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='EE1111', end_color='EE1111'))
>>> 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('AA0000'), Color('00AA00')]
>>> cs2 = ColorScale(cfvo=[first, last], color=colors)
>>> # a three color scale would extend the sequences
>>> mid = FormatObject(type='num', val=40)
>>> colors.insert(1, Color('00AA00'))
>>> cs3 = ColorScale(cfvo=[first, mid, last], color=colors)
>>> # create a rule with the color scale
>>> from openpyxl.formatting.rule import Rule
>>> rule = Rule(type='colorScale', colorScale=cs3)
```

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=None, reverse=None)
>>> # 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, reverse=None)
```

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="638EC6", showValue=None,
minLength=None, maxLength=None)
>>> # assign the data bar to a rule
>>> from openpyxl.formatting.rule import Rule
>>> rule = Rule(type='dataBar', dataBar=data_bar)
```

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.styles.differential import DifferentialStyle
>>> from openpyxl.formatting.rule import ColorScaleRule, CellIsRule,
FormulaRule
>>>
>>> wb = Workbook()
>>> ws = wb.active
>>>
>>> # Create fill
>>> redFill = PatternFill(start_color='EE1111',
...                end_color='EE1111',
...                fill_type='solid')
>>>
>>> # Add a two-color scale
>>> # Takes colors in excel 'RRGGBB' style.
>>> ws.conditional_formatting.add('A1:A10',
...             ColorScaleRule(start_type='min', start_color='AA0000',
...                            end_type='max', end_color='00AA00')
...                            )
>>>
>>> # Add a three-color scale
>>> ws.conditional_formatting.add('B1:B10',
...                ColorScaleRule(start_type='percentile', start_value=10,
start_color='AA0000',
...                               mid_type='percentile', mid_value=50,
mid_color='0000AA',
...                               end_type='percentile', end_value=90,
end_color='00AA00')
...                                 )
>>>
>>> # Add a conditional formatting based on a cell comparison
>>> # addCellIs(range_string, operator, formula, stopIfTrue, wb, font, border,
fill)
>>> # Format if cell is less than 'formula'
>>> ws.conditional_formatting.add('C2:C10',
...             CellIsRule(operator='lessThan', formula=['C$1'],
stopIfTrue=True, fill=redFill))
>>>
>>> # Format if cell is between 'formula'
>>> ws.conditional_formatting.add('D2:D10',
...             CellIsRule(operator='between', formula=['1','5'],
stopIfTrue=True, fill=redFill))
>>>
>>> # Format using a formula
>>> ws.conditional_formatting.add('E1:E10',
...             FormulaRule(formula=['ISBLANK(E1)'], stopIfTrue=True,
fill=redFill))
>>>
>>> # Aside from the 2-color and 3-color scales, format rules take fonts,
borders and fills for styling:
```

```
>>> myFont = Font()
>>> myBorder = Border()
>>> ws.conditional_formatting.add('E1:E10',
...             FormulaRule(formula=['E1=0'], font=myFont, border=myBorder,
fill=redFill))
>>>
>>> # Highlight cells that contain particular text by using a special formula
>>> red_text = Font(color="9C0006")
>>> red_fill = PatternFill(bgColor="FFC7CE")
>>> dxf = DifferentialStyle(font=red_text, fill=red_fill)
>>> rule = Rule(type="containsText", operator="containsText",
text="highlight", dxf=dxf)
>>> rule.formula = ['NOT(ISERROR(SEARCH("highlight",A1)))']
>>> ws.conditional_formatting.add('A1:F40', rule)
>>> wb.save("test.xlsx")
```

Formatting Entire Rows

Sometimes you want to apply a conditional format to more than one cell, say a row of cells which contain a particular value.

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

We want to highlight the rows where the developer is Microsoft. We do this by creating an expression rule and using a formula to identify which rows contain software developed by Microsoft.

```
>>> red_fill = PatternFill(bgColor="FFC7CE")
>>> dxf = DifferentialStyle(fill=red_fill)
>>> r = Rule(type="expression", dxf=dxf, stopIfTrue=True)
>>> r.formula = ['$A2="Microsoft"']
>>> ws.conditional_formatting.add("A1:C10", r)
```

Note

The formula uses an **absolute** reference to the column referred to, `B` in this case; but a **relative** row number, in this case `1` to the range over which the format is applied. It can be tricky to get this right but the rule can be adjusted even after it has been added to the worksheet's conditional format collection.