

# 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 formatting for numbers.

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='FFFFFFF',
...                     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)
>>>

```

## Cell Styles and Named Styles

There are two types of styles: cell styles and named styles, also known as style templates.

### Cell Styles

Cell 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 only one changes.

```

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

```

## Copying styles

Styles can also be copied

```

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

```

## Colours

Colours for fonts, backgrounds, borders, etc. can be set in three ways: indexed, aRGB or theme. Indexed colours are the legacy implementation and the colours themselves depend upon the index provided with the workbook or with the application default. Theme colours are useful for complementary shades of colours but also depend upon the theme being present in the workbook. It is, therefore, advisable to use aRGB colours.

### aRGB colours

RGB colours are set using hexadecimal values for red, green and blue.

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

The alpha value refers in theory to the transparency of the colour but this is not relevant for cell styles. The default of 00 will be prepended to any simple RGB value:









































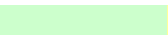

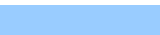



















```
>>> from openpyxl.styles import Font
>>> font = Font(color="00FF00")
>>> font.color.rgb
'0000FF00'
```

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

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

## Indexed Colours

### Standard Colours

Index					
0-4	00000000	00FFFFFF	00FF0000	0000FF00	000000FF
					
5-9	00FFFF00	00FF00FF	0000FFFF	00000000	00FFFFFF
					
10-14	00FF0000	0000FF00	000000FF	00FFFF00	00FF00FF
					
15-19	0000FFFF	00800000	00008000	00000080	00808000
					
20-24	00800080	00008080	00C0C0C0	00808080	009999FF
					
25-29	00993366	00FFFFCC	00CCFFFF	00660066	00FF8080
					
30-34	000066CC	00CCCCFF	00000080	00FF00FF	00FFFF00
					
35-39	0000FFFF	00800080	00800000	00008080	000000FF
					
40-44	0000CCFF	00CCFFFF	00CCFFCC	00FFFF99	0099CCFF
					
45-49	00FF99CC	00CC99FF	00FFCC99	003366FF	0033CCCC
					
50-54	0099CC00	00FFCC00	00FF9900	00FF6600	00666699
					
55-60	00969696	00003366	00339966	00003300	00333300
					
60-63	00993300	00993366	00333399	00333333	
					

The indices 64 and 65 cannot be set and are reserved for the system foreground and background colours respectively.

## 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 be 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")
```

## Styling Merged Cells

The merged cell behaves similarly to other cell objects. Its value and format is defined in its top-left cell. In order to change the border of the whole merged cell, change the border of its top-left cell. The formatting is generated for the purpose of writing.

```

>>> from openpyxl.styles import Border, Side, PatternFill, Font, GradientFill,
Alignment
>>> from openpyxl import Workbook
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>> ws.merge_cells('B2:F4')
>>>
>>> top_left_cell = ws['B2']
>>> top_left_cell.value = "My Cell"
>>>
>>> thin = Side(border_style="thin", color="000000")
>>> double = Side(border_style="double", color="ff0000")
>>>
>>> top_left_cell.border = Border(top=double, left=thin, right=thin,
bottom=double)
>>> top_left_cell.fill = PatternFill("solid", fgColor="DDDDDD")
>>> top_left_cell.fill = fill = GradientFill(stop=("000000", "FFFFFF"))
>>> top_left_cell.font = Font(b=True, color="FF0000")
>>> top_left_cell.alignment = Alignment(horizontal="center",
vertical="center")
>>>
>>> wb.save("styled.xlsx")

```

## Using number formats

You can specify the number format for cells, or for some instances (ie datetime) it will automatically format.

```

>>> 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'
>>>
>>> ws["A2"] = 0.123456
>>> ws["A2"].number_format = "0.00" # Display to 2dp

```

## 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
```

## Named Styles

In contrast to Cell Styles, Named Styles are mutable. They make sense when you want to apply formatting to lots of different cells at once. NB. once you have assigned a named style to a cell, additional changes to the style will **not** affect the cell.

Once a named style has been registered with a workbook, it can be referred to simply by name.

## Creating a Named Style

```
>>> from openpyxl.styles import NamedStyle, Font, Border, Side
>>> highlight = NamedStyle(name="highlight")
>>> highlight.font = Font(bold=True, size=20)
>>> bd = Side(style='thick', color="000000")
>>> highlight.border = Border(left=bd, top=bd, right=bd, bottom=bd)
```

Once a named style has been created, it can be registered with the workbook:

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

But named styles will also be registered automatically the first time they are assigned to a cell:

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

Once registered, assign the style using just the name:

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

# Using builtin styles

The specification includes some builtin styles which can also be used. Unfortunately, the names for these styles are stored in their localised forms. openpyxl will only recognise the English names and only exactly as written here. These are as follows:

- 'Normal' # same as no style

## Number formats

- 'Comma'
- 'Comma [0]'
- 'Currency'
- 'Currency [0]'
- 'Percent'

## Informative

- 'Calculation'
- 'Total'
- 'Note'
- 'Warning Text'
- 'Explanatory Text'

## Text styles

- 'Title'
- 'Headline 1'
- 'Headline 2'
- 'Headline 3'
- 'Headline 4'
- 'Hyperlink'
- 'Followed Hyperlink'
- 'Linked Cell'

## Comparisons

- 'Input'
- 'Output'
- 'Check Cell'



- 'Good'
- 'Bad'
- 'Neutral'

## Highlights

- 'Accent1'
- '20 % - Accent1'
- '40 % - Accent1'
- '60 % - Accent1'
- 'Accent2'
- '20 % - Accent2'
- '40 % - Accent2'
- '60 % - Accent2'
- 'Accent3'
- '20 % - Accent3'
- '40 % - Accent3'
- '60 % - Accent3'
- 'Accent4'
- '20 % - Accent4'
- '40 % - Accent4'
- '60 % - Accent4'
- 'Accent5'
- '20 % - Accent5'
- '40 % - Accent5'
- '60 % - Accent5'
- 'Accent6'
- '20 % - Accent6'
- '40 % - Accent6'
- '60 % - Accent6'
- 'Pandas'

For more information about the builtin styles please refer to the

`openpyxl.styles.builtins`