceChart3D attribute), 111
serAx (openpyxl.chart.chartspace.PlotArea attribute), 94
Serialisable (class in openpyxl.descriptors.serialisable), 127
Series (class in openpyxl.chart.series), 106
SeriesAxis (class in openpyxl.chart.axis), 85
SeriesFactory() (in module openpyxl.chart.series_factory), 109
SeriesLabel (class in openpyxl.chart.series), 108
serLines (openpyxl.chart.bar_chart.BarChart attribute), 88
serLines (openpyxl.chart.bar_chart.BarChart3D attribute), 89
serLines (openpyxl.chart.pie_chart.ProjectedPieChart attribute), 105
Set (class in openpyxl.descriptors.base), 124
set() (openpyxl.worksheet.header_footer.HeaderFooterItem method), 202
set_dimension() (openpyxl.drawing.drawing.Drawing method), 132
set_explicit_value() (openpyxl.cell.cell.Cell method), 78
set_password() (openpyxl.worksheet.protection.SheetProtection method), 208
set_printer_settings() (openpyxl.worksheet.worksheet.Worksheet method), 213
setDxfStyles() (openpyxl.formatting.formatting.ConditionalFormatting method), 170
setFooter() (openpyxl.worksheet.header_footer.HeaderFooter method), 201
setHeader() (openpyxl.worksheet.header_footer.HeaderFooter method), 201
setup() (openpyxl.worksheet.page.PrintPageSetup method), 204

shade (openpyxl.drawing.colors.SystemColor attribute), 131
shadow (openpyxl.cell.text.InlineFont attribute), 80
shadow (openpyxl.styles.fonts.Font attribute), 181
Shape (class in openpyxl.drawing.shape), 152
shape (openpyxl.chart.bar_chart.BarChart3D attribute), 89
shape (openpyxl.chart.series.Series attribute), 107
Shape3D (class in openpyxl.drawing.shapes), 157
shapeId (openpyxl.comments.properties.Comment attribute), 121
ShapeStyle (class in openpyxl.drawing.shapes), 157
ShapeWriter (class in openpyxl.drawing.shape), 153
shared_strings (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
sheet (openpyxl.worksheet.protection.SheetProtection attribute), 208
sheet_properties (openpyxl.worksheet.page.PrintPageSetup attribute), 204
sheet_state (openpyxl.chartsheet.chartsheet.Chartsheet attribute), 116
SheetBackgroundPicture (class in openpyxl.chartsheet.relation), 120
SheetBackgroundPicture() (in module openpyxl.chartsheet.tests.test_relation), 115
sheetId (openpyxl.workbook.names.external.ExternalRange attribute), 188
sheetname (openpyxl.chart.reference.Reference attribute), 106
sheetnames (openpyxl.workbook.workbook.Workbook attribute), 193
sheetPr (openpyxl.chartsheet.chartsheet.Chartsheet attribute), 116
SheetProtection (class in openpyxl.worksheet.protection), 206
sheetProtection (openpyxl.chartsheet.chartsheet.Chartsheet attribute), 116
SHEETSTATE_HIDDEN (openpyxl.worksheet.worksheet.Worksheet attribute), 211
SHEETSTATE_VERYHIDDEN (openpyxl.worksheet.worksheet.Worksheet attribute), 211
SHEETSTATE_VISIBLE (openpyxl.worksheet.worksheet.Worksheet attribute), 211
SheetTitleException, 186
SheetView (class in openpyxl.worksheet.views), 209
sheetView (openpyxl.chartsheet.views.ChartsheetViewList attribute), 120
sheetViews (openpyxl.chartsheet.chartsheet.Chartsheet attribute), 116
short_color() (in module openpyxl.utils.units), 187
show_gridlines (openpyxl.worksheet.worksheet.Worksheet attribute), 213
show_summary_below (openpyxl.worksheet.worksheet.Worksheet attribute), 213
show_summary_right (openpyxl.worksheet.worksheet.Worksheet attribute), 213
showBorderUnselectedTables (openpyxl.workbook.properties.WorkbookProperties attribute), 191
showBubbleSize (openpyxl.chart.label.DataLabel attribute), 98
showBubbleSize (openpyxl.chart.label.DataLabelList attribute), 98
showButton (openpyxl.worksheet.filters.FilterColumn attribute), 199
showCatName (openpyxl.chart.label.DataLabel attribute), 98
showCatName (openpyxl.chart.label.DataLabelList attribute), 98
showDLblsOverMax (openpyxl.chart.chartspace.ChartContainer attribute), 90
showDropDown (openpyxl.worksheet.datavalidation.DataValidation attribute), 194
showErrorMessage (openpyxl.worksheet.datavalidation.DataValidation attribute), 194
showFormulas (openpyxl.worksheet.views.SheetView attribute), 210
showGridLines (openpyxl.worksheet.views.SheetView attribute), 210
showHorzBorder (openpyxl.chart.chartspace.DataTable attribute), 91
showInkAnnotation (openpyxl.workbook.properties.WorkbookProperties attribute), 192
showInputMessage (openpyxl.worksheet.datavalidation.DataValidation attribute), 194
showKeys (openpyxl.chart.chartspace.DataTable attribute), 91
showLeaderLines (openpyxl.chart.label.DataLabel attribute), 98
showLeaderLines (openpyxl.chart.label.DataLabelList attribute), 99
showLegendKey (openpyxl.chart.label.DataLabel attribute), 98
showLegendKey (openpyxl.chart.label.DataLabelList attribute), 99
showNegBubbles (openpyxl.chart.bubble_chart.BubbleChart attribute),

89
showObjects (openpyxl.workbook.properties.WorkbookProperties attribute), 192
showOutline (openpyxl.chart.chartspace.DataTable attribute), 92
showOutlineSymbols (openpyxl.worksheet.properties.Outline attribute), 205
showOutlineSymbols (openpyxl.worksheet.views.SheetView attribute), 210
showPercent (openpyxl.chart.label.DataLabel attribute), 98
showPercent (openpyxl.chart.label.DataLabelList attribute), 99
showPivotChartFilter (openpyxl.workbook.properties.WorkbookProperties attribute), 192
showRowColHeaders (openpyxl.worksheet.views.SheetView attribute), 210
showRuler (openpyxl.worksheet.views.SheetView attribute), 210
showSerName (openpyxl.chart.label.DataLabel attribute), 98
showSerName (openpyxl.chart.label.DataLabelList attribute), 99
showVal (openpyxl.chart.label.DataLabel attribute), 98
showVal (openpyxl.chart.label.DataLabelList attribute), 99
showValue (openpyxl.formatting.rule.DataBar attribute), 171
showValue (openpyxl.formatting.rule.IconSet attribute), 171
showVertBorder (openpyxl.chart.chartspace.DataTable attribute), 92
showWhiteSpace (openpyxl.worksheet.views.SheetView attribute), 210
showZeros (openpyxl.worksheet.views.SheetView attribute), 210
shrinkToFit (openpyxl.styles.alignment.Alignment attribute), 177
Side (class in openpyxl.styles.borders), 178
sideWall (openpyxl.chart.bar_chart.BarChart3D attribute), 89
sideWall (openpyxl.chart.chartspace.ChartContainer attribute), 90
size (openpyxl.chart.marker.Marker attribute), 102
sizeRepresents (openpyxl.chart.bubble_chart.BubbleChart attribute), 89
sizeWithCells (openpyxl.comments.properties.ObjectAnchor attribute), 121
smooth (openpyxl.chart.line_chart.LineChart attribute), 101
smooth (openpyxl.chart.line_chart.LineChart3D attribute), 102
smooth (openpyxl.chart.series.Series attribute), 107
smooth (openpyxl.chart.series.XYSeries attribute), 108
smtClean (openpyxl.drawing.text.CharacterProperties attribute), 163
smtId (openpyxl.drawing.text.CharacterProperties attribute), 163
snd (openpyxl.drawing.text.Hyperlink attribute), 164
softEdge (openpyxl.drawing.effect.EffectList attribute), 134
SoftEdgesEffect (class in openpyxl.drawing.effect), 139
solidFill (openpyxl.chart.shapes.GraphicalProperties attribute), 109
solidFill (openpyxl.drawing.line.LineProperties attribute), 152
solidFill (openpyxl.drawing.text.CharacterProperties attribute), 163
sort (openpyxl.worksheet.protection.SheetProtection attribute), 208
sortBy (openpyxl.worksheet.filters.SortCondition attribute), 200
SortCondition (class in openpyxl.worksheet.filters), 199
sortCondition (openpyxl.worksheet.filters.SortState attribute), 200
sortMethod (openpyxl.worksheet.filters.SortState attribute), 200
SortState (class in openpyxl.worksheet.filters), 200
sortState (openpyxl.worksheet.filters.AutoFilter attribute), 197
sourceLinked (openpyxl.chart.data_source.NumFmt attribute), 95
sourceObject (openpyxl.chartsheet.publish.WebPublishItem attribute), 118
sourceRef (openpyxl.chartsheet.publish.WebPublishItem attribute), 118
sourceType (openpyxl.chartsheet.publish.WebPublishItem attribute), 118
sp (openpyxl.drawing.line.DashStop attribute), 150
sp (openpyxl.drawing.spreadsheet_drawing.AbsoluteAnchor attribute), 159
sp (openpyxl.drawing.spreadsheet_drawing.OneCellAnchor attribute), 160
sp (openpyxl.drawing.spreadsheet_drawing.TwoCellAnchor attribute), 161
sp3d (openpyxl.chart.shapes.GraphicalProperties attribute), 110
Spacing (class in openpyxl.drawing.text), 169
spAutoFit (openpyxl.drawing.text.RichTextProperties attribute), 169
spc (openpyxl.drawing.text.CharacterProperties attribute), 163
spcAft (openpyxl.drawing.text.ParagraphProperties attribute), 167

spcBef (openpyxl.drawing.text.ParagraphProperties attribute), 167
spcCol (openpyxl.drawing.text.RichTextProperties attribute), 169
spcFirstLastPara (openpyxl.drawing.text.RichTextProperties attribute), 169
spcPct (openpyxl.drawing.text.Spacing attribute), 169
spcPts (openpyxl.drawing.text.Spacing attribute), 169
SphereCoords (class in openpyxl.drawing.shapes), 158
spinCount (openpyxl.chartsheet.protection.ChartsheetProtection attribute), 118
spinCount (openpyxl.worksheet.protection.SheetProtection attribute), 208
split_named_range() (in module openpyxl.workbook.names.named_range), 189
splitPos (openpyxl.chart.pie_chart.ProjectedPieChart attribute), 105
splitType (openpyxl.chart.pie_chart.ProjectedPieChart attribute), 105
spPr (openpyxl.chart.axis.ChartLines attribute), 82
spPr (openpyxl.chart.axis.DateAxis attribute), 83
spPr (openpyxl.chart.axis.DisplayUnitsLabel attribute), 84
spPr (openpyxl.chart.axis.NumericAxis attribute), 85
spPr (openpyxl.chart.axis.SeriesAxis attribute), 86
spPr (openpyxl.chart.axis.TextAxis attribute), 87
spPr (openpyxl.chart.chartspace.ChartSpace attribute), 91
spPr (openpyxl.chart.chartspace.DataTable attribute), 92
spPr (openpyxl.chart.chartspace.PivotFormat attribute), 92
spPr (openpyxl.chart.chartspace.PlotArea attribute), 94
spPr (openpyxl.chart.error_bar.ErrorBars attribute), 97
spPr (openpyxl.chart.label.DataLabel attribute), 98
spPr (openpyxl.chart.label.DataLabelList attribute), 99
spPr (openpyxl.chart.legend.Legend attribute), 100
spPr (openpyxl.chart.marker.DataPoint attribute), 102
spPr (openpyxl.chart.marker.Marker attribute), 102
spPr (openpyxl.chart.series.Series attribute), 107
spPr (openpyxl.chart.series.XYSeries attribute), 108
spPr (openpyxl.chart.surface_chart.BandFormat attribute), 110
spPr (openpyxl.chart.title.Title attribute), 112
spPr (openpyxl.chart.trendline.Trendline attribute), 113
spPr (openpyxl.chart.trendline.TrendlineLabel attribute), 113
spPr (openpyxl.drawing.graphic.Connector attribute), 144
spPr (openpyxl.drawing.graphic.PictureFrame attribute), 149
SpreadsheetDrawing (class in openpyxl.drawing.spreadsheet_drawing), 160
sqref (openpyxl.worksheet.datavalidation.DataValidation attribute), 194
sqref (openpyxl.worksheet.views.Selection attribute), 209
srcRect (openpyxl.drawing.fill.BlipFillProperties attribute), 141
srgbClr (openpyxl.drawing.colors.ColorChoice attribute), 128
srgbClr (openpyxl.drawing.effect.GlowEffect attribute), 135
srgbClr (openpyxl.drawing.effect.InnerShadowEffect attribute), 136
srgbClr (openpyxl.drawing.effect.OuterShadowEffect attribute), 137
srgbClr (openpyxl.drawing.effect.PresetShadowEffect attribute), 138
stA (openpyxl.drawing.effect.ReflectionEffect attribute), 139
start (openpyxl.styles.borders.Border attribute), 178
startAt (openpyxl.drawing.text.AutonumberBullet attribute), 161
state (openpyxl.chartsheet.custom.CustomChartsheetView attribute), 117
state (openpyxl.worksheet.views.Pane attribute), 209
stCxn (openpyxl.drawing.graphic.NonVisualConnectorProperties attribute), 147
stdDev (openpyxl.formatting.rule.Rule attribute), 172
StockChart (class in openpyxl.chart.stock_chart), 110
stockChart (openpyxl.chart.chartspace.PlotArea attribute), 94
stop (openpyxl.styles.fills.GradientFill attribute), 180
stopIfTrue (openpyxl.formatting.rule.Rule attribute), 172
stPos (openpyxl.drawing.effect.ReflectionEffect attribute), 139
str_format (openpyxl.workbook.names.named_range.NamedRange attribute), 189
strCache (openpyxl.chart.data_source.StrRef attribute), 96
StrData (class in openpyxl.chart.data_source), 96
stretch (openpyxl.drawing.fill.BlipFillProperties attribute), 141
StretchInfoProperties (class in openpyxl.drawing.fill), 143
Strict (class in openpyxl.descriptors), 123
strike (openpyxl.cell.text.InlineFont attribute), 80
strike (openpyxl.drawing.text.CharacterProperties attribute), 163
strike (openpyxl.styles.fonts.Font attribute), 181
String (class in openpyxl.descriptors.base), 124
strLit (openpyxl.chart.data_source.AxDataSource attribute), 95
stroke (openpyxl.drawing.shapes.Path2D attribute), 155
StrRef (class in openpyxl.chart.data_source), 96
strRef (openpyxl.chart.data_source.AxDataSource attribute), 95
strRef (openpyxl.chart.series.SeriesLabel attribute), 108
strRef (openpyxl.chart.text.Text attribute), 112
StrVal (class in openpyxl.chart.data_source), 96

Style (class in openpyxl.styles), 176
style (openpyxl.cell.interface.AbstractCell attribute), 79
style (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
style (openpyxl.chart.chartspace.ChartSpace attribute), 91
style (openpyxl.drawing.graphic.Connector attribute), 144
style (openpyxl.drawing.graphic.PictureFrame attribute), 149
style (openpyxl.styles.borders.Side attribute), 178
style (openpyxl.styles.styleable.StyleableObject attribute), 184
style_array (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
style_id (openpyxl.styles.styleable.StyleableObject attribute), 184
StyleableObject (class in openpyxl.styles.styleable), 184
StyleDescriptor (class in openpyxl.styles.styleable), 184
StyleMatrixReference (class in openpyxl.drawing.shapes), 158
StyleProxy (class in openpyxl.styles.proxy), 184
summaryBelow (openpyxl.worksheet.properties.Outline attribute), 205
summaryRight (openpyxl.worksheet.properties.Outline attribute), 205
surface3DChart (openpyxl.chart.chartspace.PlotArea attribute), 94
SurfaceChart (class in openpyxl.chart.surface_chart), 110
surfaceChart (openpyxl.chart.chartspace.PlotArea attribute), 94
SurfaceChart3D (class in openpyxl.chart.surface_chart), 111
sx (openpyxl.drawing.effect.OuterShadowEffect attribute), 137
sx (openpyxl.drawing.effect.ReflectionEffect attribute), 139
sx (openpyxl.drawing.fill.TileInfoProperties attribute), 143
sy (openpyxl.drawing.effect.OuterShadowEffect attribute), 137
sy (openpyxl.drawing.effect.ReflectionEffect attribute), 139
sy (openpyxl.drawing.fill.TileInfoProperties attribute), 143
sym (openpyxl.drawing.text.CharacterProperties attribute), 163
symbol (openpyxl.chart.marker.Marker attribute), 103
syncHorizontal (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
syncRef (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
syncVertical (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
sysClr (openpyxl.drawing.colors.ColorChoice attribute), 128
sysClr (openpyxl.drawing.effect.GlowEffect attribute), 135
sysClr (openpyxl.drawing.effect.InnerShadowEffect attribute), 136
sysClr (openpyxl.drawing.effect.OuterShadowEffect attribute), 137
sysClr (openpyxl.drawing.effect.PresetShadowEffect attribute), 138
SystemColor (class in openpyxl.drawing.colors), 130
sz (openpyxl.cell.text.InlineFont attribute), 80
sz (openpyxl.drawing.text.CharacterProperties attribute), 163
sz (openpyxl.styles.fonts.Font attribute), 181

T

t (openpyxl.cell.text.PhoneticText attribute), 81
t (openpyxl.cell.text.RichText attribute), 81
t (openpyxl.cell.text.Text attribute), 81
t (openpyxl.drawing.fill.RelativeRect attribute), 143
t (openpyxl.drawing.shapes.GeomRect attribute), 155
t (openpyxl.drawing.text.RegularTextRun attribute), 167
t (openpyxl.drawing.text.TextField attribute), 170
tab (openpyxl.drawing.text.TabStopList attribute), 169
tabColor (openpyxl.chartsheet.properties.ChartsheetProperties attribute), 117
tabColor (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
tabLst (openpyxl.drawing.text.ParagraphProperties attribute), 167
tabSelected (openpyxl.chartsheet.views.ChartsheetView attribute), 120
tabSelected (openpyxl.worksheet.views.SheetView attribute), 210
TabStop (class in openpyxl.drawing.text), 169
TabStopList (class in openpyxl.drawing.text), 169
tag (openpyxl.worksheet.page.PrintOptions attribute), 203
tag (openpyxl.worksheet.properties.Outline attribute), 205
tag (openpyxl.worksheet.properties.PageSetupProperties attribute), 206
tag (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
tagname (openpyxl.cell.text.InlineFont attribute), 80
tagname (openpyxl.cell.text.RichText attribute), 81
tagname (openpyxl.cell.text.Text attribute), 81
tagname (openpyxl.chart.area_chart.AreaChart attribute), 81
tagname (openpyxl.chart.area_chart.AreaChart3D attribute), 82
tagname (openpyxl.chart.axis.ChartLines attribute), 82
tagname (openpyxl.chart.axis.DateAxis attribute), 83

tagname (openpyxl.chart.axis.DisplayUnitsLabel attribute), 84
tagname (openpyxl.chart.axis.DisplayUnitsLabelList attribute), 84
tagname (openpyxl.chart.axis.NumericAxis attribute), 85
tagname (openpyxl.chart.axis.Scaling attribute), 85
tagname (openpyxl.chart.axis.SeriesAxis attribute), 86
tagname (openpyxl.chart.axis.TextAxis attribute), 87
tagname (openpyxl.chart.bar_chart.BarChart attribute), 88
tagname (openpyxl.chart.bar_chart.BarChart3D attribute), 89
tagname (openpyxl.chart.bubble_chart.BubbleChart attribute), 90
tagname (openpyxl.chart.chartspace.ChartContainer attribute), 90
tagname (openpyxl.chart.chartspace.ChartSpace attribute), 91
tagname (openpyxl.chart.chartspace.DataTable attribute), 92
tagname (openpyxl.chart.chartspace.ExternalData attribute), 92
tagname (openpyxl.chart.chartspace.PivotFormat attribute), 92
tagname (openpyxl.chart.chartspace.PivotFormatList attribute), 92
tagname (openpyxl.chart.chartspace.PivotSource attribute), 93
tagname (openpyxl.chart.chartspace.PlotArea attribute), 94
tagname (openpyxl.chart.chartspace.PrintSettings attribute), 94
tagname (openpyxl.chart.chartspace.Protection attribute), 94
tagname (openpyxl.chart.data_source.StrData attribute), 96
tagname (openpyxl.chart.data_source.StrRef attribute), 96
tagname (openpyxl.chart.data_source.StrVal attribute), 96
tagname (openpyxl.chart.error_bar.ErrorBars attribute), 97
tagname (openpyxl.chart.label.DataLabel attribute), 98
tagname (openpyxl.chart.label.DataLabelList attribute), 99
tagname (openpyxl.chart.layout.Layout attribute), 99
tagname (openpyxl.chart.layout.ManualLayout attribute), 99
tagname (openpyxl.chart.legend.Legend attribute), 100
tagname (openpyxl.chart.legend.LegendEntry attribute), 100
tagname (openpyxl.chart.line_chart.LineChart attribute), 101
tagname (openpyxl.chart.line_chart.LineChart3D attribute), 102
tagname (openpyxl.chart.marker.DataPoint attribute), 102
tagname (openpyxl.chart.marker.Marker attribute), 103
tagname (openpyxl.chart.picture.PictureOptions attribute), 103
tagname (openpyxl.chart.pie_chart.CustomSplit attribute), 103
tagname (openpyxl.chart.pie_chart.DoughnutChart attribute), 103
tagname (openpyxl.chart.pie_chart.PieChart attribute), 104
tagname (openpyxl.chart.pie_chart.PieChart3D attribute), 104
tagname (openpyxl.chart.pie_chart.ProjectedPieChart attribute), 105
tagname (openpyxl.chart.radar_chart.RadarChart attribute), 105
tagname (openpyxl.chart.scatter_chart.ScatterChart attribute), 106
tagname (openpyxl.chart.series.Series attribute), 107
tagname (openpyxl.chart.series.SeriesLabel attribute), 108
tagname (openpyxl.chart.shapes.GraphicalProperties attribute), 110
tagname (openpyxl.chart.stock_chart.StockChart attribute), 110
tagname (openpyxl.chart.surface_chart.BandFormat attribute), 110
tagname (openpyxl.chart.surface_chart.BandFormatList attribute), 110
tagname (openpyxl.chart.surface_chart.SurfaceChart attribute), 111
tagname (openpyxl.chart.surface_chart.SurfaceChart3D attribute), 111
tagname (openpyxl.chart.text.RichText attribute), 111
tagname (openpyxl.chart.title.Title attribute), 112
tagname (openpyxl.chart.trendline.Trendline attribute), 113
tagname (openpyxl.chart.trendline.TrendlineLabel attribute), 113
tagname (openpyxl.chart.updown_bars.UpDownBars attribute), 113
tagname (openpyxl.chartsheet.chartsheet.Chartsheet attribute), 116
tagname (openpyxl.chartsheet.custom.CustomChartsheetView attribute), 117
tagname (openpyxl.chartsheet.custom.CustomChartsheetViews attribute), 117
tagname (openpyxl.chartsheet.properties.ChartsheetProperties attribute), 117
tagname (openpyxl.chartsheet.protection.ChartsheetProtection attribute), 118
tagname (openpyxl.chartsheet.publish.WebPublishItem attribute), 118
tagname (openpyxl.chartsheet.publish.WebPublishItems

attribute), 118
tagname (openpyxl.chartsheet.relation.SheetBackgroundPicture attribute), 120
tagname (openpyxl.chartsheet.views.ChartsheetView attribute), 120
tagname (openpyxl.chartsheet.views.ChartsheetViewList attribute), 120
tagname (openpyxl.comments.author.AuthorList attribute), 120
tagname (openpyxl.comments.properties.Comment attribute), 121
tagname (openpyxl.comments.properties.CommentSheet attribute), 121
tagname (openpyxl.descriptors.serialisable.Serialisable attribute), 127
tagname (openpyxl.drawing.colors.ColorChoice attribute), 128
tagname (openpyxl.drawing.colors.ColorMapping attribute), 129
tagname (openpyxl.drawing.colors.HSLColor attribute), 130
tagname (openpyxl.drawing.colors.RGBPercent attribute), 130
tagname (openpyxl.drawing.colors.SystemColor attribute), 132
tagname (openpyxl.drawing.fill.Blip attribute), 141
tagname (openpyxl.drawing.fill.BlipFillProperties attribute), 141
tagname (openpyxl.drawing.fill.GradientFillProperties attribute), 142
tagname (openpyxl.drawing.fill.GradientStop attribute), 142
tagname (openpyxl.drawing.fill.GradientStopList attribute), 142
tagname (openpyxl.drawing.fill.PatternFillProperties attribute), 143
tagname (openpyxl.drawing.fill.RelativeRect attribute), 143
tagname (openpyxl.drawing.fill.StretchInfoProperties attribute), 143
tagname (openpyxl.drawing.graphic.ChartRelation attribute), 144
tagname (openpyxl.drawing.graphic.GraphicData attribute), 144
tagname (openpyxl.drawing.graphic.GraphicFrame attribute), 145
tagname (openpyxl.drawing.graphic.GraphicObject attribute), 145
tagname (openpyxl.drawing.graphic.NonVisualDrawingProps attribute), 147
tagname (openpyxl.drawing.graphic.NonVisualGraphicFrame attribute), 148
tagname (openpyxl.drawing.graphic.NonVisualGraphicFrameProperties attribute), 148
tagname (openpyxl.drawing.graphic.NonVisualPictureProperties attribute), 148
tagname (openpyxl.drawing.graphic.PictureFrame attribute), 149
tagname (openpyxl.drawing.graphic.PictureLocking attribute), 150
tagname (openpyxl.drawing.graphic.PictureNonVisual attribute), 150
tagname (openpyxl.drawing.line.DashStop attribute), 150
tagname (openpyxl.drawing.line.LineEndProperties attribute), 150
tagname (openpyxl.drawing.line.LineJoinMiterProperties attribute), 151
tagname (openpyxl.drawing.line.LineProperties attribute), 152
tagname (openpyxl.drawing.shapes.Transform2D attribute), 158
tagname (openpyxl.drawing.spreadsheet_drawing.AbsoluteAnchor attribute), 159
tagname (openpyxl.drawing.spreadsheet_drawing.AnchorMarker attribute), 159
tagname (openpyxl.drawing.spreadsheet_drawing.OneCellAnchor attribute), 160
tagname (openpyxl.drawing.spreadsheet_drawing.SpreadsheetDrawing attribute), 160
tagname (openpyxl.drawing.spreadsheet_drawing.TwoCellAnchor attribute), 161
tagname (openpyxl.drawing.text.CharacterProperties attribute), 163
tagname (openpyxl.drawing.text.Font attribute), 164
tagname (openpyxl.drawing.text.ListStyle attribute), 165
tagname (openpyxl.drawing.text.Paragraph attribute), 165
tagname (openpyxl.drawing.text.ParagraphProperties attribute), 167
tagname (openpyxl.drawing.text.RegularTextRun attribute), 167
tagname (openpyxl.drawing.text.RichTextProperties attribute), 169
tagname (openpyxl.formatting.rule.ColorScale attribute), 170
tagname (openpyxl.formatting.rule.DataBar attribute), 171
tagname (openpyxl.formatting.rule.FormatObject attribute), 171
tagname (openpyxl.formatting.rule.IconSet attribute), 171
tagname (openpyxl.formatting.rule.Rule attribute), 172
tagname (openpyxl.packaging.manifest.FileExtension attribute), 173
tagname (openpyxl.packaging.manifest.Manifest attribute), 173
tagname (openpyxl.packaging.manifest.Override attribute), 174
tagname (openpyxl.packaging.relationship.Relationship

attribute), 174
tagname (openpyxl.packaging.relationship.RelationshipList attribute), 174
tagname (openpyxl.styles.alignment.Alignment attribute), 177
tagname (openpyxl.styles.borders.Border attribute), 178
tagname (openpyxl.styles.colors.Color attribute), 178
tagname (openpyxl.styles.differential.DifferentialStyle attribute), 180
tagname (openpyxl.styles.fills.Fill attribute), 180
tagname (openpyxl.styles.fills.GradientFill attribute), 180
tagname (openpyxl.styles.fills.PatternFill attribute), 181
tagname (openpyxl.styles.fonts.Font attribute), 182
tagname (openpyxl.styles.named_styles.NamedCellStyle attribute), 182
tagname (openpyxl.styles.named_styles.NamedCellStyleList attribute), 183
tagname (openpyxl.styles.protection.Protection attribute), 184
tagname (openpyxl.workbook.properties.CalcProperties attribute), 190
tagname (openpyxl.workbook.properties.FileVersion attribute), 190
tagname (openpyxl.workbook.properties.WorkbookProperties attribute), 192
tagname (openpyxl.worksheet.datavalidation.DataValidation attribute), 194
tagname (openpyxl.worksheet.datavalidation.DataValidationList attribute), 194
tagname (openpyxl.worksheet.drawing.Drawing attribute), 196
tagname (openpyxl.worksheet.filters.AutoFilter attribute), 197
tagname (openpyxl.worksheet.filters.FilterColumn attribute), 199
tagname (openpyxl.worksheet.filters.SortCondition attribute), 200
tagname (openpyxl.worksheet.filters.SortState attribute), 200
tagname (openpyxl.worksheet.hyperlink.Hyperlink attribute), 202
tagname (openpyxl.worksheet.page.PageMargins attribute), 203
tagname (openpyxl.worksheet.page.PrintOptions attribute), 203
tagname (openpyxl.worksheet.page.PrintPageSetup attribute), 204
tagname (openpyxl.worksheet.pagebreak.Break attribute), 205
tagname (openpyxl.worksheet.pagebreak.PageBreak attribute), 205
tagname (openpyxl.worksheet.properties.Outline attribute), 205
tagname (openpyxl.worksheet.properties.PageSetupProperties attribute), 206
tagname (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
tagname (openpyxl.worksheet.protection.SheetProtection attribute), 208
tagname (openpyxl.worksheet.views.SheetView attribute), 210
tailEnd (openpyxl.drawing.line.LineProperties attribute), 152
Target (openpyxl.packaging.relationship.Relationship attribute), 174
Target (openpyxl.workbook.names.external.ExternalBook attribute), 188
target (openpyxl.worksheet.hyperlink.Hyperlink attribute), 202
TargetMode (openpyxl.packaging.relationship.Relationship attribute), 174
TargetMode (openpyxl.workbook.names.external.ExternalBook attribute), 188
test_ctor() (openpyxl.chartsheet.tests.test_chartsheet.TestChartsheet method), 114
test_read() (openpyxl.chartsheet.tests.test_chartsheet.TestChartsheet method), 114
test_read() (openpyxl.chartsheet.tests.test_custom.TestCustomChartsheetVi
method), 114
test_read() (openpyxl.chartsheet.tests.test_custom.TestCustomChartsheetVi
method), 114
test_read() (openpyxl.chartsheet.tests.test_properties.TestChartsheetPr method), 114
test_read() (openpyxl.chartsheet.tests.test_protection.TestChartsheetProtecti
method), 115
test_read() (openpyxl.chartsheet.tests.test_publish.TestWebPublishItems method), 115
test_read() (openpyxl.chartsheet.tests.test_publish.TestWebPulishItem method), 115
test_read() (openpyxl.chartsheet.tests.test_relation.TestDrawingHF method), 115
test_read() (openpyxl.chartsheet.tests.test_relation.TestSheetBackgroundPic
method), 115
test_read() (openpyxl.chartsheet.tests.test_views.TestChartsheetView method), 115
test_read() (openpyxl.chartsheet.tests.test_views.TestChartsheetViewList method), 115
test_write() (openpyxl.chartsheet.tests.test_chartsheet.TestChartsheet method), 114
test_write() (openpyxl.chartsheet.tests.test_custom.TestCustomChartsheetV
method), 114
test_write() (openpyxl.chartsheet.tests.test_custom.TestCustomChartsheetV
method), 114
test_write() (openpyxl.chartsheet.tests.test_properties.TestChartsheetPr method), 114
test_write() (openpyxl.chartsheet.tests.test_protection.TestChartsheetProtect
method), 115
test_write() (openpyxl.chartsheet.tests.test_publish.TestWebPublishItems

method), 115
test_write() (openpyxl.chartsheet.tests.test_publish.TestWebPulishItem method), 115
test_write() (openpyxl.chartsheet.tests.test_relation.TestDrawingHF method), 115
test_write() (openpyxl.chartsheet.tests.test_relation.TestSheetBackgroundPicture method), 115
test_write() (openpyxl.chartsheet.tests.test_views.TestChartsheetView method), 115
test_write() (openpyxl.chartsheet.tests.test_views.TestChartsheetViewList method), 115
test_write_charts() (openpyxl.chartsheet.tests.test_chartsheet.TestChartsheet method), 114
TestChartsheet (class in openpyxl.chartsheet.tests.test_chartsheet), 114
TestChartsheetPr (class in openpyxl.chartsheet.tests.test_properties), 114
TestChartsheetProtection (class in openpyxl.chartsheet.tests.test_protection), 115
TestChartsheetView (class in openpyxl.chartsheet.tests.test_views), 115
TestChartsheetViewList (class in openpyxl.chartsheet.tests.test_views), 115
TestCustomChartsheetView (class in openpyxl.chartsheet.tests.test_custom), 114
TestCustomChartsheetViews (class in openpyxl.chartsheet.tests.test_custom), 114
TestDrawingHF (class in openpyxl.chartsheet.tests.test_relation), 115
TestSheetBackgroundPicture (class in openpyxl.chartsheet.tests.test_relation), 115
TestWebPublishItems (class in openpyxl.chartsheet.tests.test_publish), 115
TestWebPulishItem (class in openpyxl.chartsheet.tests.test_publish), 115
Text (class in openpyxl.cell.text), 81
Text (class in openpyxl.chart.text), 112
text (openpyxl.comments.comments.Comment attribute), 121
text (openpyxl.comments.properties.Comment attribute), 121
text (openpyxl.formatting.rule.Rule attribute), 172
text (openpyxl.worksheet.header_footer.HeaderFooterItem attribute), 202
text_color (openpyxl.drawing.shape.Shape attribute), 153
TextAxis (class in openpyxl.chart.axis), 86
TextField (class in openpyxl.drawing.text), 169
textHAlign (openpyxl.comments.properties.Properties attribute), 122
TextNormalAutofit (class in openpyxl.drawing.text), 170
TextPoint (class in openpyxl.descriptors.excel), 125
textRotation (openpyxl.styles.alignment.Alignment attribute), 177
textVAlign (openpyxl.comments.properties.Properties attribute), 122
tgtFrame (openpyxl.drawing.text.Hyperlink attribute), 164
theme (openpyxl.styles.colors.Color attribute), 178
thickBot (openpyxl.worksheet.dimensions.RowDimension attribute), 196
thickTop (openpyxl.worksheet.dimensions.RowDimension attribute), 196
thresh (openpyxl.drawing.effect.AlphaBiLevelEffect attribute), 132
thresh (openpyxl.drawing.effect.BiLevelEffect attribute), 133
tickLblPos (openpyxl.chart.axis.DateAxis attribute), 83
tickLblPos (openpyxl.chart.axis.NumericAxis attribute), 85
tickLblPos (openpyxl.chart.axis.SeriesAxis attribute), 86
tickLblPos (openpyxl.chart.axis.TextAxis attribute), 87
tickLblSkip (openpyxl.chart.axis.SeriesAxis attribute), 86
tickLblSkip (openpyxl.chart.axis.TextAxis attribute), 87
tickMarkSkip (openpyxl.chart.axis.SeriesAxis attribute), 86
tickMarkSkip (openpyxl.chart.axis.TextAxis attribute), 87
tile (openpyxl.drawing.fill.BlipFillProperties attribute), 141
TileInfoProperties (class in openpyxl.drawing.fill), 143
tileRect (openpyxl.drawing.fill.GradientFillProperties attribute), 142
time_to_days() (in module openpyxl.utils.datetime), 186
timedelta_to_days() (in module openpyxl.utils.datetime), 186
timePeriod (openpyxl.formatting.rule.Rule attribute), 172
tIns (openpyxl.drawing.text.RichTextProperties attribute), 169
tint (openpyxl.drawing.colors.SystemColor attribute), 132
tint (openpyxl.drawing.fill.Blip attribute), 141
tint (openpyxl.styles.colors.Color attribute), 178
TintEffect (class in openpyxl.drawing.effect), 139
Title (class in openpyxl.chart.title), 112
title (openpyxl.chart.axis.DateAxis attribute), 83
title (openpyxl.chart.axis.NumericAxis attribute), 85
title (openpyxl.chart.axis.SeriesAxis attribute), 86
title (openpyxl.chart.axis.TextAxis attribute), 87
title (openpyxl.chart.chartspace.ChartContainer attribute), 90
title (openpyxl.chartsheet.publish.WebPublishItem attribute), 118
title (openpyxl.drawing.graphic.NonVisualDrawingProps attribute), 147
title_maker() (in module openpyxl.chart.title), 112
TitleDescriptor (class in openpyxl.chart.title), 112
to (openpyxl.drawing.spreadsheet_drawing.TwoCellAnchor

attribute), 161
to_excel() (in module openpyxl.utils.datetime), 186
to_tree() (openpyxl.chart.chartspace.PlotArea method), 94
to_tree() (openpyxl.chart.series.Series method), 107
to_tree() (openpyxl.chartsheet.chartsheet.Chartsheet method), 116
to_tree() (openpyxl.comments.properties.CommentSheet method), 121
to_tree() (openpyxl.descriptors.nested.EmptyTag method), 126
to_tree() (openpyxl.descriptors.nested.Nested method), 126
to_tree() (openpyxl.descriptors.nested.NestedText method), 126
to_tree() (openpyxl.descriptors.sequence.NestedSequence method), 127
to_tree() (openpyxl.descriptors.sequence.Sequence method), 127
to_tree() (openpyxl.descriptors.sequence.ValueSequence method), 127
to_tree() (openpyxl.descriptors.serialisable.Serialisable method), 127
to_tree() (openpyxl.packaging.manifest.Manifest method), 173
to_tree() (openpyxl.packaging.relationship.RelationshipList method), 174
to_tree() (openpyxl.styles.fills.GradientFill method), 180
to_tree() (openpyxl.styles.fills.PatternFill method), 181
to_tree() (openpyxl.worksheet.page.PrintPageSetup method), 204
to_tree() (openpyxl.worksheet.related.Related method), 209
tooltip (openpyxl.drawing.text.Hyperlink attribute), 164
tooltip (openpyxl.worksheet.hyperlink.Hyperlink attribute), 202
top (openpyxl.styles.borders.Border attribute), 178
top (openpyxl.styles.fills.GradientFill attribute), 180
top (openpyxl.worksheet.filters.Top10 attribute), 200
top (openpyxl.worksheet.page.PageMargins attribute), 203
Top10 (class in openpyxl.worksheet.filters), 200
top10 (openpyxl.worksheet.filters.FilterColumn attribute), 199
topLeftCell (openpyxl.worksheet.views.Pane attribute), 209
topLeftCell (openpyxl.worksheet.views.SheetView attribute), 210
Transform (class in openpyxl.drawing.colors), 132
Transform2D (class in openpyxl.drawing.shapes), 158
transitionEntry (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
transitionEvaluation (openpyxl.worksheet.properties.WorksheetProperties attribute), 206
Trendline (class in openpyxl.chart.trendline), 112
trendline (openpyxl.chart.series.Series attribute), 107
trendline (openpyxl.chart.series.XYSeries attribute), 109
TrendlineLabel (class in openpyxl.chart.trendline), 113
trendlineLbl (openpyxl.chart.trendline.Trendline attribute), 113
trendlineType (openpyxl.chart.trendline.Trendline attribute), 113
Tuple (class in openpyxl.descriptors.base), 124
TwoCellAnchor (class in openpyxl.drawing.spreadsheet_drawing), 160
twoCellAnchor (openpyxl.drawing.spreadsheet_drawing.SpreadsheetDrawi attribute), 160
tx (openpyxl.chart.axis.DisplayUnitsLabel attribute), 84
tx (openpyxl.chart.series.Series attribute), 107
tx (openpyxl.chart.series.XYSeries attribute), 109
tx (openpyxl.chart.title.Title attribute), 112
tx (openpyxl.chart.trendline.TrendlineLabel attribute), 113
tx (openpyxl.drawing.fill.TileInfoProperties attribute), 143
tx1 (openpyxl.drawing.colors.ColorMapping attribute), 129
tx2 (openpyxl.drawing.colors.ColorMapping attribute), 129
txPr (openpyxl.chart.axis.DateAxis attribute), 83
txPr (openpyxl.chart.axis.DisplayUnitsLabel attribute), 84
txPr (openpyxl.chart.axis.NumericAxis attribute), 85
txPr (openpyxl.chart.axis.SeriesAxis attribute), 86
txPr (openpyxl.chart.axis.TextAxis attribute), 88
txPr (openpyxl.chart.chartspace.ChartSpace attribute), 91
txPr (openpyxl.chart.chartspace.DataTable attribute), 92
txPr (openpyxl.chart.chartspace.PivotFormat attribute), 92
txPr (openpyxl.chart.label.DataLabel attribute), 98
txPr (openpyxl.chart.label.DataLabelList attribute), 99
txPr (openpyxl.chart.legend.Legend attribute), 100
txPr (openpyxl.chart.legend.LegendEntry attribute), 100
txPr (openpyxl.chart.title.Title attribute), 112
txPr (openpyxl.chart.trendline.TrendlineLabel attribute), 113
ty (openpyxl.drawing.fill.TileInfoProperties attribute), 143
type (openpyxl.cell.text.PhoneticProperties attribute), 80
type (openpyxl.drawing.effect.EffectContainer attribute), 133
type (openpyxl.drawing.line.LineEndProperties attribute), 151
type (openpyxl.drawing.text.AutonumberBullet attribute), 161
type (openpyxl.drawing.text.TextField attribute), 170
type (openpyxl.formatting.rule.FormatObject attribute),

171
type (openpyxl.formatting.rule.Rule attribute), 173
Type (openpyxl.packaging.relationship.Relationship attribute), 174
type (openpyxl.styles.colors.Color attribute), 178
type (openpyxl.styles.fills.GradientFill attribute), 180
Type (openpyxl.workbook.names.external.ExternalBook attribute), 188
type (openpyxl.worksheet.datavalidation.DataValidation attribute), 194
type (openpyxl.worksheet.filters.DynamicFilter attribute), 198
type (openpyxl.worksheet.header_footer.HeaderFooterItem attribute), 202
TYPE_BOOL (openpyxl.cell.cell.Cell attribute), 77
TYPE_ERROR (openpyxl.cell.cell.Cell attribute), 77
TYPE_FORMULA (openpyxl.cell.cell.Cell attribute), 77
TYPE_FORMULA_CACHE_STRING (openpyxl.cell.cell.Cell attribute), 77
TYPE_INLINE (openpyxl.cell.cell.Cell attribute), 77
TYPE_NULL (openpyxl.cell.cell.Cell attribute), 77
TYPE_NUMERIC (openpyxl.cell.cell.Cell attribute), 77
TYPE_STRING (openpyxl.cell.cell.Cell attribute), 77
Typed (class in openpyxl.descriptors.base), 125
typeface (openpyxl.drawing.text.Font attribute), 164
tzname() (openpyxl.utils.datetime.GMT method), 185

U

u (openpyxl.cell.text.InlineFont attribute), 80
u (openpyxl.drawing.text.CharacterProperties attribute), 163
u (openpyxl.styles.fonts.Font attribute), 182
uFill (openpyxl.drawing.text.CharacterProperties attribute), 163
uFillTx (openpyxl.drawing.text.CharacterProperties attribute), 163
uiObject (openpyxl.comments.properties.Properties attribute), 122
uLn (openpyxl.drawing.text.CharacterProperties attribute), 163
uLnTx (openpyxl.drawing.text.CharacterProperties attribute), 163
UNDERLINE_DOUBLE (openpyxl.styles.fonts.Font attribute), 181
UNDERLINE_DOUBLE_ACCOUNTING (openpyxl.styles.fonts.Font attribute), 181
UNDERLINE_SINGLE (openpyxl.styles.fonts.Font attribute), 181
UNDERLINE_SINGLE_ACCOUNTING (openpyxl.styles.fonts.Font attribute), 181
unique (openpyxl.descriptors.sequence.Sequence attribute), 127
UniversalMeasure (class in openpyxl.descriptors.excel), 125
unmerge_cells() (openpyxl.worksheet.worksheet.Worksheet method), 213
unpack_rules() (in module openpyxl.formatting.formatting), 170
up (openpyxl.drawing.shapes.Backdrop attribute), 153
upBars (openpyxl.chart.updown_bars.UpDownBars attribute), 114
update() (openpyxl.formatting.formatting.ConditionalFormatting method), 170
updateLinks (openpyxl.workbook.properties.WorkbookProperties attribute), 192
UpDownBars (class in openpyxl.chart.updown_bars), 113
upDownBars (openpyxl.chart.line_chart.LineChart attribute), 101
upDownBars (openpyxl.chart.line_chart.LineChart3D attribute), 102
upDownBars (openpyxl.chart.stock_chart.StockChart attribute), 110
upright (openpyxl.drawing.text.RichTextProperties attribute), 169
uri (openpyxl.descriptors.excel.Extension attribute), 125
uri (openpyxl.drawing.graphic.GraphicData attribute), 144
useA (openpyxl.drawing.effect.ColorChangeEffect attribute), 133
useFirstPageNumber (openpyxl.worksheet.page.PrintPageSetup attribute), 204
usePrinterDefaults (openpyxl.worksheet.page.PrintPageSetup attribute), 204
userInterface (openpyxl.chart.chartspace.Protection attribute), 95
userShapes (openpyxl.chart.chartspace.ChartSpace attribute), 91
utcoffset() (openpyxl.utils.datetime.GMT method), 185

V

v (openpyxl.chart.data_source.NumVal attribute), 96
v (openpyxl.chart.data_source.StrVal attribute), 96
v (openpyxl.chart.series.SeriesLabel attribute), 108
val (openpyxl.chart.error_bar.ErrorBars attribute), 97
val (openpyxl.chart.series.Series attribute), 108
val (openpyxl.drawing.colors.SystemColor attribute), 132
val (openpyxl.formatting.rule.FormatObject attribute), 171
val (openpyxl.worksheet.filters.CustomFilter attribute), 197
val (openpyxl.worksheet.filters.DynamicFilter attribute), 198
val (openpyxl.worksheet.filters.Top10 attribute), 200
valAx (openpyxl.chart.chartspace.PlotArea attribute), 94
VALID_TYPES (openpyxl.cell.cell.Cell attribute), 77

valIso (openpyxl.worksheet.filters.DynamicFilter attribute), 198
value (openpyxl.cell.cell.Cell attribute), 78
value (openpyxl.cell.interface.AbstractCell attribute), 79
value (openpyxl.cell.read_only.ReadOnlyCell attribute), 79
value (openpyxl.styles.colors.Color attribute), 178
value (openpyxl.workbook.names.named_range.NamedRange attribute), 189
value (openpyxl.workbook.names.named_range.NamedValue attribute), 189
VALUE_TAG (openpyxl.reader.worksheet.WorkSheetParser attribute), 175
ValueDescriptor (class in openpyxl.formatting.rule), 173
ValueSequence (class in openpyxl.descriptors.sequence), 127
varyColors (openpyxl.chart.area_chart.AreaChart attribute), 81
varyColors (openpyxl.chart.area_chart.AreaChart3D attribute), 82
varyColors (openpyxl.chart.bar_chart.BarChart attribute), 88
varyColors (openpyxl.chart.bar_chart.BarChart3D attribute), 89
varyColors (openpyxl.chart.bubble_chart.BubbleChart attribute), 90
varyColors (openpyxl.chart.line_chart.LineChart attribute), 101
varyColors (openpyxl.chart.line_chart.LineChart3D attribute), 102
varyColors (openpyxl.chart.pie_chart.DoughnutChart attribute), 103
varyColors (openpyxl.chart.pie_chart.PieChart attribute), 104
varyColors (openpyxl.chart.pie_chart.PieChart3D attribute), 104
varyColors (openpyxl.chart.pie_chart.ProjectedPieChart attribute), 105
varyColors (openpyxl.chart.radar_chart.RadarChart attribute), 105
varyColors (openpyxl.chart.scatter_chart.ScatterChart attribute), 106
vba_code (openpyxl.worksheet.worksheet.Worksheet attribute), 213
Vector3D (class in openpyxl.drawing.shapes), 158
vert (openpyxl.drawing.text.RichTextProperties attribute), 169
vertAlign (openpyxl.cell.text.InlineFont attribute), 80
vertAlign (openpyxl.styles.fonts.Font attribute), 182
vertical (openpyxl.styles.alignment.Alignment attribute), 177
vertical (openpyxl.styles.borders.Border attribute), 178
verticalCentered (openpyxl.worksheet.page.PrintOptions attribute), 203
verticalCentered() (openpyxl.worksheet.page.PrintPageSetup method), 204
verticalDpi (openpyxl.worksheet.page.PrintPageSetup attribute), 204
vertOverflow (openpyxl.drawing.text.RichTextProperties attribute), 169
view (openpyxl.worksheet.views.SheetView attribute), 210
view3D (openpyxl.chart.bar_chart.BarChart3D attribute), 89
view3D (openpyxl.chart.chartspace.ChartContainer attribute), 90
visible (openpyxl.worksheet.dimensions.Dimension attribute), 195

W

w (openpyxl.chart.layout.ManualLayout attribute), 99
w (openpyxl.drawing.line.LineEndProperties attribute), 151
w (openpyxl.drawing.line.LineProperties attribute), 152
w (openpyxl.drawing.shapes.Bevel attribute), 153
w (openpyxl.drawing.shapes.Path2D attribute), 155
W3CDTF_to_datetime() (in module openpyxl.utils.datetime), 185
WebPublishItem (class in openpyxl.chartsheet.publish), 118
webPublishItem (openpyxl.chartsheet.publish.WebPublishItems attribute), 118
WebPublishItem() (in module openpyxl.chartsheet.tests.test_publish), 115
WebPublishItems (class in openpyxl.chartsheet.publish), 118
webPublishItems (openpyxl.chartsheet.chartsheet.Chartsheet attribute), 116
WebPublishItems() (in module openpyxl.chartsheet.tests.test_publish), 115
width (openpyxl.drawing.drawing.Drawing attribute), 132
width (openpyxl.worksheet.dimensions.ColumnDimension attribute), 195
windowProtection (openpyxl.worksheet.views.SheetView attribute), 210
wireframe (openpyxl.chart.surface_chart.SurfaceChart attribute), 111
wireframe (openpyxl.chart.surface_chart.SurfaceChart3D attribute), 111
wMode (openpyxl.chart.layout.ManualLayout attribute), 99
Workbook (class in openpyxl.workbook.workbook), 192
WorkbookAlreadySaved, 186

WorkbookProperties (class in openpyxl.workbook.properties), 190
workbookViewId (openpyxl.chartsheet.views.ChartsheetView attribute), 120
workbookViewId (openpyxl.worksheet.views.SheetView attribute), 210
Worksheet (class in openpyxl.worksheet.worksheet), 211
WorkSheetParser (class in openpyxl.reader.worksheet), 175
WorksheetProperties (class in openpyxl.worksheet.properties), 206
worksheets (openpyxl.workbook.workbook.Workbook attribute), 193
wrap (openpyxl.drawing.text.RichTextProperties attribute), 169
wrapText (openpyxl.styles.alignment.Alignment attribute), 177
write() (openpyxl.drawing.shape.ShapeWriter method), 153
write_cell() (in module openpyxl.writer.etree_worksheet), 214
write_cell() (in module openpyxl.writer.lxml_worksheet), 214
write_cols() (in module openpyxl.writer.worksheet), 215
write_comments() (openpyxl.comments.writer.CommentWriter method), 123
write_comments_vml() (openpyxl.comments.writer.CommentWriter method), 123
write_conditional_formatting() (in module openpyxl.writer.worksheet), 215
write_content_types() (in module openpyxl.packaging.manifest), 174
write_data() (openpyxl.writer.excel.ExcelWriter method), 214
write_drawing() (in module openpyxl.writer.worksheet), 215
write_external_book_rel() (in module openpyxl.workbook.names.external), 188
write_external_link() (in module openpyxl.workbook.names.external), 188
write_format() (in module openpyxl.writer.worksheet), 215
write_header_footer() (in module openpyxl.writer.worksheet), 215
write_hyperlinks() (in module openpyxl.writer.worksheet), 215
write_mergecells() (in module openpyxl.writer.worksheet), 215
write_only (openpyxl.workbook.workbook.Workbook attribute), 193
write_properties_app() (in module openpyxl.writer.workbook), 215
write_rels() (in module openpyxl.writer.relations), 214
write_root_rels() (in module openpyxl.writer.workbook), 215
write_rows() (in module openpyxl.writer.etree_worksheet), 214
write_rows() (in module openpyxl.writer.lxml_worksheet), 214
write_string_table() (in module openpyxl.writer.strings), 214
write_theme() (in module openpyxl.writer.theme), 215
write_workbook() (in module openpyxl.writer.workbook), 215
write_workbook_rels() (in module openpyxl.writer.workbook), 215
write_worksheet() (in module openpyxl.writer.worksheet), 215
WriteOnlyCell() (in module openpyxl.writer.write_only), 215
WriteOnlyWorksheet (class in openpyxl.writer.write_only), 215
writer (openpyxl.writer.write_only.WriteOnlyWorksheet attribute), 216

X

x (openpyxl.chart.layout.ManualLayout attribute), 99
x (openpyxl.drawing.shapes.AdjPoint2D attribute), 153
x (openpyxl.drawing.shapes.Point2D attribute), 155
x (openpyxl.drawing.shapes.Point3D attribute), 156
x_axis (openpyxl.chart.area_chart.AreaChart attribute), 82
x_axis (openpyxl.chart.area_chart.AreaChart3D attribute), 82
x_axis (openpyxl.chart.bar_chart.BarChart attribute), 88
x_axis (openpyxl.chart.bar_chart.BarChart3D attribute), 89
x_axis (openpyxl.chart.bubble_chart.BubbleChart attribute), 90
x_axis (openpyxl.chart.line_chart.LineChart attribute), 101
x_axis (openpyxl.chart.line_chart.LineChart3D attribute), 102
x_axis (openpyxl.chart.radar_chart.RadarChart attribute), 105
x_axis (openpyxl.chart.scatter_chart.ScatterChart attribute), 106
x_axis (openpyxl.chart.stock_chart.StockChart attribute), 110
x_axis (openpyxl.chart.surface_chart.SurfaceChart3D attribute), 111
xfId (openpyxl.styles.named_styles.NamedCellStyle attribute), 182
xfrm (openpyxl.chart.shapes.GraphicalProperties attribute), 110

xfrm (openpyxl.drawing.graphic.GraphicFrame attribute), 145
xfrm (openpyxl.drawing.graphic.GroupShapeProperties attribute), 146
xml_source (openpyxl.worksheet.read_only.ReadOnlyWorksheet attribute), 208
xMode (openpyxl.chart.layout.ManualLayout attribute), 100
xSplit (openpyxl.worksheet.views.Pane attribute), 209
xVal (openpyxl.chart.series.Series attribute), 108
xVal (openpyxl.chart.series.XYSeries attribute), 109
xWindow (openpyxl.worksheet.datavalidation.DataValidationList attribute), 194
XYSeries (class in openpyxl.chart.series), 108

Y

y (openpyxl.chart.layout.ManualLayout attribute), 100
y (openpyxl.drawing.shapes.AdjPoint2D attribute), 153
y (openpyxl.drawing.shapes.Point2D attribute), 155
y (openpyxl.drawing.shapes.Point3D attribute), 156
y_axis (openpyxl.chart.area_chart.AreaChart attribute), 82
y_axis (openpyxl.chart.area_chart.AreaChart3D attribute), 82
y_axis (openpyxl.chart.bar_chart.BarChart attribute), 88
y_axis (openpyxl.chart.bar_chart.BarChart3D attribute), 89
y_axis (openpyxl.chart.bubble_chart.BubbleChart attribute), 90
y_axis (openpyxl.chart.line_chart.LineChart attribute), 101
y_axis (openpyxl.chart.line_chart.LineChart3D attribute), 102
y_axis (openpyxl.chart.radar_chart.RadarChart attribute), 105
y_axis (openpyxl.chart.scatter_chart.ScatterChart attribute), 106
y_axis (openpyxl.chart.stock_chart.StockChart attribute), 110
y_axis (openpyxl.chart.surface_chart.SurfaceChart3D attribute), 111
year (openpyxl.worksheet.filters.DateGroupItem attribute), 198
yMode (openpyxl.chart.layout.ManualLayout attribute), 100
ySplit (openpyxl.worksheet.views.Pane attribute), 209
yVal (openpyxl.chart.series.Series attribute), 108
yVal (openpyxl.chart.series.XYSeries attribute), 109
yWindow (openpyxl.worksheet.datavalidation.DataValidationList attribute), 194

Z

z (openpyxl.drawing.shapes.Point3D attribute), 156
z (openpyxl.drawing.shapes.Shape3D attribute), 157
z_axis (openpyxl.chart.area_chart.AreaChart3D attribute), 82
z_axis (openpyxl.chart.bar_chart.BarChart3D attribute), 89
z_axis (openpyxl.chart.line_chart.LineChart3D attribute), 102
z_axis (openpyxl.chart.surface_chart.SurfaceChart3D attribute), 111
zoom (openpyxl.drawing.shapes.Camera attribute), 154
zoomScale (openpyxl.chartsheet.views.ChartsheetView attribute), 120
zoomScale (openpyxl.worksheet.views.SheetView attribute), 210
zoomScaleNormal (openpyxl.worksheet.views.SheetView attribute), 210
zoomScalePageLayoutView (openpyxl.worksheet.views.SheetView attribute), 210
zoomScaleSheetLayoutView (openpyxl.worksheet.views.SheetView attribute), 210
zoomToFit (openpyxl.chartsheet.custom.CustomChartsheetView attribute), 117
zoomToFit (openpyxl.chartsheet.views.ChartsheetView attribute), 120